D15 • User DLL API

D15.1 • Overview

This chapter explains how to develop a user DLL for connection to the data browser. The DLL must comply with a specific API (Application Programming Interface). The API allows functions with an arbitrary number of input channels, as well as any number of input floating point arguments. In programmatic form, the API supports for functions of type:

```
NewY = Function(X, Y1, Y2, .. , Yn, a1, a2, .. , an);
```

where the Y's are the channel values, and the a's are the floating point arguments.

The browser interface supports two types of calls. You can select a channel in the browser, and call a user function with this channel as first argument, followed by possible floating-point values. The result may override the source channel (same type/mnemo), or be added as a new channel to the parent container. Alternatively, you can create in the Data Store a "User function object" corresponding to a function with any number of input channels, drop those channels on the object, and generate the result. The output mnemonic may be forced by the DLL of be user defined. Beware that this choice will govern unit conversions for the resulting channel.

D15.2 • Sample source

The User DLL toolkit includes this document as well as a sample VC++ 6 project. The project comprises the files: ExternalDll.h, ExternalDll.cpp, ExternalDll.dsp, and ExternalDll.def.

In the form delivered by Kappa, the ExternalDll project creates a DLL called SimpleDeviationCorrection. Two input channels are expected, a density, and a deviation. The DLL returns the density divided by the cosine of the deviation. The project can be used as a template for building anything else.

The next section reviews the API

D15.3 • DLL API

Below is the list of the functions in the DLL interface. Recall that the general form of the functions supported by the API is:

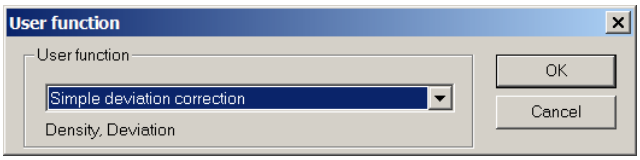
```
NewY = Function(X, Y1, Y2, .. , Yn, a1, a2, .. , an);
```

where the Y's are the channel values, and the a's are the floating point arguments.

D15.3.1 ● General definitions

```
DLL_EXPORT void WINAPI GetFunctionName(char* FunctionName);
// FunctionName: returned character string
```

This function is used by Emeraude to display the function name, e.g. below “Simple deviation correction”.




```
DLL_EXPORT void WINAPI SetAbsentValue (double dAbsentValue);
```

This function sets a global variable gAbsentValue in the DLL and will be called by Emeraude if it is present in the DLL. The advantage of this function is to give the ability for the DLL to return absent values in the Emeraude sense

D15.3.2 ● Input channel information

```
DLL_EXPORT int WINAPI GetNumberOfInputY ();
```

This function returns the number of required input channels. Recall that a DLL with only one input channels can be called directly with a right click on any browser channel + “Edit” - “User function”. A DLL with multiple input channels can be called after selecting the Data Store node, right click and then “User Function”.

```
DLL_EXPORT PROPERTY WINAPI GetInputYProperties(int nIndex);
// nIndex: index of the considered input channel; starts at 0
```

The API provides the ability to specify the property of the input channels so that Emeraude can make some type checking when you supply the input to a DLL. For instance, with a function of several inputs, you typically drop the input channels on a “user function object” in the browser, and as you do so, Emeraude checks that the types correspond to what the DLL requires.

Properties are defined by an enum, found in the file ExternalDll.h. If the property of a given input channel does not matter, you can use the property “ANY”.

```
enum PROPERTY{                                  //Internal unit
      ANY,
      TIME,         // Time                     hours
      DEPTH,        // Depth                    Meter
      PRES,         // Pressure                 Pascal
      ID,           // Internal diameter        Meter
      TREL,         // Relative temperature     Celsius
      TABS,         // Absolute temperature     Kelvin
      CSPEED,       // Cable speed              m/s
```

```
      TENS,        // Tension                    kg
      GRAY,        // Gamma ray                  GAPI
      SPIN,        // Spinner                    rps
      Q,           // Donwhole rate              m3/s
      QG,          // Surface gas rate           m3/s
      QL,          // Surface liquid rate        m3/s
      MU,          // Viscosity                  Pa.s
      DENS,        // Density                    kg/m3
      DEV,         // Deviation                  radians
      CONDUCTIVITY,//Conductivity                S/m
      NOUNIT,      // Without a unit (holdups, Other)
      CAPACITANCE,// Capacitance                 cps
      RUGOSITY,    //Roughness                   none
      K,           // Permeability               m2
      CAPTURE,     // Capture cross section      c.u.
      SALINITY,    // Salinity                   PPM
      HEAT_CAPACITY,     // Heat capacity        cal/(g.°C)
      HEAT_CONDUCTIVITY,//Heat conductivity     cal/(s.m.°C)
      MOBILITY,          // Mobility             m2/Pa.sec
};
```

!! Warning: Emeraude internally works in SI units and the channels are passed in this unit. The right column above indicates the internal units in which values are retrieved by the DLL.

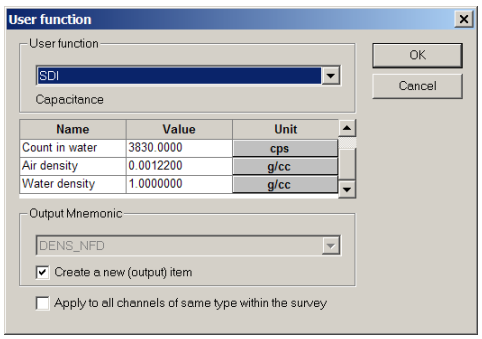
D15.3.3 ● Input parameters information

```
DLL_EXPORT int WINAPI GetNumberOfInputCste();
```

This function returns the number of required input parameters.

```
DLL_EXPORT void WINAPI GetInputCsteInfo(int nIndex, char* Name,   double *dMin,
double *dMax, double *dDefault, PROPERTY* Prop);
// nIndex: index of the considered input parameters; starts at 0
// Name: Name of the parameter
// dMin/dMax: minimum/maximum value for the parameter (inclusive)
// dDefault: default parameter value
// Prop: Property, as for the input channels
```

The above information is retrieved by Emeraude to display a grid for parameter input, implementing interval checking and unit conversion, see below.



!! Warning: recall that if you specify a property different from NOUNIT, dMin, dMax, and dDefault as well as the value retrieved by the DLL are in internal unit, as specified in the PROPERTY enum above.

D15.3.4 • Output

```
DLL_EXPORT BOOL WINAPI GetOutputMnemo(char* szMnemo, int iszMnemo);
//szMeno: the Mnemonic string
//iszMnemo: the number of character in the szMnemo string
// Return FALSE if you do not want to specify the mnemo, TRUE otherwise.
```

This function can be used to specify internally what the output mnemonic should be. This mnemonic must correspond to an internal Emeraude mnemo, or a user mnemo. If the output mnemonic is specified in the DLL it cannot be changed at the time of the call. When the mnemonic is not indicated, you can select the mnemonic from a list. The choice must be made carefully as by choosing a mnemonic you also chose a property, and recall that the output channel values will be treated as in the internal unit of the selected property.

```
DLL_EXPORT double Function(int iNbChannels, double X, double *Y, int iNbCste,
double *dCste);
// int nNbChannels: the number of input channels
// double X: the depth in meters
// double *Y: the array of Y values for the input vectors
// int iNbCste: the number of input parameters
// double dCtse: their value
```

The function above corresponds to the call actual DLL function, at least to its application at a particular depth. When there are several input channels, the depths of the output are the depths of the first input channel. All other channels are interpolated at those points.