Netfabb Automation

Lua Scripting

Reference manual

Version: March 2018

Table of Content

1 Lua in Netfabb

Lua is Netfabb's method of choice to facilitate automation tasks inside the software, namely as scripting language for Autodesk Netfabb Ultimate.

This document contains the following topics:

- a Lua scripting reference for all general Netfabb-specific extensions
- Lua calls specific to the Slice Commander
- and Lua calls for the addon 3S of Netfabb

1.1 Supported Lua functionality

Netfabb supports generally the version 5.1 of the Lua language definition, see: http://www.lua.org/manual/5.1/

From the Lua 5.1 standard Netfabb supports the core language specification and some but not all of the libraries.

Lua Library	Supported	Comment
Base	Yes, with exceptions	Some execptions: • dofile:no – use system:executescriptfile() • print: no: use system:log()
Coroutine Manipulation	Yes	
Modules	No	Use system:executescriptfile()
String manipulation	Yes	
Table manipulation	Yes	
Mathematical Functions	Yes	
Input and output Facilities	No	
Operating System Facilities	No	
The Debug library	No	

1.2 Lua Scripting in Autodesk Netfabb Ultimate 2018.3

One can access the scripting cababilities of Autodesk Netfabb only in the Ultimate Edition. One can execute general Lua scripts in the main Lua Automation Module in the default module of Autodesk Netfabb and special scripts working on slices in the Slice Commander of Autodesk Netfabb. In addition, the 3S Module can also execute Lua scripts. However, we refer to the Manual for a description of that feature.

1.2.1 Main Lua Automation Module

The Lua Automation Module module can be found under **Prepare > Execute Lua Script**. In the Lua Automation module a Lua script can be loaded, saved and manually edited. The syntax can be checked before the script is executed. All the main Lua commands can be accessed within the script. This includes the general Lua commands and commands regarding slices, however excludes the special Lua commands for the 3S Module. The Lua Automation module can be used without any connections to the current content of the Desktop application by loading, manipulation and saving again an arbitrary FABBPROJECT. However, in most cases one wants to manipulate the content of the current project opened in Autodesk Netfabb. Therefore, the variable “tray” is predefined with the current build room. This allows a direct and fast access to all meshes in the build room and allows to manipulate them. Several examples scripts are availiable.

Scriptname	Description
DesktopLUAScript_example1_MeshSimpleOperations	Loads and manipulates the current tray from the desktop aplication
DesktopLUAScript_example2_fabbproject	Loads and manipulates a fabbproject independently from the Desktop Appplication
DesktopLUAScript_example3_PackTray	Packs the current build room
DesktopLUAScript_example4_AddMeshToTray	Loads and adds a mesh to the current tray
DesktopLUAScript_example5_TrayRepairIfNeeded	Goes through the meshes in the tray and repair them if its necessary

DesktopLUAScript_example6_LoadFromFolderIntoTray	Adds files from a directory to the current trayDesktopLUAScript_example1_MeshSimpleOperations

1.2.2 Lua Execution in Slice Commander

In the slice commander a right click on the Slice opens the Pop-Up menue. Under Extended the entry “Execute Lua script” can be found. Here a script can be directly loaded and immediately executed. The variable “slice” is predefined and corresponds to the selected slice(s). For each selected slice the script is executed. Please note that any changes needs to returned with a new slice using *system:addslicetotree*. The script “SliceCommanderLUAScript_example1_Offset” demonstrates this.

2 Scripting Reference

2.1 General Lua Syntax

The Lua Scripting Language is a compact bytecode programming language, capable of most common language constructions. While including a variety of functionally for example for mathematical calculations or string manipulations, its automatic conversion of data-types guarantees a steep learning curve and quick results. Therefore, it is also widely used in a lot of applications (e.g. Apache HTTP Server, Adobe Photoshop, Video Lan Client) and especially in commercial games (e.g. Baldur's Gate, World of Warcraft, Crysis). A detailed reference and a general introduction can be found on the Lua website (***http://www.lua.org***). A very comprehensive and recommended book is "Programming in Lua", which is also available on the Web (***http://www.lua.org***). There is also a vivid community, which provides a large source for information and examples.

Netfabb's functions are included by Lua's basic object interface. While not delivering many sophisticated techniques like creating inheritances of classes, there is enough functionality to create a good interface for three-dimensional mesh handling. As a general rule, all occurring properties of object instances can be used like generic variables in the code. A simple access by "**object.*property***" is sufficient. Object methods are handled somehow uncommon, since they are characterized by a colon, so the general syntax is always like

result = object:method (param1, param2, param3); *- - generic method call*

2.2 Reference: System object

The central connection between Netfabb and the executed Lua script is the system object. As a global variable, it is accessible from anywhere in the script and is the starting point for the creation of all other objects - like meshes, xml files or database connections.

2.2.1 Properties

Property	Read / Write	Type	Description
unixtime	read only	number	returns the current time as unix timestamp
timer	read only	number	returns the number of seconds since the program has started
isquiet	read only	boolean	returns if the logging to stdout is disabled
paramcount	read only	number	returns the number of command line parameters
majorversion	read only	number	returns the major version of the Netfabb (e.g. 9)
minorversion	read only	number	returns the minor version of the Netfabb (e.g. 0)
subversion	read only	number	returns the sub-version of the Netfabb (e.g. 0)

versionstring	read only	String	returns the version string of Netfabb (e.g. '9.0.0')
result	read only	Mesh	Resulting Mesh object
Exitcode	read only	number	Exit code
Linebreak	read only	string	Line break characters
buildversionstring	read only	string	Version string

2.2.2 'system' method overview

2.2.2.1 System and file functions

Name	Syntax	Description
calculatemd5	Md5 = system:calculatemd5(filename:string);	Calculates MD5 hash sum of a file
checklibrary	Result = system:checklibrary(libname:string);	Checks for the desired library
cleargarbage	system:cleargarbage();	Calls the garbage collector
createdirectory	Result = system:createdirectory(dirname:string);	Creates a directory
createuuid	UUID = system:createuuid();	Return a UUID
directoryexists	boolean = system:directoryexists(dirname:string);	Returns TRUE, when directory exists
getallfilesindirectory	xmlfilelist = system:getallfilesindirectory (dirname:string);	Returns a xml file with all files in a directory
executescriptfile	system:executescriptfile(scriptname:string);	Executes another Lua Script
excludepathdelimiter	String = system:excludepathdelimiter(inputstring:string);	Returns string excluding the system-specific path delimiter (at the end)
extractfilename	Filename = system:extractfilename(inputstring:string);	Returns the filename only from a string
extractfileext	Filenameext = system:extractfilenameext(inputstring:string);	Returns the file name extension from a string
fileexists	boolean = system:fileexists(filename:string);	Returns TRUE, when file exists
getdatestring	date = system:getdatestring();	Returns current date as a string
getfilesize	Size = system:getfilesize(filename:string);	Returns the size of a file
getparam	P = system:getparam(index:integer);	Returns system parameter "*index*" of the script
gettimestring	time = system:gettimestring();	Returns current time as a string
includepathdelimiter	String = system:includepathdelimiter(inputstring:string);	Returns string with system-specific path delimiter (at the end)
extractfilepath	String = system:extractfilepath(inputstring; trailingpathdelimiter: bool)	Returns a string with the file path. If trailingpathdelimiter is true the trailing path delimiter are included, otherwise not

logtofile	system:logtofile(filename:string);	Log Standard Output to file
safealphanumeric	Result = system:safealphanumeric(inputstring:string;withpunctuationsigns:boolean);	Checks and returns a string for standard alphanumeric signs. Boolean flag = TRUE allows also punctuation signs ('.',',',-')
shellexecute	system:shellexecute(cmd:string, param:string, doWait:boolean, ShowState:int, directory:string)	Executes a shell command "*cmd*" with the shell parameters "param" doWait: if TRUE, waits until the command has finished ShowState: standard Shellexecute Showstate, defaults to SW_HIDE directory: change to this directory for command execution Example/note for windows: system:shellexecute("cmd", "/c copy cube.stl test.stl",1, 1);
showsavedialog	filename_to_save = system:showsavedialog(ext:string, [ext2: string]);	Open file save dialog with parameter for the file extension, returns filename. The second, optional parameter suggests a filename for the save.
createbatchslicer	Result = system:createbatchslicer();	Create a slicer object that works without user interface (see 39)
createstamper	Result = system:createstamper();	Create a LUAStamper, which allows to label meshes
createimageprocessing	Ip = system:createimageprocessing();	Creates an image processing object.
createstringmap	Map = system:createstringmap()	Create a string map for key / value pairs
Messagedlg	system:messagedlg(message: String);	Show a message dialog
Yesnodlg	Result = system:yesnodlg(message: String, [withCancel: boolean]);	Displays a “Yes/No” dialog with “message” as text. The optional “withCancel option determines, whether also a “Cancel” option should be shown. Result: 1 for Yes, 0 for No, -1 for other.
Inputdlg		
Showprogressdlg	system:showprogressdlg(defaultcallback: Boolean);	Shows the progress dialog
Hideprogressdlg	System: hideprogressdlg();	Hides the progress dialog created by system:showprogressdlg

Setprogress	system:setprogress(Percent, Message, [translate]);	Set the progress dialog with a progress value and a message. The third parameter (optional) selects whether or not the message should be translated (true = default)
slicemesh	system:slicemesh(mesh, lazersiye, fromZ, toZ, createInMemory);	Slices a mesh with the given parameters. Function returns a LUASlice Object
setloggingtooglwindow	system:setloggingtooglwindow(value);	Available for the Lua Automation module. Allows that all output is piped to the OGL warning window. *Value* is a boolean
formatlength		
formatarea		
formatvolume		
formattraffic		
registerextension		
showopendialog		

showdirectoryselectdialog	Directorystring = system: showdirectoryselectdialog(bAllowcreate:boolean, bPerformCreate:boolean, bPrompt:boolean)	Opens a directory selection dialog. Parameters: **bAllowcreate**: An edit box allows the user to type in the name of a directory that does not exist. This option does not create a directory: the application must read the name of the selected directory and create it if desired. **bPerformCreate**: Used only in combination with bAllowCreate. If the user enters a directory name that does not exist, the directory selection dialog creates it. **bPrompt**: Used only in combination with bAllowCreate. Displays a message box that informs the user when the entered directory does not exist and asks if the directory should be created. If the user chooses OK, the directory is created if the option set includes bPerformCreate. If the option set does not include bPerformCreate, the directory is not created: the application must read the directory name and create it. Returns string with directory name or empty, if nothing selected
tonumber_safe		
openurl	system:openurl(zipfile: string)	Open a URL in the default browser
createzip	zipobject = system:createzip(zipfile: string)	Creates a new zipfile object. If used without zifile parameter it creates the object in memory only. See Zipobject.

Examples

Name	Example	Return value
log	system:log("*Hello World*");	-

logtofile	system:logtofile("*my_file.log*");	-
getparam	filename_to_process = system:getparam(0);	*string*: string containing the desired parameter
shellexecute	system:shellexecute("*mail*", -s subject "*info@netfabb.com*")	-
checklibrary	system:checklibrary("*mysql*");	-
showsavedialog	local filename = system:showsavedialog ("wrl");	String: with the file name, or empty string

2.2.2.2 Mesh Loading and Mesh Creation

Name	Syntax	Description
createheightmapmesh	Mesh = system:createheightmapmesh(filename:string; width:integer, height:integer; depth:integer; invert:boolean; mingray:integer);	Creates a mesh from a JPG, PNG or BMP heightmapfile in the defined dimension. The mingray defines the minimal grey value between 0 and 255. Invert defines, whether the image should be inverted
createimagemesh	Mesh = system:createimagemesh(filename:string; width:integer, height:integer; depth:integer; treshhold:integer);	Creates a mesh from a JPG, PNG or BMP file in the defined dimension. The threshold is the defined grey value for separation between 0 and 255.
createtextmesh	Mesh=system:createtextmesh(text : string, width:integer, height:integer; depth:integer)	Creates a mesh from a string. The x,y,z size is given by width, height and depth.
createmesh	system:createmesh();	Creates a mesh with no triangles
load3ds	system:load3ds(filename:string);	Loads an 3DS file
load3mf	system:load3mf(filename:string);	Loads an 3MF file
loadamf	system:loadamf(filename:string);	Loads an AMF file
loadgts	system:loadgts(filename:string);	Loads an GTS file
loadncm	system:loadncm(filename:string);	Loads an NCM file
loadobj	system:loadobj(filename:string);	Loads an OBJ file
loadply	system:loadply(filename:string);	Loads an PLY file
loadstl	system:loadstl(filename:string);	Loads an STL file
loadvoxel	system:loadvoxel(filename:string);	Loads an SVX file
loadvrml	system:loadvrml(filename:string);	Loads an VRML file
loadx3d	system:loadx3d(filename:string);	Loads an X3D file
loadzpr	system:loadzpr(filename:string);	Loads an ZPR file

Examples

Name	Example	Return value

createmesh	mesh = createmesh();	*mesh object*: an empty mesh object
load3ds	system:loadgts("*test.3ds*");	*mesh object*: a mesh object of the "Autodesk 3D Modeling Format" file
load3mf	system:load3mf("*test.3mf*");	*mesh object*: a mesh object of the "3MF Basic Microsoft" file
loadamf	system:loadamf("*test.amf*");	*mesh object*: a mesh object of the "Additive Manufacturing File"
loadgts	system:loadgts("*test.gts*");	*mesh object*: a mesh object of the "Gnu Tesselated Surfaces" file
loadncm	system:loadncm("*test.ncm*");	*mesh object*: a mesh object of the "Netfabb Compressed Mesh" file
loadobj	system:loadobj("*test.obj*");	*mesh object*: a mesh object of the "Wave Front OBJ" file
loadply	system:loadply("*test.ply*");	*mesh object*: a mesh object of the "Stanford Polygon" file
loadstl	system:loadstl("*test.stl*");	*mesh object*: a mesh object of the "Surface Tesselation Language" file
loadvoxel	system:loadvoxel("*test.svx*");	*mesh object*: a mesh object of the "Simple Voxels" file
loadvrml	system:loavrml("*test.vrml*");	*mesh object*: a mesh object of the "Virtual Reality Modeling Language" file
loadx3d	system:loadx3d("*test.x3d*");	*mesh object*: a mesh object of the "Extensible 3D-ASCII" file

2.2.2.3 XML, Text Handling and Database connections

Name	Syntax	Description
createtextfile	textfile = system:createtextfile();	Creates an empty text file
createxml	Xmlfile = system:createxml();	Creates an empty XML file
loadtextfile	textfile = system:loadtextfile(filename:string);	Loads an text file from disk
loadxml	Xmlfile = system:loadxml(filename:string);	Loads an XML file from disk
loadxmlfromurl	Xmlfile = system:loadxml(URL:string);	Loads an XML file from an URL
connecttoodbc	DBConnection = system:connecttoodbc(connectionstring (DSN), login, password : all strings);	Connects to an ODBC datasource

Examples

Name	Example	Return value

createxml	system:createxml();	*XML object:* XML file object for further processing
loadxml	system:loadxml("i*nput.xml*");	*XML object:* XML file object for further processing
connecttoodbc	system:connecttoodbc("netfabb", "user", "pw");	*DB Object*: Database object for further processing

2.2.3 Further functionality

2.2.3.1 GL Context

Name	Syntax	Description
createoglcontext	GLContext = system:createoglcontext(width: integer; heigth: integer; AXServer: string)	Creates GL Context with xidth x height size. AXServer is an optional argument

2.2.3.2 Testsuite Framework

Name	Syntax	Description
createtestsuite	subsuite = system:createtestsuite(suitename: string)	Creates a new testsuite with specified name.

2.2.3.3 CadImport

Name	Syntax	Description
createtecadimport	cadimporter = system:createcadimport()	Creates a new Importer for CAD Files

2.2.3.4 Webgl Export

Name	Syntax	Description
createwebglexport()	webglexport = system:createwebglexport();	Creates webgl context

2.2.3.5 Fabbproject

Name	Syntax	Description
loadfabbproject()	fabbproject = system:loadfabbproject(filename: String);	Open Fabbproject
newfabbproject()	Newfabbproject = system:newfabbproject()	Create new and empty Fabbproject

2.3 Reference: Mesh Objects

A mesh object contains the full information about a triangulated mesh, and may be used for altering, transforming and exporting the mesh data.

2.3.1 Properties

Property	Read / Write	Type	Description
area	read only	Number	Returns the area
calculationerror	read only	Boolean	Returns whether the last calculation has failed
calculationfailed	read only	String	Returns the message for a failed calculation
edgecount	read only	Number	Returns the number of edges
facecount	read only	Number	Returns the number of triangles
holecount	read only	Number	Returns the number of holes
ismanifold	read only	Boolean	True, if the mesh has a closed surface
isnice	read only	Boolean	Calculates if the mesh is manifold (Deprecated)
isorientable	read only	Boolean	True, if flipped triangles can be repaired
isoriented	read only	Boolean	True, if the mesh has no flipped triangles
isok	read only	Boolean	True, if the mesh is a) 'ismanifold', b) 'isoriented' and c) has a positive volume
loadingfailed	read only	Boolean	Returns, whether a load has failed
loadingerror	read only	String	Returns the loading error
nodecount	read only	Number	Returns the number of nodes/points
facecount	read only	Number	Number of Faces (Triangles)
shellcount	read only	Number	Returns the number of shells
volume	read only	Number	Returns the volume (in mm^3)
outboxvolume	read only	Number	Returns the mesh outbox volume (in mm^3)
outboxbasearea	read only	Number	Returns the mesh outbox base area (in mm^2)
outboxheight	read only	Number	Returns the mesh outbox height (in mm)

2.3.2 Method Overview

2.3.2.1 Format conversion

Name	Syntax	Description
savetostl	mesh:savetostl(filename:string);	Saves the mesh as a binary STL file
savetoasciistl	mesh:savetoasciistl(filname:string, name:string);	Saves the mesh as a ASCII STL file

savetox3d	mesh:savetox3d(filename:string, binary:boolean);	Saves the mesh as a X3D file
savetogts	mesh:savetogts(filename:string);	Saves the mesh as a GTS file
savetoobj	mesh:savetoobj(filename:string);	Saves the mesh as a OBJ file
savetoncm	mesh:savetoncm(filename:string);	Saves the mesh as a NCM file
savetoamf	mesh:savetoamf(filename:string;binary:boolean);	Saves the mesh as a AMF file
savetoply	mesh:savetoply(filename:string);	Saves the mesh as a PLY file
saveto3ds	mesh:saveto3ds(filename:string);	Saves the mesh as a 3DS file
saveto3mf	mesh:saveto3mf(filename:string);	Saves the mesh as a 3MF file
savetovrml	mesh:savetovrml(filename:string;fixlengtheachrow:boolean; endmarkeachrow:boolean);	Saves the mesh as a VRML file
savetovoxel	mesh:savetovoxel(filename:string);	Saves the mesh as a Voxel file
savetozpr	mesh:saveto3ds(filename:string);	Saves the mesh as a ZPR file
shellasmesh	Newmesh = mesh:shellasmesh(shellnumber: Number)	Extract a shell as new mesh object. Same as "LUATrayMesh.shellasmesh" but without the matrix of a traymesh. See "LUAMeshObject.shellcount" for the number of meshes

All routines return TRUE if successful.

Examples

Name	Syntax	Return value
savetostl	mesh:savetostl("*my_mesh_fixed.stl*");	TRUE if successful
savetoasciistl	mesh:savetoasciistl("*my_fixed_ascii.stl*", "*My fixed ASCII STL*");	TRUE if successful
savetox3d	mesh:savetox3d("*my_mesh_fixed.x3d*", true);	TRUE if successful
savetogts	mesh:savetogts("*my_mesh_fixed.gts*");	TRUE if successful
savetoobj	mesh:savetoobj("*my_mesh_fixed.obj*");	TRUE if successful
savetoncm	mesh:savetoncm("*my_mesh_fixed.ncm*");	TRUE if successful

2.3.2.2 Mesh Transformation

Name	Syntax	Description
movetoorigin	mesh:movetoorigin();	Translates the mesh to the origin
move	mesh:move(x:integer, y:integer, z:integer);	Translates the mesh by a given vector
scale	mesh:scale(x:integer, y:integer, z:integer);	Scales the mesh by a given factor
rotate	mesh:rotate(x:integer, y:integer, z:integer, angle:integer);	Rotates the mesh by a given angle around a given axis

calcoutbox	Outbox = mesh:calcoutbox();	Calculates the outbox of a mesh, return Outbox object.
invert	mesh:invert();	Inverts the orientation of all triangles
release	mesh:release();	Releases the memory of a mesh. It cannot be accessed after this method
dupe	Newmesh = mesh:dupe();	Creates a copy of the mesh
merge	mesh:merge(anothermesh:string);	Adds another mesh into the mesh
addtriangle	Mesh:addtriangle(P1_X, P1_Y, P1_Z, P2_X, P2_Y, P2_Z, P3_X, P3_Y, P3_Z: all number);	Adds a triangle to a mesh
minimizeoutbox	Mesh:minimizeoutbox(mode : string);	Rotates the mesh to minimize the volume or the height and the base area of the mesh outbox. Permitted values of the parameter mode are Volume – the volume of the outbox is minimized; VolumeFlat - the volume of the outbox is minimized and the mesh is oriented in such a way that the shortest edges of the outbox are parallel to the axis z. HeightBase – the height of the outbox is minimized then the outbox base area is minimized at the constant outbox height (i.e. by a rotation around the axis z).
zcompensation	Mesh:zcompensation(value : integer);	Applies Z-Compensation to Mesh, Value in mm.
create_partorienter	orienter = mesh:create_partorienter()	Creates and returns new orienter object for this mesh, see PartOrienter

Examples

Name	Example	Return value
movetoorigin	mesh:movetoorigin();	-
move	mesh:move(*200, 50, -100*);	-
scale	mesh:scale(*0.01*); *[if y and z are not specified, x value is taken]*	-
rotate	mesh:rotate(*1, 2, 0.5, 90*);	-
calcoutbox	my_outbos = mesh:calcoutbox();	*Outbox object*: object representing the outbox of the mesh
invert	mesh:invert();	-
release	mesh:release();	-
dupe	mesh:dupe();	*Mesh object*: returns a copy of the mesh. **Attention**: may use a lot of memory
merge	mesh:merge(*mesh_of_inner_shell*);	-

2.3.2.3 Analyze meshes

Name	Syntax	Description
wallthicknesstest	Boolean result = mesh:wallthicknesstest(criticaldistance:number, criticalsurface:number in %; colormesh: boolean);	Checks the wallthickness of a mesh. Returns true if the wallthickness is not smaller than criticaldistance for more than the criticalsurface area. If colormesh is true, the results of the wallthickness will be written into the color information of the mesh.
fastwallthicknesstest	Boolean result = mesh:fastwallthicknesstest(criticaldistance:number, criticalsurface:number in %;colormesh: boolean);	Checks the wallthickness of a mesh, as above, but stops after fail condition is recognized. If colormesh is true, the results of the wallthickness will be written into the color information of the mesh.
comparewith	mesh:comparewith(mesh2:string, hausdorffdistancethreshold:number, distancethreshold:number, fraction:number in %);	Checks if the mesh is different to mesh2. This is done in two steps: if the haussdorffdistance is larger than the given *hausdorffdistancethreshold* the tests fails or a given *fraction* of the measured distances is larger the *distancethreshold*
calculatehausdorffdistanceto	mesh:calculatehausdorffdistanceto(mesh2:string);	Returns the Hausdorffdistance

checksanity	mesh:checksanity(mesh2:string);	Returns TRUE if the mesh is sane
getsupportvolume	mesh:getsupportvolume(criticleangle:number);	Returns the volume of the support in mm³. The area, which needs support, is calculated using the parameter *criticalangle*. The normal of a triangle with respect to the z-axis defines the critical angle, i.e. a small critical angle means that only a small area needs support. If the calculation fails, -1 is returned.
getbuildvolume	mesh:getBuildVolume(offset:number)	Returns the value of the inner buildvolume in mm³. The parameter *offset* gives the thickness of the surface. If the calculation fails, -1 is returned.
gettotalcontourlength	mesh:gettotalcontourlength(layersize)	Returns the value of the length of all outer contourlengts in mm. The parameter *layersize* gives the thickness of the layersize. If the calculation fails, -1 is returned.
getcenterofgravity	Center = mesh:getcenterofgravity();	Returns the center of gravity of the mesh as LUAVector3
createalignment	TLUAAlignment alignment = mesh:createalignment(mesh2:meshobject, matrix1, matrix2: luamatrix)	Creates a new alignment object which stores the principal axes and the resulting matrix. See 4.28 for a detailed property list mesh2: the mesh that can be transformed with the resulting matrix to overlap the given mesh. Matrix1, matrix2: optional parameters, used as modeltoword matrices when set

Examples

Name	Example	Return value
wallthicknesstest	wallThicknessPassed = mesh:wallthicknesstest(*0.4, 20*);	*boolean*: returns **true** if the area which has a thinner wallthickness as *criticaldistance* [mm] is less than *criticalsurface* [%] of the total area.
comparewith	testresult = mesh:comparewith(*mesh2, 0.4, 0.1, 10*);	*integer :* For each point of mesh the closest

		distance to mesh2 is calculated. This distribution of distances is afterward analyzed. The distribution is used for two tests. **Test1**: If only distance is larger than *distance* test1 fails. **Test2**: if a *fraction* of the distance distribution is larger than *threshold* test2 fails. Return 0 if meshes are equal, returns 2 if test1 fails, returns 4 if test2 fails, and returns 6 is both test fails.
calculatehausdorffdistanceto	hasudorffdistance = mesh:calculatehausdorffdistanceto(*mesh2*);	*number*: For each point of the mesh the closest distance to mesh2 is calculated. The maximum of these distances, which called Hausdorffdistance, is returned.
getdownskinarea	Areasum = mesh:getdownskinarea(*downskinangle: number*);	Calculates the downskin area
getupskinarea	Areasum = mesh:getupskinarea(*downskinangle: number*);	Calculates the upskin area
wallthicknessadvtest		Internal

Property	Read / Write	Type	Description
boundaryedgecount	read	Number	Get the number of boundary edges of the mesh
badedgecount	read	Number	Get the number of bad edges of the mesh
boundarylength	read	Number	Get the length of the boundary of the mesh
flippedtriangles	read	Number	Get the number of flipped triangles of the mesh
shadowarea	read	Number	Get the shadow area of the mesh

2.3.2.4 Boolean Operations

Name	Syntax	Description

unify	mesh:unify(epsilon:number);	Unifies shells of a nice mesh. This is actually the 3d boolean function of Netfabb. This does the actual operation of fusing 2 meshes. As an "add" function, if you just merge 2 meshes, as a "substract" operation, when you invert one mesh before. The epsilon parameter is syntactically required, but internally ignored (this was originally a precision parameter, but that has been changed with a scaling algorithm. The syntax is for backward compatibility)
bool	Result = mesh:bool(asecondmesh: TLUAMesh; operationtype: number);	Bools the current mesh with a second. If operationtype is 0 the mesh is added, if it is 1 the mesh is subtracted.
intersect	mesh:intersect();	Works similar to the unify function only that it builds the intersect of 2 merged meshes
insertselfintersections	mesh:insertselfintersections(epsilon:number);	Splits up a mesh along its self intersections, same as: Self-intersections: Split off in Netfabb Professionals Repair actions
wrap	mesh:wrap();	Wraps a mesh. Only the outer part of a mesh is left. Uses a parallel multicore algorithm
wrapsinglecore	mesh:wrapsinglecore();	Wraps a mesh. Only the outer part of a mesh is left. Uses the old singlecore algorithm

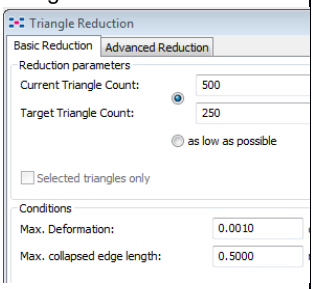
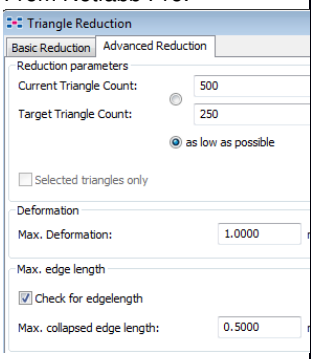
Examples

Name	Example	Return value
unify	found_self_intersections = mesh:unify(*0.05*);	*boolean:* returns if any self-intersections have been found (or if they have been disregarded due to the epsilon value). Dependent on the mesh geometry, this function could create a lot of sall shells, which should be cleared by ”removeghostshells”
intersect	found_self_intersections = mesh:intersect();	boolean: Similar to unify

insertselfintersections	found_self_intersections = mesh:insertselfintersections(*0.05*);	*boolean:* returns if any self-intersections have been found (or if they have been disregarded due to the epsilon value). **Attention**: This function may create a lot of shells, but it is capable of orienting not-orienable surfaces.
wrap	mesh:wrap()	-

2.3.2.5 Mesh Repair

Name	Syntax	Description
removedoublefaces	mesh:removedoublefaces(number);	Removes double triangles from the mesh
closetrivialholes	mesh:closetrivialholes();	Close trivial holes
closeallholes	mesh:closeallholes(number);	Closes all non-trivial holes
fixflippedtriangles	mesh:fixflippedtriangles();	Tries to create a uniform mesh orientation
invertnegativeshells	mesh:invertnegativeshells();	Inverts all shells with negative volumes
removedegeneratefaces	mesh:removedegeneratefaces(tol:number);	Removes degenerated faces
stitch	mesh:stitch(tol:number, preserveorientation:boolean);	Stitches open triangle borders
removenegativeshells	mesh:removenegativeshells();	Remove shells with negative volume
removeghostshells	mesh:removeghostshells(epsilon:number);	Remove small shells
hollow	Newmesh = mesh:hollow(offset:number, rastersize:number, smoothen:boolean);	Returns a mesh, which is by the offset smaller than the original mesh. The rastersize is the resolution. The smoothing only occurs at the resolution limit. Offset and rastersize should both be positive (in mm). In order to hollow an actual part the meshes still have to be merged.
Cut	Newmesh= mesh:cut(plane:number;height:number;topbottom:number);	Cuts the mesh enlong the x,y, or z plane. Plane 0 corresponds to the z-y plane, 1 to the z-x plane and 2 to the x-y plane. The height gives the position of the plane, topbottom can be either 1, then the top mesh will be returned or 2, then the bottom mesh will be returned. Topbottom is optional, default is that the top mesh is returned.
makeorientable	mesh:makeorientable();	Makes Mesh orientable
preparetopologyforexport	mesh:preparetopologyforexport();	Prepares topology for STL export

reducelod	mesh:reducelod(target:number, deformation:number, edgelength:number);	Reduces number of trianglesFrom Netfabb Pro:  Target: Target Triangle count Deformation: Max. Deformation (cm3) edgelength: Max. Collapsed edge length
reducelodadvanced	mesh:reducelodadvanced(count:number, deformation:number, edgelength:number);	Reduces number of triangles From Netfabb Pro:  Target: Target Triangle count Deformation: Max. Deformation (cm3) edgelength: Max. Collapsed edge length
meshnodesparsification	mesh:meshnodesparsification(Epsilon:number, DegenerationUnits:number);	Sparsificated nodes of a mesh
repairsimple	mesh:repairsimple();	Same as the Netfabb Professional "simple" repair makro script
repairenhanced	mesh:repairenhanced();	Same as the Netfabb Professional "enhanced" repair makro script
repairextended	mesh:repairextended();	Same as the Netfabb Professional "extended" repair makro script
removeproblemareas	mesh:removeproblemareas();	Removes tiny faces which are causing many self-intersections

Examples

Name	Example	Return value

removedoublefaces	removed_count = mesh:removedoublefaces();	*integer:* Returns how many triangles have been removed. Two triangles are ”double” if they share exactly the same corners. If they share slightly the same corners, this function does not harm them. Stitching and degenerate face removal couldalign these cases.
closetrivialholes	trivial_hole_count = mesh:closetrivialholes();	*integer:* Returns the number of trivially closed holes of the mesh, i.e. all missing triangles or quadrangles. This function does not alter the mesh in any other way and is safe to call anytime.
closeallholes	hole_count = mesh:closeallholes();	*integer:* Returns the number of closed holes of the mesh, i.e. all nontrivial and non-bad holes. This function can be dangerous to call and add unwanted walls into the mesh.
fixflippedtriangles	mesh_is_orientable = mesh:fixflippedtriangles();	*boolean:* returns if the full mesh has been made oriented. This does not mean that the shell as a whole has a positive volume.
invertnegativeshells	negative _shell_count = mesh:invertnegativeshells();	*integer:* number of shells inverted
removedegeneratefaces	mesh:removedegeneratefaces(*0.02*);	-
stitch	mesh:stitch(*0.01*, true);	*integer* numbers of edges having been stitched

2.3.3 Model Package

2.3.3.1 Model Package

Name	Syntax	Description
createmodelpackage	mesh:createmodelpackage(barwidth, barthickness, partspacing, gridsizexy, grindsizez: number);	Creates a model package for the current mesh. The result is the package mesh.

2.3.4 Mesh Support

The API has only one method, which uses an automated support definition, which can be generated by the Netfabb Desktop software.

Name	Syntax	Description
createsupport	support = stl:createsupport(xmlfile: string)	Creates automatic Support based on instructions in XML parameter file.

Example:

```
stl = system:loadstl('file.stl')
support = stl:createsupport('support.xml')
support:savetostl('support.stl')
```

2.3.5 LUAMeshObject and TLUATrayMesh

TLUAMeshObject

The TLUAMeshObject is the Lua representation of the Pascal "TNGM_Mesh_Advanced" class. All modification functions are applied directly to the mesh

TLUATrayMesh

The TLUATrayMesh is the Lua representation of the Pascal "TNGM_MeshTreeMesh" class. It contains a TLUAMeshObject as member "FLUAMesh" published in the Pascal property "TLUATrayMesh.Mesh" and exported to Lua with the member "LUATrayMesh.mesh". Please note that the "LUATrayMesh.mesh is only a copy of the original LUAMesh in the Lua Automation module. All modification functions of the LUATrayMesh do not modify the mesh but only its properties in the tree.

2.4 Reference: XML file Object

An XML file object contains the full information of an XML file. It can be used to read XML information from the disk as well as output information in the XML format to disk or to stdout.

2.4.1 Properties

Property	Read / Write	Type	Description
childcount	read only	Number	Returns the number of xml nodes of the root entry
rootname	read / write	String	Section name of the root entry

2.4.2 Method Overview

Name	Syntax	Description

childexists	xmlfile:childexists(nodename:string);	Returns if nodename in the XML file exists
addvalue	xmlfile:addvalue(nodename:string, value:string/int);	Adds a value to the XML file
savetofile	xmlfile:savetofile(filename:string);	Writes the XML file to disk
dump	xmlfile:dump()	Dumps the XML file on stdout. Even if the cloud utilities are in quiet mode
findchild	Childnode = xmlfile:Findchild(childname:string);	Finds child node
addchild	xmlfile:Addchild(childname:string, adddouble:boolean);	Adds child node, adddouble allows for duplicates
getchildvalue	Childvalue = xmlfile:Getchildvalue(childname:string);	Returns child value
getchildindexed	Childvalue = xmlfile:Getchildindexed(childindex:string);	Returns child value by index
getchildint	ChildIntvalue = xmlfile:Getchildint(child:string);	Gets the Integer value of child node
getchildintdef	ChildIntvalue = xmlfile:Getchildintdef(child:string;value:number);	Gets the Integer value of child or value
getchildfloat	ChildIntvalue = xmlfile:Getchildfloat(child:string);	Gets Float value of child node
getchildfloatdef	ChildIFloatvalue = xmlfile:Getchildintdef(child:string;value:number);	Gets Float value of child node or value

Examples

Name	Syntax	Description
childexists	xmlfile:childexists("*parameters/fixingmode*", 2);	*boolean:* returns **true** if the desired node exists
addvalue	xmlfile:addvalue("*parameters/fixingmode*", 2);	-
savetofile	xmlfile:savetofile("*var/log/test.xml*");	-
dump	xmlfile:dump()	-

2.5 Reference: XML node Object

An XML node object contains the full information of an XML node.

2.5.1 Properties

Property	Read / Write	Type	Description
childcount	read only	Number	Returns the number of xml nodes of the root entry
section	Read / write	String	Section name of the root entry
value	Read only	String	Value of the node

2.5.2 Method Overview

Name	Syntax	Description
childexists	node:childexists(nodename:string);	Returns if nodename in the XML file exists
addvalue	node:addvalue(nodename:string, value:string/int);	Adds a value to the XML file
getattributeint	IntValue = node:getattributeint(nodename:string);	Gets integer value of node
getattributefloat	FloatValue = node:getattributefloat(nodename:string);	Gets float value of node
findchild	Childnode = node:Findchild(childname:string);	Finds childnode
addchild	node:Addchild(childname:string, adddouble:boolean);	Adds child node, adddouble allows for duplicates
getchildvalue	Childvalue = node:Getchildvalue(childname:string);	Returns child value
getchildindexed	Childvalue = node:Getchildindexed(childindex:string);	Returns child value by index
getchildint	ChildIntvalue = node:Getchildint(child:string);	Gets the Integer value of child node
getchildintdef	ChildIntvalue = node:Getchildintdef(child:string;value:number);	Gets the Integer value of child or value
getchildfloat	ChildIntvalue = node:Getchildfloat(child:string);	Gets Float value of child node
getchildfloatdef	ChildIFloatvalue = node:Getchildintdef(child:string;value:number);	Gets Float value of child node or value

2.6 Reference: text file Object

The text file object represents a text file.

2.6.1 Properties

Property	Read / Write	Type	Description
linecount	read only	Number	Returns the number of lines

2.6.2 Method Overview

Name	Syntax	Description
loadfromfile	Result = txtfile:loadfromfile(filename:string);	Loads text file from disk, returns TRUE when successful
clear	Txtfile:clear();	Empties the textfile object
savetofile	Result = txtfile:savetofile(filename:string);	Writes textfile, TRUE, when successful
setline	Result = txtfile:setline(linenumber:number; text:string);	Writes 'text' at line number: 'linenumber', TRUE when successful
getline	resultline = txtfile:getline(linenumber:number);	Returns textline at linenumber
writeline	txtfile:writeline(text:string);	Appends 'text' to text file

2.7 Reference: Outbox Object

Represents an outbox

2.7.1 Properties

Property	Read / Write	Type	Description
minx	Read/write	Number	Returns the min_x (in mm)
Miny	Read/write	Number	Returns the min_y (in mm)
minz	Read/write	Number	Returns the min_z (in mm)
maxx	Read/write	Number	Returns the max_x (in mm)
maxy	Read/write	Number	Returns the max_y (in mm)
maxz	Read/write	Number	Returns the max_z (in mm)

2.7.2 Method Overview

None.

2.8 Reference: database connection Object

Database connection and query handler.

2.8.1 Properties

Property	Read / Write	Type	Description
connected	read only	Boolean	Returns TRUE, if the DB connection is established
lasterror	Read only	String	The last DB error

2.8.2 Method Overview

Name	Syntax	Description
sendquery	Result = dbconnection:sendquery(query:string);	Returns TRUE when successful, the query result can be evaluated wuth "getresult".
getresult	Queryresult = dbconnection:getresult();	Returns the QueryResult Object
getinsertid	insertid = dbconnection:getinsertid();	Returns the last insert id
checkiftableexists	Result = dbconnection:checkiftableexists(tablename:string);	Returns TRUE, if table with tablename exists

createttable	Result = dbconnection:createttable(tablename:string);	Creates a new table
Disconnect	dbconnection:Disconnect();	Disconnects from the DB
getuniquestring	uniquestring = dbconnection:getuniquestring();	Gets unique string

2.9 Reference: query result Object

The result object of a database query.

2.9.1 Properties

Property	Read / Write	Type	Description
valid	read only	Boolean	Returns TRUE, if the query result is valid
fieldcount	Read only	Number	Number of fields in current row

2.9.2 Method Overview

Name	Syntax	Description
getfieldcount	Result = queryresult:getfieldcount();	Returns number of fields in row
nextrow	queryresult:nextrow();	Put the result pointer to the next row
getfield	StringResult = queryresult:getfield(fieldnumber:number);	Returns the String value of the field in fieldnumber
getintegerfield	IntResult = queryresult:getintegerfield(fieldnumber:number);	Returns the Integer value of the field in fieldnumber
getfloatfield	FloatResult = queryresult:getintegerfield(fieldnumber:number);	Returns the Float value of the field in fieldnumber

2.10 Reference: OGLRendering Object

A render context for previews. The OGLRendering Object is created by calling " system:createoglcontext". The rendering object itself works with:

- models: individual meshes
- scenes: arrangements of models

2.10.1 Properties

None.

2.10.2 Method Overview

Name	Syntax	Description
savetojpeg	oglcontext:savetojpeg (filename:string, quality: integer, threaded: boolean);	Saves current scene to JPEG file as 'filename'. Quality: JPEG quality, default: 90 Threaded: optional parameter, when true the export runs in a separate thread
Savetopng	oglcontext:savetopng (filename:string, threaded: boolean);	Saves current scene to PNG file as 'filename'. Threaded: optional parameter, when true the export runs in a separate thread
Savetobmp	oglcontext:Savetobmp (filename:string, threaded: boolean);	Saves current scene to BMP file as 'filename'. Threaded: optional parameter, when true the export runs in a separate thread
lookatmodelfromsurroundingsphere	oglcontext:lookatmodelfromsurroundingsphere (modelid, eyex, eyey, eyez, upx, upy, upz, offset: all integer);	Sets the camera position eye (x,y,z) = eye vector up (x,y,z) = the UP vector describes the roll of the camera by saying which point is "up" in the camera's orientation. offset = distance to mesh object (for more information: search for opengl and eye and up vector)
rotate	oglcontext:Rotate (Modelid:integer, angle: integer);	Rotates the model across the camera vector (needs to be set up first with Lookatmodelfromsurrounding sphere)
scale	oglcontext:Scale (x,y,z:all float);	Scales the context
setlightpos	oglcontext:Setlightpos (x,y,z,type: all integer);	Sets the position of lightsource, type 1 = point lightsource, otherwise unidirectional
setBackgroundColor	oglcontext:SetBackgroundColor (r, g, b: all integer);	Sets background colour to RGB value
setBackgroundGradient	oglcontext:SetBackgroundGradient (ar, ag, ab, br, bg, bb, cr, cg, cb, dr, dg, db: all integer);	Background gradient map: RBG values for the points a, b, c and d. A = bottom left B = bottom right C = top right D = top left
setreflectivity	oglcontext:Setreflectivity (modelid: integer, reflectivity: float);	Sets the reflectivity factor (e.g. "0.6") for the model
render	oglcontext:Render ();	Renders the scene
swapbuffers	oglcontext:swapbuffers ();	OpenGL glutswapbuffers function

createmodel	Modelid:integer = oglcontext:createmodel (mesh:mesh object, loadcolors: boolean, loadtextures: boolean, matrix: tluamatrix)	Puts a mesh object into the oglrender context, returns the modelID. loadcolors: optional parameter, default is true. Flag to allow loading of vertex colors.Setting the flag to false allows the setting of a uniform color using setmodelcolor. loadtextures: optional parameter, default is true. Flag to allow loading of vertex textures. matrix: optional parameter, transforms the model and sets the transformed bounding box when set
freemodel	oglcontext:Freemodel (modelid:integer);	Remoces model from scene and deletes reference
addmodeltoscene	oglcontext:addmodeltoscene (modelid:integer);	Adds the model of 'modelID' to the current scene.
removemodelfromscene	oglcontext:Removemodelfromscene (modelid:integer);	Removes model from scene
setenvironmentmodel	oglcontext:Setenvironmentmodel (modelid:integer);	Sets the environment model
setModelColor	oglcontext:setmodelcolor(modelid: integer, r, g, b, a: float)	Changes the uniform color of a model. In order to usethis method vertex colors need to be disabled in the createmodel call.
setModelTextureEnabled	oglcontext:setmodeltextureenabled(textureenabled: boolean)	Changes the textureendabled flag
release	*oglcontext:release ();*	*DEBUG ONLY: Releases the current oglcontext (allowing to create a new one)*
Translate	*oglcontext:Translate (x,y,z:all float);*	*DEBUG ONLY: OpenGL translate function*
Multmatrix	*oglcontext:Multmatrix (modelid + 16 integer);*	*DEBUG ONLY*
Identity	*oglcontext:Identity ();*	*DEBUG ONLY: Load the OpenGL identity matrix ("glloadidentity()")*
releaserawbuffer	*oglcontext:Releaserawbuffer ();*	*DEBUG ONLY*
pushmodelmatrix	*oglcontext:Pushmodelmatrix (modelid:integer);*	*DEBUG ONLY*
popmodelmatrix	*oglcontext:Popmodelmatrix (modelid:integer);*	*DEBUG ONLY*
pushimage	*oglcontext:Pushimage ();*	*DEBUG ONLY*

2.11 Reference: Test framework interface

The Lua test framework provides functionality for uniform test suites and convenient result file formats.

2.11.1 Properties

Property	Read / Write	Type	Description
childcount	read only	Number	Number of direct sub test suites
name	read / write	String	Name of the test suite
success	read / write	Boolean	True for success, false for failure
errormessage	read / write	String	Errormessage of this test on failure
duration	read / write	Number	Duration of this test in seconds
failurecount	read	Number	Number of failed sub tests
testcount	read	Number	Number of all sub tests

2.11.2 Method overview

Name	Syntax	Description
asserttrue	suite:asserttrue(emassage: string; value: boolean)	Asserts for value = true. For false value, the test fails and sets the errormessage
assertequalsnumber	suite:assertequalsnumber(emassage: string; n1, n2, accuracy: Number)	Asserts for n1 = n2 with optional accuracy. For not equal numbers, the test fails and sets the errormessage.
assertequalsmeshproperties	suite:assertequalsmeshproperties(emassage: string; mesh1, mesh2: MeshObject)	Asserts for equal properties of the two meshes: NodeCount, EdgeCont and FaceCount. For not equal meshes, the test fails and sets the errormessage.
assertequalsmeshgeometry	suite:asserequalsmeshgeometry(emassage: string; mesh1, mesh2: MeshObject; accuracy: Number)	Asserts for equal geometry of the two meshes with meshcompare considering to the optional accuracy. For not equal meshes, the test fails and sets the errormessage.
fails	suite:fails(emassage: string)	The test fails. The parameter specifies the error message of the failed test.
finishtest	suite:finishtest()	Finishes the current test. This call calculates the duration of the test.
createtestsuite	subsuite = suite:createtestsuite(name: string)	Creates a new sub test suite for the current suite.

savetojunitxml	suite:savetojunitxml(file: string)	Saves the test result into specified file. The format is a junit convenient xml format.
savetocsv	suite:savetocsv(file: string)	Saves the test result into specified file. The format is a comma separated format.

2.12 Webservice Interface

The Lua Webservice Interface allows Lua scripts to call web services. At the moment only Stratasys Connectivity services are supported, so the class "TLUAWsClient" must be extended for new services. These methods are currently only available in the Engine context.

2.12.1 Properties

None.

2.12.2 Method Overview

Name	Syntax	Description
preparecall	wsclient:preparecall(method: string)	Prepares the call for the method passed
addparameter	wsclient:addparameter(parameter: string)	Adds a parameter for the call initialized before
call	wsclient:call()	Calls the method specified in "preparecall" with the parameters set with "addparameters". Returns true if the call succeeded, false otherwise
getresultvalue	wsclient:getresultvalue(name: string)	Returns the value of the successful call specified by name
isresultvalue	wsclient:isresultvalue(name: string)	Returns true if a result value with the given name does exist
getresultstring	wsclient:getresultstring()	Returns a string with all name-pair result values of the call, for debugging purposes
endcall	wsclient:endcall()	Ends a call. Calling this function deletes all result values. This function must be called before any other webservice functions are called
getnexterror	wsclient:getnexterror()	Gives the next error. Returns an empty string if no further erros are in the list

2.13 Reference: Image Processing

The Lua Image Processing Interface provides functionality for simple image processing and algorithms.

2.13.1 Properties

None.

2.13.2 Method Overview

Name	Syntax	Description
loadimage	Ip:loadimage(file: string)	Loads image from specified file and returns a lua-image-object

2.14 Reference: Image Object

2.14.1 Properties

Property	Read / Write	Type	Description
width	read only	Number	Width of the image
height	read only	Number	Height of the image

2.14.2 Method Overview

Name	Syntax	Description
saveto	img:saveto(file: string)	Saves image to specified file
laplace	img:laplace()	Performs the laplace filter on the image
tograyscale	img:tograyscale()	Converts colored image to grayscale
gausssmoothing	img:gausssmoothing(sigmar: number)	Performs the gausssmoothing filter with specified sigmar value
invert	img:invert()	Inverts colors of the image
detectedges	img:detectedges()	Detects edges of the image by the canny-algorithm. The resulting image is black with white one pixel wide edges.
compareto	img:compareto(img: object)	Compares the image to another specified image pixel by pixel. The result ist the percentage of equality (50% = random image, 100% = equal image).

deltato	img:deltato(img: object)	Calculates the difference between the two images. Same areas are black, difrerences are higlighted white
colortotransparent	img:colortotransparent(r, g, b: number)	Converts one color to transparency. The color is specified by three rgb parameter (0-255).
paint	img:paint(img: object)	Draw a image over another image. Transparent areas in the top image will be ignored.
setcolor	img:setcolor(r, g, b: number)	Override the color of the entire image. The color is specified by three rgb parameter (0-255). Transparency will stay the same.

2.15 Reference: 2dPackingtool Object

2D Packing is deprecated. For reference only.

2.15.1 Properties

None.

2.15.2 Method Overview

Name	Syntax	Description
resetraster	packingtool:resetraster(SizeX, SizeY, OffsetX, OffsetY: number)	Sets the size of the packing tool. Offset is optional and can be used to have a clear distance to the tray borders
markusedspace	packingtool:markusedspace(AMesh: TLUAMeshObject)	
addpackingobject	packingtool:addpackingobject(AMesh: TLUAMeshObject)	Add a mesh to the packing tool
place	packingtool:place(AMessage: String)	Calculate the packed placement of all meshes
placepart	packingtool:placepart(AMesh: TLUAMeshObject)	Calculate the placement for a single mesh
applyplacement	packingtool:applyplacement(AMesh: TLUAMeshObject)	Apply the calculated placement to a mesh object
sortbypartheight	packingtool:sortbypartheight()	Set the sorting to "SortByPartHeight"
sortbysize	packingtool:sortbysize()	Set the sorting to "SortBySize"
addrectanglebin	packingtool:addrectanblebin(x, y, offsetX, offsetY: Number)	Add a subdivision for the placement

2.16 Reference: CadImporter Object

2.16.1 Properties

None.

2.16.2 Method Overview

Name	Syntax	Description
loadmodel	cadmodel = importer:loadmodel('handle.step', 0.1, 20, 5)	Load the model "handel.step" with a maximum surface deviation of 0.1 and a angle tolerance and a maximal edge length of 5 mm
Loadmodel	Cadmodel = importer:loadmodel('handle.step', 5)	Load the model "handle.step" with the techsoft Detail level Very High Detail. Detail Level reaches from 1 – Very Low to 5 – Very High

2.17 Reference: CADImportModel Object

Sometimes a CAD File consists of several single files. You can access the number of single entities by the property *entitycount* and get an entity's name by the method *getentityname(Index).* You can create a mesh with all entities using the method *createmesh* or you can create a mesh with a single entity using the property *createsinglemesh.*

2.17.1 Properties

Name	Syntax	Description
trianglecount	triangles = cadmodel.trianglecount	Holds the number of triangles of the cadmodel
entitycount	entities = cadmodel.entitycount	Holds the number of entities of the cadmodel

2.17.2 Method Overview

Name	Syntax	Description
createmesh	mesh = cadmodel:createmesh()	Creates a mesh with all entities
createsinglemesh	mesh = cadmodel:createsinglemesh (EntityIndex: Number);	Creates a mesh with a single entity

getentityname	name = cadmodel:getentityname(EntityIndex: Number);	Returns the entity's name

2.18 Reference: webglExport Object

2.18.1 Properties

None.

2.18.2 Method Overview

Name	Syntax	Description
setExportFlags	webglexport:setExportFlags(normal: Boolean, colours: Boolean, texture coordinates: Boolean, secondary colours: Boolean)	Enables or disables the export of normals, colours, texture coordinates and secondary colours
exportMesh	webglexport:exportMesh(mesh: mesh, filename: string);	Exports a binary dump of the modle, which is readable by the Netfabb webgl stub.

2.19 Reference: Fabbproject API

The fabbproject API allows to create, load, modify and save fabbproject files used by Netfabb.

2.19.1 Fabbproject class

This is the central class of the fabbproject API. It holds references to the trays within the project and allows saving. The methods "System:loadfabbproject" and "System:newfabbproject" (Fabbproject) create instances of the class.

2.19.1.1 Method overview

Name	Syntax	Description

savetofile	fabbproject:savetofile(filename: String)	Saves a fabbproject to a file
addtray	fabbproject:addtray(name: String, machinesize_x: Number, machinesize_y: Number, machinesize_z: Number)	Adds a new tray to the fabbproject. “Name” is the name of the tray, machinesize_x, _y, and _z are the sizes of the tray.
Gettray	fabbproject:gettray(index: Integer)	Retrieve a tray from a fabbproject
totalheight	fabbproject:totalheight()	The sum of heights of all trays (only the height of the space that is occupied by parts is included)

2.19.1.2 Properties

Property	Read / Write	Type	Description
name	Read / write	String	Name of the fabbproject's filename. Is empty when a new fabbproject is created and filled in load/save operations
root	read	String	Options that were not saved. Will be updated in save operations
traycount	read	number	Number of trays within the fabbproject

2.19.2 Reference: LUATray

A fabbproject can contain more than one tray. The trays can be obtained with the method "LUAFabbproject:gettray", the number of trays with the member "LUAFabbproject.traycount". Each tray has a property "root" which is the main LUAMeshGroup of the tray

2.19.2.1 Method overview

Name	Syntax	Description
getuuid	Uuid = tray:getuuid()	Retrieve the UUID of the tray
createpacker	Packer = tray:createpacker(packer_id: Number);	Create an object of type “LUAPacker” using the implementation defined in “packer_id”. See constants “packingid_null”, “packingid_outbox”, “packingid_2d”, “packingid_3d”, “packingid_montecarlo” for the correct id
checkforcollisions	Result = checkforcollisions(ARasterSize: Number)	Checks the tray for collisions. Returns an instance of "**LUACollisionResult**" (Reference: LUACollisionResult)

2.19.2.2 Properties

Property	Read / Write	Type	Description
name	Read / write	String	Name of the tray
root	read	LUAMeshGroup	Get the root of the tray
machinesize_x	Read / write	number	X-Size of the machine
machinesize_y	Read / write	number	Y-Size of the machine
machinesize_z	Read / write	number	Z-Size of the machine

errormessage	Read	String	Error message for tray actions, at the moment only used by “Packing3d”
filling_degree	Read	number	Calculates the percentage of the tray that is filled with the parts currently inside the tray. **Note**: parts that are outside or partly outside the tray will not be included in the calculation **Note**: the filling degree is only calculated to the maximum Z-Value of the parts in the tray. The vertical size of the machine is not taken into consideration.
packingid_outbox	Read	number	Constant id for the outbox packer. Used for “LUATray:pack”
packingid_2d	Read	number	Constant id for the 2d packer. Used for “LUATray: createpacker ”
packingid_3d	Read	number	Constant id for the 3d packer. Used for “LUATray: createpacker ”
packingid_montecarlo	Read	number	Constant id for the monte carlo packer. Used for “LUATray: createpacker ”

2.19.3 Reference: LUAMeshGroup

LUAMeshGroups are used to organize the meshes within a tray. Each LUAMeshGroup can contain meshes and subgroups. The root group of a tray is stored in "LUATray:root"

2.19.3.1 Method overview

Name	Syntax	Description
getmesh	Mesh = tray:getmesh(Index: Integer)	Get a TLUATrayMesh from the group
addmesh	Tray:addmesh(Mesh: LUAMesh)	Add a LUAMesh to the group
removemesh	Tray:removemesh(Mesh: TLUATrayMesh)	Remove a mesh from the group
addsubgroup	Subgroup = tray:adsubgroup(Name: String)	Add a new subgroup to the group
getsubgroup	Subgroup = tray:addsubgroup(Index: Integer)	Get a subgroup by index
deletesubgroup	Tray:deletesubgroup(Subgroup: LUASubgroup)	Remove and delete a subgroup

2.19.3.2 Properties

Property	Read / Write	Type	Description
name	Read / write	String	Name of the meshgroup
meshcount	read	Number	Get the number of meshes within the group
parent	read	LUAMeshGroup	The parent meshgroup

2.19.4 Reference: LUATrayMesh

The LUATrayMesh object contains the data of the actual meshes of a fabbproject, i.e. an instance of LUAMesh (Reference: Mesh Objects) and its position, scale and rotation

2.19.4.1 Method overview

Name	Syntax	Description
translate	Mesh:translate(x, y, z: Number)	Translate a mesh
Scale	Mesh:scale(x, y, z: Number)	Scale a mesh
Rotate	Mesh: rotate(axis_x, axis_y, axis_z, angle: Number)	Rotate a mesh by a defined angle (angle) around a defined axis (axis_x, axis_y, axis_z)
getuuid	Uuid = mesh:getuuid()	Get the UUID of a mesh
setmatrix	Mesh:setmatrix(matrix: LUAMatrix4)	Set the matrix of the mesh
savetostl	Mesh:savetostl(AFileName: String)	Export the mesh including all transformations to stl
savetoasciistl	Mesh:savetoasciistl(AFileName: String)	Export the mesh including all transformations to ascii stl
savetogts	Mesh:savetogts(AFileName: String)	Export the mesh including all transformations to gts
savetoobj	Mesh:savetoobj(AFileName: String)	Export the mesh including all transformations to obj
savetoncm	Mesh:savetoncm(AFileName: String)	Export the mesh including all transformations to ncm
savetoamf	Mesh:savetoamf(AFileName: String)	Export the mesh including all transformations to amf
savetox3d	Mesh:savetox3d(AFileName: String)	Export the mesh including all transformations to x3d
savetoply	Mesh:savetoply(AFileName: String)	Export the mesh including all transformations to ply
saveto3ds	Mesh:saveto3ds(AFileName: String)	Export the mesh including all transformations to 3ds
saveto3mf	Mesh:saveto3mf(AFileName: String)	Export the mesh including all transformations to 3mf
savetozpr	Mesh:savetozpr(AFileName: String)	Export the mesh including all transformations to ZPR
savetovrml	Mesh:savetovrml(AFileName: String)	Export the mesh including all transformations to VRML
setpackingoption	Mesh:setpackingoption(AOption: String; Avalue: String)	Add a packing option to the mesh

getpackingoption	Option = Mesh:setpackingoption(AOption: String)	Get a packing option from the mesh **Note**: the following read/write options are available: - priority (number, 1-10) - restriction (‘locked’, ‘norestriction’) - rotate (‘arbitrary’, ‘forbidden’, ‘zaxis’) These options have to be set before the start of the packing. The following read-only option is available: - state ('packed', 'leftover', 'not_packable', ‘colliding', 'excluded', 'ignored', 'indefinite') It describe the state of the part after the packing process: ‘packed’ – the part is packed normally; ‘leftover’ – the part could not be packed because the number of the parts was too large for the given tray; 'not_packable' – the part is too large for the given tray; ‘colliding' – the part in its initial position was colliding with a tray wall or with another part, while it was ‘locked’ or the packer option ‘start_from_current_positions‘ was chosen. 'excluded' – the part was excluded from the packing process by the user; ‘ignored’ – the part was ignored by the packer (for instance, because its mesh was empty or invisible) ‘Indefinite’ – the packing process has not been started yet or was aborted.
shellasmesh	Newmesh = mesh:shellasmesh(shellnumber: Number)	Extract a shell as new mesh object. Same as “LUAMeshObject.shellasmesh” but takes the matrix in the fabbproject into account. See “LUATrayMesh.mesh.shellcount” for the number of meshes

2.19.4.2 Properties

Property	Read / Write	Type	Description
Name	Read / write	String	Name of the mesh
Mesh	read	LUAMesh	Get the mesh of the traymesh. Please note that this is only a copy of the LUAMesh in the Lua Automation module.
parent	Read / write	LUAMeshGroup	Get or Set the group the mesh belongs to

color	Read/Write	Number	Set or get the color, (only the color used to display the mesh, not the mesh color information)
volume	read	Number	Get the volume of the tray mesh (including tansformation, in mm^3)
Area	read	Number	Get the area of the tray mesh (including tansformation, in mm^2)
outboxvolume	read	Number	Get the outbox volume of the tray mesh (including tansformation, in mm^3)
outboxheight	read	Number	Get the outbox height of the tray mesh (including tansformation, in mm)
outboxbasearea	read	Number	Get the outbox base area of the tray mesh (including tansformation, in mm^2)
matrix	read	LUAMatrix4	Get the transformation matrix of the mesh
Uuid	read	String	UUID of the mesh

2.19.5 Reference: LUACollisionResult

An instance of this object is returned by the “checkforcollisions” function of the LUATray object (Reference: LUATray). It contains information about the collisions within a tray.

2.19.5.1 Method overview

Name	Syntax	Description
getcollisionmesh	Mesh = collisionresult:getcollisionmesh(0)	Returns a colliding mesh. **Note**: if a collision is detected the two collising meshes are inserted into the list one by one, i.e. Mesh at index 0 collides with mesh at index 1 and so on **Note**: if a single mesh is more often than once in the list means that the mesh collides with more than one other mesh

2.19.5.2 Properties

Property	Read / Write	Type	Description
meshcount	Read	Number	Number of colliding meshes in the list **Note**: both colliding meshes are inserted into the list, so a meshcount of "2" means there is 1 collision

2.19.6 Reference: LUAPacker

An instance of this object is returned by the “createpacker” function of the LUATray object (Reference: LUATray). It can be used to pack the content of the tray.

2.19.6.1 Method overview

Name	Syntax	Description
pack	Result = packer:pack()	Pack the tray with the desired algorithm. Returns a packer-specific error code (general: 0 = no error)
getparametername	Result = packer:getparemetername(Index: Number)	Returns the name of the parameter with the given index. See tray.parametercount for the number of parameters available.
getparametertype	Result = packer: getparametertype(Index: Number)	Returns the type of the parameter with the given index as string (“integer”,”float”, “boolean”). See tray.parametercount for the number of parameters available.
getparametertypeid	Result = packer: getparametertypeid(Index: Number)	Returns the name of the parameter with the given index as constant (see LUAPacker.optInteger, LUAPacker.optFloat, LUAPacker.optBoolean). See tray.parametercount for the number of parameters available.
getparametervalue	Result = packer: getparametervalue(Index: Number)	Returns the value of the parameter with the given index. Note that the type of the returned value differs depending on the type of the paramter. See tray.parametercount for the number of parameters available.

2.19.6.2 Properties

Property	Read / Write	Type	Description
borderspacingxy	Read/Write	Number	The desired minimal distance from the border of the tray
borderspacingz	Read/Write	Number	The desired minimal distance from the build platform of the tray

minimaldistance	Read/Write	Number	The desired minimal distance between the object in the tray
optInteger	Read	Number	Constant for parameter type “Integer”
optFloat	Read	Number	Constant for parameter type “Float”
optBoolean	Read	Number	Constant for parameter type “Boolean”
parametercount	Read	Number	The number of parameters of the packer. Use the methods “tray:getparameter…” to get details about any parameters.
[Additional Parameter Name]	Read/Write	Number	Read or write the packer-specific parameters. Please refer to the documentation of the packer implementations for details.

2.19.7 Reference: LUAPacker – Monte Carlo packer

An instance of this object is returned by the function of the LUATray: object packer = tray:createpacker(tray.packingid_montecarlo); (Reference: LUATray)

2.19.7.1 Method overview

Name	Syntax	Description

pack	Result = packer:pack()	Pack the tray with the desired algorithm. Returns an error code that has the following meaning: 0 the packing is done. No problem has been detected. 1 the packing is done. There was not enough space in the tray to fit all items. 2 the packing is done. Some items are too large and cannot be fitted in the tray. 3 all the items are too large. None can be fitted into the tray. 4 there were no items to pack 5 starting the packing from the current item positions was not possible. Possible solutions: a) increase the packing quality b) try smaller values of the parameters min_dist and/or border_spacing_xy; c) do not start from current positions (set start_from_current_positions to false or drop this parameter) 13 unspecified error. Please contact the support team.

2.19.7.2 Properties

This section describes packer-specific properties of the Monte Carlo packer.

Property	Read / Write	Type	Description
packing_quality	Read/Write	Number	Is an integer number ranging from -8 (corresponding to the lowest packing density) to +8 (corresponding to the highest packing density). The higher the packing quality the more time is needed for the packing process. The default value is packing_quality=0
z_limit	Read/Write	Number	z_limit is the height in mm of the platform space that is allowed to contain the parts (the distance from the platform floor to the upper point of the packable region). It is a nonnegative real number which should not exceed the platform height: z_limit<=MachineSizeZ. If the parameter value does not satisfy these conditions, the default value is used instead. The default value is the platform height MachineSizeZ

start_from_current_positions	Read/Write	Boolean	Determines if the packing shoud be started from the current positions of the items (true) or an initial placement procedure has to be run before the packing begins (false). The default value is false.

2.19.8 Reference: LUAPacker – 3d Scanline packer

An instance of this object is returned by the function of the LUATray: object packer = tray:createpacker(tray.packingid_3dpacker); (Reference: LUATray)

2.19.8.1 Method overview

Name	Syntax	Description
pack	Result = packer:pack()	Pack the tray with the desired algorithm. Returns an error code that has the following meaning: 0 the packing is done. No problem has been detected. All parts could be packed. 1 Error occurred, please check the indivual mesh pack results

2.19.8.2 Properties

This section describes packer-specific properties of the Scanline packer.

Property	Read / Write	Type	Description
anglecount	Read/Write	Number	Integer, which gives the number of rotation, e.g. 4 means that 0, 90, 180 and 270 Degrees are tried. Values between 0,7 are allowed.
rastersize	Read/Write	Number	Integer, gives the rastersize of the rastersize. Values between 1 and 5 are allowed.
coarsening	Read/Write	Number	Integer, increases the rastersize. Values between 1 and 9 are allowed.

interlockingprotection	Read/Write	Number	Boolean, enables the interlocking protection
placeoutside	Read/Write	Number	Boolean, if true parts are moved outside the tray, if there could not be moved.
minimizeoutbox	Read/Write	Number	Boolean, if true, the outboxes of the meshes will be minimized.
allowrotationaxis	Read/Write	Number	Boolean, if true the parts will be rotated around the z-axis.

2.19.9 Reference: LUAPacker – 2d packer

An instance of this object is returned by the function of the LUATray: object packer = tray:createpacker(tray.packingid_2d); (Reference: LUATray). Please note that this packer is only available in the Lua Automation Module of the Desktop version

.Method overview

Name	Syntax	Description
pack	Result = packer:pack()	Pack the tray with the desired algorithm

2.19.9.1 Properties

This section describes packer-specific properties of the 2d packer.

Property	Read / Write	Type	Description
rastersize	Read/Write	Number	Float number, ranges between 0.1 and 10 mm. Gives the voxelsize which is internally for the algorithm. Named “voxelsize” in the Desktop GUI for the 2dpacker.

anglecount	Read/Write	Number	Integer, which gives the number of rotation, e.g. 4 means that 0, 90, 180 and 270 Degrees are tried. Values between 0,7 are allowed. Named “Z-Rotation” in the Desktop GUI for the 2dpacker.
coarsening	Read/Write	Number	Integer, increases the rastersize. Values between 1 and 9 are allowed.
placeoutside	Read/Write	Boolean	Boolean, if true parts are moved outside the tray, if there could not be moved.
packeonlyselected	Read/Write	Boolean	Boolean, if true only parts are packed, which have selected in the Desktop application

2.19.10 Reference: LUAPacker – outbox packer

An instance of this object is returned by the function of the LUATray: object packer = tray:createpacker(tray. packingid_outbox); (Reference: LUATray).

2.19.10.1 Method overview

Name	Syntax	Description
pack	Result = packer:pack()	Pack the tray with the desired algorithm

2.19.10.2 Properties

This section describes packer-specific properties of the Monte Carlo packer.

Property	Read / Write	Type	Description
rastersize	Read/Write	Number	Float number, ranges between 0.1 and 10 mm. Gives the voxelsize which is internally for the algorithm. Named “voxelsize” in the Desktop GUI for the 2dpacker.

pack2D	Read/Write	Boolean	If true, only planar packing is used, i.e. the parts are not packed in z.

2.20 LUAVector3

The LUA representation of a vector with 3 components (x, y, z)

2.20.1 Properties

Property	Read / Write	Type	Description
x	Read / write	Number	X-Component of the vector
y	Read / write	Number	Y-Component of the vector
z	Read / write	Number	Z-Component of the vector

2.21 LUAArray

Property	Read / Write	Type	Description
length	Read	Integer	Length of the array

Name	Syntax	Description
get	array:get(index: integer)	Get the array value at index
set	array:set(index: integer; value: number)	Set the array value at index

2.22 LUAStringMap

The LUA representation of a map of strings (key / value pairs)

Property	Read / Write	Type	Description
count	Read	Integer	Number of items in the map

Name	Syntax	Description
setitem	Map:setitem(AKey: String; Avalue: String)	Set an Item (add or replace)
getitem	Item = Map:getitem(AKey: String)	Get an Item by key
deleteitem	Map:deleteitem(AKey: String)	Delete an item with a key

getitembyindex	KeyItem = Map:getitembyindex(AIndex: Integer)	Get an item with key by index (Format: “Key”=”Value”)

2.23 LUAMatrix4

The Lua representation of a 4x4 Matrix.

Name	Syntax	Description
get	matrix:get(x: integer; y: integer)	Get the matrix value at x and y
set	matrix:set(x: integer; y: integer; value: number)	Set the matrix value at x and y

2.24 LUAAlignment

2.24.1 Properties

Property	Read / Write	Type	Description
newmodeltoworldmatrix	Read	TLUAMatrix4f	A new model to world matrix for the mesh/model to be transformed
transformationmatrix	Read	TLUAMatrix4f	The matrix that needs to be multiplied with the model to world matrix to align the mesh/model to be aligned with the reference mesh/model
firstaxis	Read	TLUAVector3f	First principal axis designating the shortest axis of the reference mesh. Can be used to set up the camera for rendering
secondaxis	Read	TLUAVector3f	Second principal axis designating the longest axis of the reference mesh. Can be used to set up the camera for rendering
thirdaxis	Read	TLUAVector3f	Third principal axis designating the remaining axis of the reference mesh. Can be used to set up the camera for rendering

2.25 LUAStamper

This Lua objects allows to label meshes. A text is stamped on the meshes. The position of the text, the estimated plane of the text, and the direction where is up needs to be given.

2.25.1 Properties

Property	Read / Write	Type	Description
depth	Read /write	number	The depth of the label.
height	Read /write	number	The height of the label in mm. The width is scaled to match the original ratio.
isinverted	Read /write	boolean	If true the text is inverted

issubtracted	Read /write	boolean	If false, the label is added to meshes, otherwise is subtracted
pos	Read	TLUAVector3f	The position of the label.
normal	Read	TLUAVector3f	The normal of the plane of the label.
upvector	Read	TLUAVector3f	The defines, what is up and down.

2.25.2 Methods

Name	Syntax	Description
stamp	newmesh = stamper:stamp(Mesh: LUAMesh, text : string)	Labels the text on the mesh. Creates a new mesh as an output.
setpos	stamper:setpos(APos1: number, APos2: number, APos2: number)	Sets the positon
setnormal	stamper:setnormal(APos1: number, APos2: number, APos2: number)	Sets the normal
setupvector	stamper:setupvector(APos1: number, APos2: number, APos2: number)	Sets the upvector (defines what is up and what is down for text)

2.26 ZipObject

This object is used for creating/managing ZIP files.

2.26.1 Properties

None.

2.26.2 Method Overview

Name	Syntax	Description
addstring	Zipobject:addstring(stringtoadd:string, zipname:string);	Adds stringtoadd to zipobject as zipname
addfile	Zipobject:addfile(filetoadd:string, zipname:string);	Adds filetoadd to zipobject as zipname
addtext	Zipobject:addtext(texttoadd:Lua Textfile object, zipname:string);	Adds texttoadd (a Lua textfile object) to zipobject as zipname
addxml	Zipobject:addxml(xmltoadd:Lua XML object, zipname:string);	Adds xmltoadd (a Lua XML object) to zipobject as zipname

exportfile	Zipobject:exportfile(filename:string);	Exports the ZIPfile as filename. File cannot be changed afterwards by API

3 Slice Objects

There exists three objects regarding slices:

- Slice Layer
- Slice Image Exporter
- Slice Object

Slice object is the main object.

3.1 Slice Layer

Property	Read / Write	Type	Description
contourcount	read	number	returns the number of contours of a slice layer
hatchcount	read	number	returns the number of hatches of a slice layer
contourlength	read	number	returns the contourlength a slice layer
hatchlength	read	number	returns the hatchlength a slice layer
minx	read	number	minimum X of a slice layer
miny	read	number	minimum Y of a slice layer
maxx	read	number	maximum X of a slice layer
maxy	read	number	maximum Y of a slice layer

Method	Syntax	Description
sort	slice:sort(?:number, ?:number);	
extractcontour	slice:extractcontour(?:number);	
begincontour	slice:begincontour();	
endcontour	slice:endcontour();	
addpoint	slice:addpoint(?:number, ?:number);	
move	slice:move(?:number, ?:number);	
getpointcount	slice:getpointcount();	returns the amount of points of a slice layer
getpointx	slice:getpointx(?:number, ?:number);	
getpointy	slice:getpointy(?:number, ?:number);	

3.2 Slice Image Exporter

Property	Read / Write	Type	Description
basefilename	read / write	string	base file name of the images
dpix	read / write	number	dpi in X
dpiy	read / write	number	dpi in Y

width	read / write	number	width of image in pixels
height	read / write	number	height of image in pixels
left	read / write	number	left space of the image in pixels
top	read / write	number	top space of the image in pixels
antialiased	read / write	boolean	Shall the image be antialiased?
writehatches	read / write	boolean	Shall the hatches be written?
writeclosedcontours	read / write	boolean	Shall the closed contours be written?
writeopencontours	read / write	boolean	Shall the open contours be written?
fillclosedcontours	read / write	boolean	Shall the closed contours be filled?
exportmonochromeimages	read / write	boolean	Shall monochrome images be exported?
dofillingbooled	read / write	boolean	
hatchcolor	read / write	number	color of the hatches
closedcontourcolor	read / write	number	color of the closed contours
opencontourcolor	read / write	number	color of the open colours
hatchwidth	read / write	number	width of the hatch in mm
closedcontourwidth	read / write	number	width of the closed contours in mm
opencontourwidth	read / write	number	width of the open contours in mm
fillcolor	read / write	number	color of the filling
backgroundcolor	read / write	number	color of the background
minx	read	number	minimum X of the image
miny	read	number	minimum Y of the image
minz	read	number	minimum Z of the image
maxx	read	number	maximum X of the image
maxy	read	number	maximum Y of the image
maxz	read	number	maximum Z of the image

Method	Syntax	Description
exportpng	slice:exportpng(identifier:string);	export slices as png
exportpngmulticore	slice:exportpngmulticore(identifier:string);	export slices as png with multicore support for faster calculation
exportbmp	slice:exportbmp(identifier:string);	export slices as bmp

3.3 Slice Object

Property	Read / Write	Type	Description
layercount	read	number	returns count of layers
layersize	read	number	returns thickness of the layers (if consistent throughout the slice)
name	read / write	number	returns / sets name of the slice
minx	read	number	minimum X of the slice in mm
miny	read	number	minimum Y of the slice in mm
minz	read	number	minimum Z of the slice in mm
maxx	read	number	maximum X of the slice in mm
maxy	read	number	maximum Y of the slice
maxz	read	number	maximum Z of the slice

Method	Syntax	Description
createoffset	slice:createoffset(offset[mm]: float, is_inner_offset:boolean, roundness [degree]:float = 30);	Creates an offset (inner or outer given by *is_inner_offset*) with a given roundness; *;* returns the result as new slice object
createsimplehatching	slice:createsimplehatching(hatchdistance:float , angle:float =0, angleincrement:float = 0, onlyeachlayer:int = 1, hatchoriginincrement:float = 0);	Creates a hatching on the sliceobject; you can increment the angle at each layer by setting *angleincrement;* hatches are created only at ech layer by setting *onlyeachlayer;* with the parameter *hatchoriginincrement* the hatchs can be shifted at each layer; returns the result as new slice object
smooth	slice:smooth(value:number);	Smoothes a slicer; returns the result as new slice object
reduce	slice:reduce(tolerance[mm]: float);	Reduces the points of a slice*;* returns the result as new slice object

moveslice	slice:moveslice(X[mm]:float, Y[mm]:float, Z[mm]:float);	Moves a slice by the given values;returns nothing
savetofile	slice:savetofile(identifier:string, slycetype:int, layersize:float, minz:float, maxZ:float,);	Exports a slice to a given slice format; *identifier* gives the name; *slicetype* gives the slice type (0-USF; 1- SLI; 2-CLI; 3- CLS; 4-SLC) ;returns nothing
removeselfintersections	slice:removeselfintersections();	Removes the self intersections; returns the result as new slice object
createupskin	slice:createupskin();	Creates an upskinr; returns the result as new slice object
createdownskin	slice:createdownskin();	Creates a downskinr;returns the result as new slice object
substractslice	slice:subtractslice(sliceobject:object);	Subtracts from slice the given slice *sliceobject.;* returns the result as new slice object
loadlayer	slice:loadlayer(index:int);	Returns a the slice layer at the given *index;* returns the result as new slice layer object
createaggregation	slice:createaggregation(Offset[mm]:float, Accuracy:float);	Create an aggregation; returns the result as new slice object
calculateslicearea	slice:calculateslicearea(sliceheight:float);	Calculates the area of a slice; slice is selected by its height; return the value
calculateslicecontour	slice:calculateslicecontour(sliceheight:float);	Calculate the length of a slice; slice is selected by its height; return the value
calculateslicearealayerindex	slice:calculateslicearea(sliceindex:int);	Calculates the area of a slice; slice is selected by its index; return the value
calculateslicecontourlayerindex	slice:calculateslicecontour(sliceindex:int);	Calculate the length of a slice; slice is selected by its index; return the value
createlayerunification	slice:createlayerunification(?:number);	Combine all slice files of the slice objects, starting with the heighest slice

getlayerz	slice:getlayerz(layerindex:int);	Returns the Z height of a layer
createimagerenderer	slice:createimagerenderer(layerindex:int, size:float);	Create an image of the slice
createrenderer	slice:createrenderer(layerfactor:int);	
executecalculation	slice:executecalculation(debugmessage:string , layerthickness:number);	Really executes all calculations
createstripefilling	slice:createstripfilling(HatchDistance:float, StripeWidth:float = 10, StripeGap:float, Angle:float = 0, AngleIncrement:float = 0, OnlyEachLayer:int = 1, SortType:int)	Create a strip filling
createquadfilling	slice:createquadfilling(HatchDistance:float, Angle :float = 0 , AngleIncrement :float =0, OnlyEachLayer : int = 1, HatchOriginIncrement :float = 0, QuadSizeX:float =20 , QuadSizeY:float =20)	Create a quad filling
duplicate	slice:duplicate	Duplicate the slice object;
removeselfintersectionmulticore	slice:removeselfintersections();	Removes the self intersections using multi cpu cores; returns the result as new slice object