uc5595 (view source)

Defined in: uf_std.h

Overview

uc5595 import icad binary report

Required License(s)

gateway

```
int uc5595
(
    const char * cp1,
    double * rp2,
    tag_t * nr3,
    char cr4 [ MAX_LINE_BUFSIZE ]
)
```

const char *	**cp1**	
double *	**rp2**	
tag_t *	**nr3**	
char	**cr4 [MAX_LINE_BUFSIZE]**	Output

---

uc5596 (view source)

Defined in: uf_std.h

Overview

uc5596 export NX objects to an icad binary report

Required License(s)

gateway

```
int uc5596
(
    const char * cp1,
    tag_t * np2,
    int ip3,
    char cr4 [ MAX_LINE_BUFSIZE ]
)
```

const char *	**cp1**	
tag_t *	**np2**	
int	**ip3**	
char	**cr4 [MAX_LINE_BUFSIZE]**	Output

UF_STD_ask_stl_file_type (view source)

Defined in: uf_std.h

Overview

This routine enquires the type of STL file for the specified filename.
The type of files are

UF_STD_STL_FILE_TYPE_ASCII -- Ascii STL file
UF_STD_STL_FILE_TYPE_BINARY -- Binary STL file
UF_STD_STL_FILE_TYPE_EXT_BINARY -- Extended binary STL file
UF_STD_STL_FILE_TYPE_UNSURE -- Either ascii or binary STL file
UF_STD_STL_FILE_TYPE_NONE -- Not an STL file

Note: Binary files which aren't STL binary files may be identified
incorrectly as extended binary STL files.

Environment

Internal and External

See Also

UF_STD_import_stl_binary_file
UF_STD_import_stl_ascii_file
UF_STD_set_default_stl_params

History

This function was originally released in V16.0.

Required License(s)

gateway

```
int UF_STD_ask_stl_file_type
(
    char * filename,
    UF_STD_stl_file_type_p_t file_type
)
```

char *	**filename**	Input	Filename of potential STL file
UF_STD_stl_file_type_p_t	**file_type**	Output	STL file type

UF_STD_close_stl_file (view source)

Defined in: uf_std.h

Overview

Closes an STL file.

Environment

Internal and External

Required License(s)

gateway

```
int UF_STD_close_stl_file
(
    void * file_handle
)
```

void *	**file_handle**	Input	Handle of file

UF_STD_create_activeweb_file (view source)

Defined in: uf_std.h

Overview

Create an ActiveWeb output file from geometry definitions in the part.

NOTES: This is a WNTI only function
Most of the input parameters can be left as empty strings and
appropriate values will be assigned from environment variables.
The following environment variables are used:

Variable Default for Field
UGII_ACTIVEWEB_GEOM_SERVER Node name for geometry server
UGII_ACTIVEWEB_ATTR_SERVER Node name for attribute server
UGII_ACTIVEWEB_SERVER_DIR Relative path for amm files
UGII_ACTIVEWEB_GEOM_DIR Path for geometry files
UGII_ACTIVEWEB_ATTR_DIR Path to attribute database directory
UGII_ACTIVEWEB_LOCAL_SERVER Local path to amm directory
UGII_ACTIVEWEB_LOCAL_GEOM Local path to geometry directory
UGII_ACTIVEWEB_LOCAL_WEB Local path to web server directory

EXAMPLE:

```
UF_STD_create_activeweb_file(
"c:\temp", "cypci204", "AmGeombase", "c:\\GeomFiles\\",
"phantom", "c:\\AttrDatabase\\",
"", "", "", 1.0,
UF_STD_ACTIVEWEB_CREATE_ATTR | UF_STD_ACTIVEWEB_ERASE_LOG );
```

This command would generate files in the temp directory.
The geometry server is "cypci204" and the local directory for amm files
is "AmGeombase". Attributes are on "phantom" under the directory
c:\\AttrDatabase. No attempt will be made to move files to output
directories so all of the generated files will be in the working
directory. The flags are set so that attributes are created and the
log files will be erased.

Environment

Internal and External

History

This function was originally released in V16.0.

Required License(s)

gateway

```
int UF_STD_create_activeweb_file
(
    char * working_directory,
    char * base_name,
    char * geom_server,
    char * server_directory,
    char * geometry_directory,
    char * attribute_server,
    char * attribute_directory,
    char * local_server_directory,
    char * local_geom_directory,
    char * local_web_directory,
    double tolerance,
    int mode_flags
)
```

char *	**working_directory**	Input	The working directory to be used for data files.
char *	**base_name**	Input	The default root name for all the generated files.
char *	**geom_server**	Input	The node name or IP address of the geometry server.
char *	**server_directory**	Input	The relative directory path on the geometry server for the .amm files.
char *	**geometry_directory**	Input	The relative or absolute path on the geometry server to the 3d data files (.3di or .3dd).
char *	**attribute_server**	Input	The name or IP address for the attribute/ annotation database server.
char *	**attribute_directory**	Input	The absolute path on the attribute serve to the .mdb database files.
char *	**local_server_directory**	Input	The absolute directory path for the location of the .amm files generated by the translator.
char *	**local_geom_directory**	Input	The absolute directory path for the location of the 3di and 3dd geometry data files.
char *	**local_web_directory**	Input	The absolute directory path for the location of the am3 web files. This directory will probably be the same location that other HTML, TIF or VRML files are being written.
double	**tolerance**	Input	Adjustment to facet tolerance. Values larger than 1.0 produce fewer output facets. Values less than 1.0 increase the number of facets generated.
int	**mode_flags**	Input	Bit field to control output contents. Flags are: UF_STD_ACTIVEWEB_CREATE_ATTR Controls creation of attribute database. UF_STD_ACTIVEWEB_ERASE_LOG Erases log files after successful optimization. UF_STD_ACTIVEWEB_MOVE_FILES Moves generated files to the output directories as specified by the local_server_directory, the local_geom_directory, and the

			local_web_directory. If the move flag is set and the directory spec for one of the local directories is blank, then that move will be ignored. UF_STD_ACTIVEWEB_RECURSIVE Apply translation to all components of an assembly. For component parts the base_name will be the part name. UF_STD_ACTIVEWEB_NO_OPTIMIZE Generates the polygon data (.3dx) file but does not invoke the optimizer to create any of the other files. This is useful if you desire to manually perform the optimization at another time.

UF_STD_create_vrml_file (view source)

Defined in: uf_std.h

Overview

Create a Virtual Reality Modeling Language (VRML) output file from geometry definitions in the part file. If no mode flags are specified a file is produced that contains only geometry and matches the VRML 1.0 specification. You can OR mode_flags to obtain different outputs.

This function can add Photo materials and textures to the export file. Any lights or backgrounds set up by NX in the current view can also be included in the VRML file. The file is in ASCII 3D data format for geometry that is used by the World Wide Web.

To output a VRML 2.0 spec file with light sources but no materials, textures, or backgrounds you could call this function as follows:
UF_STD_create_vrml_file("test.wrl", 1.0,
UF_STD_OUTPUT_LIGHTS | UF_STD_OUTPUT_VRML_2);

Environment

Internal and External

Required License(s)

gateway

```
int UF_STD_create_vrml_file
(
    char * file_name,
    double tolerance,
    int mode_flags
)
```

char *	**file_name**	Input	Name for the output VRML file.
double	**tolerance**	Input	adjustment to facet tolerance. Values larger than 1.0 produce fewer output facets. Values less than 1.0 increase the number of facets generated.
int	**mode_flags**	Input	Control bit fields for controlling output facets. The following are valid flags: UF_STD_OUTPUT_LIGHTS - output light sources when set.

UF_STD_OUTPUT_MATERIALS - output material properties if set.
UF_STD_OUTPUT_TEXTURES - output image textures if set.
UF_STD_OUTPUT_BACKGROUND - output background/foreground information
UF_STD_OUTPUT_VRML_2 - use VRML 2.0 specification if set.

UF_STD_export_icad_geometry (view source)

Defined in: uf_std.h

Overview
Exports geometry to an ICAD transfer file.

Environment
Internal and External

Required License(s)
gateway

```
int UF_STD_export_icad_geometry
(
    const char * file_spec,
    tag_t * objects,
    int count
)
```

const char *	**file_spec**	Input	File specification of binary transfer file to export
tag_t *	**objects**	Input	Array of object tags to export
int	**count**	Input	Number of objects

UF_STD_import_icad_geometry (view source)

Defined in: uf_std.h

Overview
import an icad transfer file

Environment
Internal and External

Required License(s)
gateway

```
int UF_STD_import_icad_geometry
(
    const char * file_spec,
    double matrix [ 12 ] ,
    tag_t * ug_tag
```

)

const char *	**file_spec**	Input	File specification of binary transfer file to import
double	**matrix [12]**	Input	Orientation matrix of reference set: [0-8] Reference Set Orientation Matrix [9-11] Reference Set Origin
tag_t *	**ug_tag**	Output	tag of reference set

UF_STD_import_stl_ascii_file (view source)

Defined in: uf_std.h

Overview

This routine imports the specified ascii STL file with the given STL parameters. The ascii STL file can be in gzipped format.

Environment

Internal and External

See Also

UF_STD_import_stl_binary_file
UF_STD_ask_stl_file_type
UF_STD_set_default_stl_params

History

This function was originally released in V16.0.

Required License(s)

gateway

```
int UF_STD_import_stl_ascii_file
(
    char * filename,
    UF_STD_stl_params_p_t parameters,
    long * parser_line,
    int * num_topologies,
    tag_t * * topologies,
    int * * facets_per_topol
)
```

char *	**filename**	Input	Name of ascii STL file
UF_STD_stl_params_p_t	**parameters**	Input	STL input parameters
long *	**parser_line**	Output	Last line reached when parsing file
int *	**num_topologies**	Output	Number of topologies created
tag_t * *	**topologies**	Output to UF_*free*	Array of topology tags. This array must be freed by calling UF_free.

int * *	**facets_per_topol**	Output to UF_*free*	Array of number of facets per topology. This array must be freed by calling UF_free.

UF_STD_import_stl_binary_file (view source)

Defined in: uf_std.h

Overview
This routine imports the specified binary STL file with the given STL parameters. The binary STL file cannot be in gzipped format, but can be an extended STL file.

Environment
Internal and External

See Also
UF_STD_import_stl_ascii_file
UF_STD_ask_stl_file_type
UF_STD_set_default_stl_params

History
This function was originally released in V16.0.

Required License(s)
gateway

```
int UF_STD_import_stl_binary_file
(
    char * filename,
    UF_STD_stl_params_p_t parameters,
    int * num_facets,
    tag_t * topology
)
```

char *	**filename**	Input	Name of binary STL file
UF_STD_stl_params_p_t	**parameters**	Input	STL import parameters
int *	**num_facets**	Output	Number of facets specified in the STL file
tag_t *	**topology**	Output	Topology tag of new facet model

UF_STD_import_vrml_file (view source)

Defined in: uf_std.h

Overview
Imports a VRML file into the work part according to the specified parameters.

Environment
Internal and External

See Also
UF_STD_set_default_vrml_params.
UF_STD_vrml_params_p_t
Refer to the example

History
This function was originally released in V15.0.

Required License(s)
gateway

```
int UF_STD_import_vrml_file
(
    char * filename,
    UF_STD_vrml_params_p_t parameters,
    int * num_errors,
    int * num_warnings,
    int * n_topologies,
    tag_p_t * topologies
)
```

char *	**filename**	Input	Name of file to import
UF_STD_vrml_params_p_t	**parameters**	Input	a Import parameters
int *	**num_errors**	Output	Number of errors found whilst parsing
int *	**num_warnings**	Output	Number of warnings found whilst parsing
int *	**n_topologies**	Output	Number of new facet topologies created
tag_p_t *	**topologies**	Output to UF_*free*	Array of tags of the created facet topologies. Must be freed by the caller.

UF_STD_open_binary_stl_file (view source)

Defined in: uf_std.h

Overview
Opens an STL file for binary output and return its handle.

Environment
Internal and External

See Also
UF_STD_close_stl_file

Required License(s)
(moldflow_ppa or sla_3d_systems)

```
int UF_STD_open_binary_stl_file
(
    char * file_name,
    logical append,
    char * header,
    void * * file_handle
)
```

char *	**file_name**	Input	Name of STL file to be written

logical	**append**	Input	Append flag:
			TRUE = Append
			FALSE = No append

char *	**header**	Input	Header text for start of binary file
void * *	**file_handle**	Output	Handle of file

UF_STD_open_text_stl_file (view source)

Defined in: uf_std.h

Overview
Opens an STL file for text output and return its handle.

Environment
Internal and External

See Also
UF_STD_close_stl_file

Required License(s)
gateway

```
int UF_STD_open_text_stl_file
(
    char * file_name,
    logical append,
    void * * file_handle
)
```

char *	**file_name**	Input	Name of STL file to be written

logical	**append**	Input	Append flag:
			TRUE = Append
			FALSE = No append

void * *	**file_handle**	Output	Handle of file

UF_STD_put_sheets_in_stl_file (view source)

Defined in: uf_std.h

Overview

Puts a facetized set of sheets in an STL file. The first sheet in the sheets array must have its normal in the correct direction to indicate the outside of the shell and it must have only one face.

Environment

Internal and External

Required License(s)

(moldflow_ppa or sla_3d_systems)

```
int UF_STD_put_sheets_in_stl_file
(
    void * file_handle,
    tag_t csys,
    int num_sheets,
    tag_t sheets [ ] ,
    double min_edge_len,
    double max_edge_len,
    double facet_toler,
    double adj_toler,
    int * num_negated,
    tag_p_t * negated,
    int * num_errors,
    UF_STD_stl_error_p_t * error_info
)
```

void *	**file_handle**	Input	Handle of file
tag_t	**csys**	Input	Specifies the coordinate system with respect to which output data is to be made. If specified as NULL_TAG, the current WCS is used.
int	**num_sheets**	Input	Number of sheets in shell
tag_t	**sheets [ ]**	Input	Solid bodies to facetize for STL
double	**min_edge_len**	Input	This argument is not used, it for future enhancements (Minimum facet edge length).
double	**max_edge_len**	Input	Maximum facet edge length
double	**facet_toler**	Input	Facet to surface tolerance
double	**adj_toler**	Input	Adjacency tolerance
int *	**num_negated**	Output	Number of sheets with inverted normals
tag_p_t *	**negated**	Output to UF_*free*	Tags of sheets with inverted normals

int *	**num_errors**	Output	Number of errors
UF_STD_stl_error_p_t *	**error_info**	Output to UF_*free*	Error information - NULL if there is none. This must be free by calling UF_free.

UF_STD_put_solid_in_stl_file (view source)

Defined in: uf_std.h

Overview
Puts a facetized solid in an STL file.

Environment
Internal and External

Required License(s)
(moldflow_ppa or sla_3d_systems)

```
int UF_STD_put_solid_in_stl_file
(
    void * file_handle,
    tag_t csys,
    tag_t body,
    double min_edge_len,
    double max_edge_len,
    double facet_toler,
    int * num_errors,
    UF_STD_stl_error_p_t * error_info
)
```

void *	**file_handle**	Input	Handle of file
tag_t	**csys**	Input	This argument is no longer used. Set to NULL_TAG.
tag_t	**body**	Input	Solid body to facetize for STL
double	**min_edge_len**	Input	This argument is not used, it for future enhancements (Minimum facet edge length).
double	**max_edge_len**	Input	Maximum facet edge length
double	**facet_toler**	Input	Facet to surface tolerance (Interactive Triangle Tolerance)
int *	**num_errors**	Output	Number of errors
UF_STD_stl_error_p_t *	**error_info**	Output to UF_*free*	Error information - NULL if there is none. This must be free by calling UF_free.

UF_STD_set_default_stl_params (view source)

Defined in: uf_std.h

Overview
This routine sets the given parameters to default.

Environment
Internal and External

See Also
UF_STD_import_stl_ascii_file
UF_STD_import_stl_binary_file
UF_STD_ask_stl_file_type

History
This function was originally released in V16.0.

Required License(s)
gateway

```
int UF_STD_set_default_stl_params
(
    UF_STD_stl_params_p_t params
)
```

UF_STD_stl_params_p_t	**params**	Output	Default STL parameters

UF_STD_set_default_vrml_params (view source)

Defined in: uf_std.h

Overview
Initializes the params data structure with the default VRML import parameter values.

Environment
Internal and External

See Also
UF_STD_import_vrml_file
UF_STD_vrml_params_p_t
Refer to the example

History
This function was originally released in V15.0.

Required License(s)
gateway

```
int UF_STD_set_default_vrml_params
(
    UF_STD_vrml_params_p_t params
)
```

UF_STD_vrml_params_p_t	**params**	Output	Default VRML import parameters