Enhanced Coating DLL

API Specification

Lambda Research Corporation

Introduction

This API specification identifies all function calls and parameters passed between the main executable, TracePro, written by Lambda Research Corporation, and the Enhanced Coating DLL (herein referred to as "DLL"), written by a licensee of TracePro. This functionality requires the Expert edition of TracePro. All references to "TracePro" within this document imply TracePro Expert.

The header files constitute a key portion of this document. The header files are shipped with TracePro in electronic form. In the event that a shipped header file differs from the description shown in this document, the shipped header file takes precedence since it was actually used in the compilation of the software. Please notify Lambda of any discrepancies and this document will be updated.

Document Layout

This specification identifies and describes function calls made by TracePro into the DLL. A syntax sheet is provided for each function, identifying and explaining the following:

- the frequency by which the function is called by TracePro
- the parameters passed by TracePro to each function
- the parameters returned by each function to TracePro

Calling Frequencies

The following calling frequencies apply:

- ***Per Simulation*** – Functions with this frequency are called once per simulation. A simulation consists of two phases, an audit phase and a raytrace phase.
- ***Per Wavelength*** – Functions with this frequency are called once per wavelength. This occurs in a portion of the audit phase. Once the raytrace phase begins, the raytrace proceeds one wavelength at a time. As the simulation switches from one wavelength to another, this portion of the audit phase is executed to update the model for the next wavelength to be traced.
- ***Per Surface*** – Functions with this frequency are called once for each surface in the TracePro model. These functions are called in the audit phase of the simulation.
- ***Per Surface Intersection*** -- Functions with this frequency are called each time a ray intersects a surface with a Coating DLL type surface property on it. These functions are called in the raytrace phase of the simulation.

Return Codes, Signals, and Constants -- TraceProDLL.h

All return codes, signals, and constants are defined in the header file, TraceProDLL.h. This header file is included when TracePro is compiled and built. DLL writers should use these definitions and not hard code numbers when their usage is warranted.

Description of Return Codes

The return codes for functions in the DLL are broken into two categories. One set for functions returning a double value; and another set for functions returning an integer value. The two sets mirror each other.

```
// for functions returning a double value
// (mostly for backward compatibility)

// no message to be output by TracePro
const double COATING_DLL_RETURN_CODE_OK = (double)0;

// OK and TracePro to output message
const double COATING_DLL_RETURN_CODE_OK_WITH_MESSAGE = (double)1;

// TracePro to output message but keep processing
const double COATING_DLL_RETURN_CODE_ERROR_GENERAL = (double)2;

// TracePro to output message and stop processing
const double COATING_DLL_RETURN_CODE_ERROR_CRITICAL = (double)3;


// for functions returning an integer value
// integer return codes (IRCs)

// no message to be output by TracePro
const int COATING_DLL_IRC_OK = 0;

// OK and TracePro to output message
const int COATING_DLL_IRC_OK_WITH_MESSAGE = 1;

// TracePro to output message but keep processing
const int COATING_DLL_IRC_ERROR_GENERAL = 2;

// TracePro to output message and stop processing
const int COATING_DLL_IRC_ERROR_CRITICAL = 3;
```

Function: fnInitDll
Calling Frequency: ***Per Surface***

Function Prototype:

```
// Function address for TracePro random number generator
typedef double (*RAND_FN)(void);
RAND_FN random_function;

double fnInitDll(
        RAND_FN         address,
        Long            nUniqueSurfaceID = 0,    // default for backward compatibility
        LPTSTR          szMessage = ""           // default for backward compatibility
        );
```

Description:

This is an initialization function called by TracePro to setup the random number generator for use by the DLL. This function is also the first announcement of the unique surface identifier. The writer of the DLL should use this function to store this unique surface ID in a variable for use throughout the DLL.

Parameters:

nUniqueSurfaceID:

unique surface ID established during the audit phase of the simulation

Data flow direction: TracePro → DLL

szMessage:

message sent to TracePro's message window if the return code is non-zero

Data flow direction: DLL → TracePro

Function: fnEvaluateCoating
Calling Frequency: ***Per Surface Intersection***

Function Prototype:

```
double fnEvaluateCoating(
        double*         input,
        int             input_size,
        double*         user,
        int             user_size,
        double*         results,
        int             results_size,
        long            nUniqueSurfaceID = 0,    // default for backward compatibility
        LPTSTR          szMessage = ""           // default for backward compatibility
        );
```

Description and Parameters:
NOTE: nUniqueSurfaceID and szMessage have the same meaning as described in the other functions.

The communication between TracePro and the DLL in the evaluation function, fnEvaluateCoating**,** is primarily through the use of 2 arrays, called the "input" array and the "results" array.

"input" array description
Data flow direction: TracePro → DLL

Description	**Variable**
wavelength	input[WAVELENGTH]
temperature	input[TEMPERATURE]
position X	input[RAY_POS_X]
position Y	input[RAY_POS_Y]
position Z	input[RAY_POS_Z]
incident direction X	input[RAY_DIR_X]
incident direction Y	input[RAY_DIR_Y]
incident direction Z	input[RAY_DIR_Z]
flux S0, (Stokes Vector)	input[RAY_FLUX_S0]
flux S1, (Stokes Vector)	input[RAY_FLUX_S1]
flux S2, (Stokes Vector)	input[RAY_FLUX_S2]
flux S3, (Stokes Vector)	input[RAY_FLUX_S3]
flux SX, (Stokes Vector)	input[RAY_FLUX_SX]
flux SY, (Stokes Vector)	input[RAY_FLUX_SY]
flux SZ, (Stokes Vector)	input[RAY_FLUX_SZ]
surface normal X	input[SURF_NORM_X]
surface normal Y	input[SURF_NORM_Y]
surface normal Z	input[SURF_NORM_Z]
n reflection side	input[N_REFL_SIDE]
k reflection side	input[K_REFL_SIDE]
n transmission side	input[N_TRAN_SIDE]
k transmission side	input[K_TRAN_SIDE]

"results" array description
Data flow direction: DLL → TracePro

Depending on the value assigned to the results[SIGNAL_TO_TRACEPRO], the results array will be interpreted differently by TracePro. There are three signals that TracePro will recognize. They are:

-COATING_DLL_SIGNAL_NONE
-COATING_DLL_SIGNAL_USE_MUELLER_MATRIX
-COATING_DLL_SIGNAL_FULL_RAY_CONTROL

1.) By assigning as follows:

```
results[SIGNAL_TO_TRACEPRO] = COATING_DLL_SIGNAL_NONE;
```

TracePro will recognize the following variables in the results array:

Description	**Variable**
s – absorptance	results[ABSO_S]
p – absorptance	results[ABSO_P]
s – reflectance	results[REFL_S]
p – reflectance	results[REFL_P]
s – transmittance	results[TRAN_S]
p – transmittance	results[TRAN_P]
Phase reflection [deg]	results[PHAS_R]
Phase transmission [deg]	results[PHAS_T]

2.) By assigning as follows:

```
results[SIGNAL_TO_TRACEPRO] = COATING_DLL_SIGNAL_USE_MUELLER_MATRIX;
```

TracePro will also recognize the following variables in the results array (the above variables for case 1 are also used):

Description	**Variable**
16 Mueller Matrix components for reflection	results[REFL_MM_xy] x -> 0 to 3 ; y-> 0 to 3
16 Mueller Matrix components for transmission	results[TRAN_MM_xy] x -> 0 to 3 ; y-> 0 to 3

3.) By assigning as follows:

results[SIGNAL_TO_TRACEPRO] = COATING_DLL_SIGNAL_FULL_RAY_CONTROL;

TracePro will recognize the following variables in the results array:

Description	**Variable**
s – absorptance	results[ABSO_S]
p – absorptance	results[ABSO_P]
output direction X	results[RAYOUT_DIR_X]
output direction Y	results[RAYOUT_DIR_Y]
output direction Z	results[RAYOUT_DIR_Z]
output Stokes, S0	results[RAYOUT_FLUX_S0]
output Stokes, S1	results[RAYOUT_FLUX_S1]
output Stokes, S2	results[RAYOUT_FLUX_S2]
output Stokes, S3	results[RAYOUT_FLUX_S3]
output Stokes, SX	results[RAYOUT_FLUX_SX]
output Stokes, SY	results[RAYOUT_FLUX_SY]
output Stokes, SZ	results[RAYOUT_FLUX_SZ]

When TracePro is sent the signal, COATING_DLL_SIGNAL_FULL_RAY_CONTROL, the 10 quantities (for direction and flux) are used in conjunction with two existing array elements, ABSO_S and ABSO_P, to control the next ray to propagate from the surface.

Surface absorption will occur based on the values of the ABSO_S and ABSO_P array elements. This will yield a convenient way to terminate rays by setting these to unity.

Only one ray is generated from the surface. Energy conservation, if desired, is controlled by the writer of the DLL. The energy conservation equation is:

results[RAYOUT_FLUX_S0] ==
input[RAY_FLUX_S0] *
(1 – 0.5 * results[ABSO_S] – 0.5 * results[ABSO_P])

The writer of the DLL assumes the responsibility of supplying a valid Stokes vector. TracePro will not perform any error checking for this. A valid Stokes vector must satisfy these conditions:

results[RAYOUT_FLUX_S0] >= 0.0

and…

(results[RAYOUT_FLUX_S0])^2 >= (results[RAYOUT_FLUX_S1])^2 +
(results[RAYOUT_FLUX_S2])^2 +
(results[RAYOUT_FLUX_S3])^2

Function: fnAnnounceOMLPath
Calling Frequency: *Per Simulation*

Function Prototype:

```
int fnAnnounceOMLPath(
        LPCTSTR     szPath,
        Long        nUniqueSurfaceID,
        LPTSTR      szMessage
        );
```

Description:

This function announces the full filename, including the path, of the OML file. If the simulation is run before the file is saved, the empty string is sent to the DLL.

Parameters:

szPath:

full filename of OML file including path

Data flow direction: TracePro → DLL

nUniqueSurfaceID:

unique surface ID established during the audit phase of the simulation

Data flow direction: TracePro → DLL

szMessage:

message sent to TracePro's message window if the return code is non-zero

Data flow direction: DLL → TracePro

Function: fnAnnounceDataDirectory
Calling Frequency: *Per Simulation*

Function Prototype:

```
int fnAnnounceDataDirectory(
        LPCTSTR     szDataDir,
        Long        nUniqueSurfaceID,
        LPTSTR      szMessage
        );
```

Description:

This function announces the data directory to the DLL. This is entered through the View|Options menu selection. In the General frame, the entry is labeled as "Coating DLL Directory:".

Parameters:

szDataDir:

data directory specified to TracePro

Data flow direction: TracePro → DLL

nUniqueSurfaceID:

unique surface ID established during the audit phase of the simulation

Data flow direction: TracePro → DLL

szMessage:

message sent to TracePro's message window if the return code is non-zero

Data flow direction: DLL → TracePro

Function: fnAnnounceSurfaceInfo
Calling Frequency: *Per Surface*

Function Prototype:

```
int fnAnnounceSurfaceInfo (
        LPCTSTR     szObjectName,
        LPCTSTR     szSurfaceName,
        LPCTSTR     szSurfacePropertyCatalog,
        LPCTSTR     szSurfacePropertyName,
        LPCTSTR     szCustomizedSurfaceParameter,
        long        nUniqueSurfaceID,
        LPTSTR      szMessage
        );
```

Description:

This function announces various surface information to the DLL.

Parameters:

szObjectName:

name of this surface's object

Data flow direction: TracePro → DLL

szSurfaceName:

the surface name

Data flow direction: TracePro → DLL

szSurfacePropertyCatalog:

the catalog in which this surface's surface property resides

Data flow direction: TracePro → DLL

szSurfacePropertyName:

the name of the surface property applied to this surface

Data flow direction: TracePro → DLL

szCustomizedSurfaceParameter:

customized surface parameter entered from the Apply Properties dialog box when applying the surface property

Data flow direction: TracePro → DLL

nUniqueSurfaceID:

unique surface ID established during the audit phase of the simulation

Data flow direction: TracePro → DLL

szMessage:

message sent to TracePro's message window if the return code is non-zero

Data flow direction: DLL → TracePro

Function: fnAnnounceLocalBoundingBox
Calling Frequency: *Per Surface*

Function Prototype:

```
int fnAnnounceLocalBoundingBox(
        double*         LocalBoundingBox,
        long            nUniqueSurfaceID,
        LPTSTR          szMessage
        );
```

Description:

This function announces the three orthogonal dimensional extents of a bounding box in the local coordinate system specified by the user.

Parameters:

LocalBoundingBox:

LocalBoundingBox is an array of 6 double values which contains the X, Y, and Z-dimension limits of local bounding box. These limits are with respect to the coordinate system defined by the origin, normal vector, and up vector when applying the surface property to this surface. The six individual elements of the array are obtained as follows:

LocalBoundBox[BOX_XMIN],
LocalBoundBox[BOX_XMAX],
LocalBoundBox[BOX_YMIN],
LocalBoundBox[BOX_YMAX],
LocalBoundBox[BOX_ZMIN], and
LocalBoundBox[BOX_ZMAX]

Data flow direction: TracePro → DLL

nUniqueSurfaceID:

unique surface ID established during the audit phase of the simulation

Data flow direction: TracePro → DLL

szMessage:

message sent to TracePro's message window if the return code is non-zero

Data flow direction: DLL → TracePro

Function: fnAnnounceRaytraceStart
Calling Frequency: *Per Simulation*

Function Prototype:

```
int fnAnnounceRaytraceStart(
        long            nUniqueSurfaceID,
        LPTSTR          szMessage
        );
```

Description:

This function announces the start of the raytrace.

Parameters:

nUniqueSurfaceID:

unique surface ID established during the audit phase of the simulation

Data flow direction: TracePro → DLL

szMessage:

message sent to TracePro's message window if the return code is non-zero

Data flow direction: DLL → TracePro

Function: fnAnnounceWavelengthStart
Calling Frequency: *Per Wavelength*

Function Prototype:

```
int fnAnnounceWavelengthStart(
        double      wavelength,
        long        nUniqueSurfaceID,
        LPTSTR      szMessage
        );
```

Description:

This function announces the start of the raytrace for a particular wavelength.

Parameters:

wavelength:

the wavelength, in microns, that will be raytraced next

Data flow direction: TracePro → DLL

nUniqueSurfaceID:

unique surface ID established during the audit phase of the simulation

Data flow direction: TracePro → DLL

szMessage:

message sent to TracePro's message window if the return code is non-zero

Data flow direction: DLL → TracePro

Function: fnAnnounceWavelengthFinish
Calling Frequency: *Per Wavelength*

Function Prototype:

```
int fnAnnounceWavelengthFinish(
        double      wavelength,
        long        nUniqueSurfaceID,
        LPTSTR      szMessage
        );
```

Description:

This function announces the completion of the raytrace for a particular wavelength.

Parameters:

wavelength:

the wavelength, in microns, that was just raytraced

Data flow direction: TracePro → DLL

nUniqueSurfaceID:

unique surface ID established during the audit phase of the simulation

Data flow direction: TracePro → DLL

szMessage:

message sent to TracePro's message window if the return code is non-zero

Data flow direction: DLL → TracePro

Function: fnAnnounceRaytraceFinish
Calling Frequency: *Per Simulation*

Function Prototype:

```
int fnAnnounceRaytraceFinish(
        int             nStatus,
        long            nUniqueSurfaceID,
        LPTSTR          szMessage
        );
```

Description:

This function announces the completion of the raytrace.

Parameters:

nStatus:

the raytrace finishing status

one of the following const values will be sent by TracePro:

```
const int COATING_DLL_RAYTRACE_FINISH_OK = 0;
const int COATING_DLL_RAYTRACE_FINISH_USER_CANCEL = 1;
const int COATING_DLL_RAYTRACE_FINISH_CATCH_EXCEPTION = 2;
const int COATING_DLL_RAYTRACE_FINISH_CRITICAL_ERROR_FROM_DLL = 3;
```

Data flow direction: TracePro → DLL

nUniqueSurfaceID:

unique surface ID established during the audit phase of the simulation

Data flow direction: TracePro → DLL

szMessage:

message sent to TracePro's message window if the return code is non-zero

Data flow direction: DLL → TracePro