Technical documentation for VantagePro Dll 2.42

October, 2009

Description of the essential files included with the VantageProDll.

readme.htm	This file, which documents the DLL constants, functions and structures, with examples of DLL usage.
VantagePro.dll	Dll for communicating with VantagePro station. It should be copied to your project or the windows directory.
SiUSBxp.dll, SiUSBxp.lib	VantagePro DLL uses this DLL internally for certain configurations. It should be copied to your project or the windows directory.
CP210xManufacturing.dll, CP210xManufacturing.lib	VantagePro DLL uses this DLL internally for certain configurations. It should be copied to your project or the windows directory.
VantagePro.lib	Import library for the Dll. When compiling in C/C++ environment, include this library in the link path and VantageProDll.h as the header file for your source files.
VantagePro.h	C/C++ header file containing the declarations for the Dll functions. This file is needed only for C/C++ programmers.
VantageProDll.vb	Visual Basic header file containing the declarations for the Dll functions. This file is needed only for Visual Basic programmers.
license.rtf	License agreement for the Dll usage.

Both the Debug version (under Debug sub-directory) and the Release version (in the default DLL directory or Release sub-directory) of VantagePro.dll and VantagePro.lib are included in this package. Our 3rd party Dlls (SiUSBxp.dll and CP210xManufacturing.dll) are included too. A copy of the Debug VantagePro.dll package is also in each of the two DLL Sample Applications.
A copy of the Device Driver Installers for direct USB and Virtual Comm USB are also included. Please manually install them from the sub-directory under "Drivers", if needed.

IMPORTANT NOTES:

1. To avoid ambiguity of function calls, VantagePro dll should not be loaded with any other Weather Dll's (which may contain the same function signatures). Also, be sure siusbxp.dll and CP210xManufacturing.dll are in the same directory as VantagePro.Dll. Or this error ambiguous message shall be displayed: Unable to load module "VantagePro.dll".

2. There are 2 functions for Setting communication Time-Out in DLL 2.40/2.41/2.42. The original SetCommTimeoutVal_V() is included for compatibility with application using previous DLL version. However, further tests show that this function might hang if the communication cable is disconnected. A new method of SetVantageTimeouVal_V(), which is widely used in our popular WeatherLink software, is added to DLL 2.40/2.41/2.42. It is the recommended function for replacing SetCommTimeoutVal_V(). Please refer to the SetVantageTimeoutVal_V() function section for its usage, and replace your existing SetCommTimeoutVal_V() with SetVantageTimeoutVal_V() if you encounter TimeOut error. Please contact Davis Instruments for support if you have TimeOut issue with this new SetVantageTimeoutVal_V( ) function.

Table of Contents

Note: The syntax used in the following constant, structure and function definitions is based on C/C++. Visual Basic support had just been added and that is documented in the Appendixes, along with Example programs for DLL usage.

DLL Functions

All of the function names in this DLL end with the suffix "_V" to help avoid name conflicts with other functions or variables in your program.

Functions that are new or modified in version 2.4 are in **Bold** type.

DLL Initialization Functions

GetDllVersion_V	OpenCommPort_V	OpenDefaultCommPort_V	
CloseCommPort_V	SetCommTimeoutVal_V	**GetUnits_V**	**SetUnits_V**
SetRainCollectorModel_V	GetRainCollectorModel_V	**OpenUSBPort_V**	**OpenTCPIPPort_V**
CloseUSBPort_V	**CloseTCPIPPort_V**	**SetVantageTimeoutVal_V**	**SetVantageTimeoutVal_V**
GetUSBDevSerialNumber_V			

Lowlevel Functions

GetSerialChar_V	PutSerialStr_V	PutSerialChar_V

Station Configuration Functions

InitStation_V			
GetModelNo_V	GetStationTime_V	SetStationTime_V	GetArchivePeriod_V
SetArchivePeriod_V	GetStationFirmwareDate_V	**GetStationFirmwareVersion_V**	GetReceptionData_V
GetVantageLat_V	SetVantageLat_V	GetVantageLon_V	SetVantageLon_V
SetVantageLamp_V	SetNewBaud_V	GetVantageTxConfig_V	SetVantageTxConfig_V
GetBarometerData_V	PutBarometer_V	PutTotalRain_V	
GetRainCollectorModelOnStation_V		GetAndSetRainCollectorModelOnStation_V	
SetRainCollectorModelOnStation_V			
GetWindCupSize_V	**SetWindCupSize_V**	GetTempAvg_V	SetTempAvg_V
GetTimeZoneSettings_V	SetTimeZoneSettings_V	**IsNewLoopPackageSupported_V**	

Current Data Functions

LoadCurrentVantageData_V			
GetBarometer_V	GetOutsideTemp_V	GetDewPt_V	GetWindChill_V
GetInsideTemp_V	GetInsideHumidity_V	GetOutsideHumidity_V	GetTotalRain_V
GetDailyRain_V	GetMonthlyRain_V	GetStormRain_V	
GetWindSpeed_V	GetWindDir_V	GetWindDirStr_V	GetRainRate_V
GetET_V	GetMonthlyET_V	GetYearlyET_V	GetSolarRad_V
GetUV_V	GetHeatIndex_V	GetActiveAlarms_V	**GetCurrentDataByID_V**
GetCurrentDataStrByID_V	GetStartOfCurrentStorm_V	GetSunriseTime_V	GetSunsetTime_V

Alarm Functions

LoadVantageAlarms_V	SetVantageAlarms_V	ClearVantageAlarms_V

GetBarRiseAlarm_V GetBarFallAlarm_V GetTimeAlarm_V GetTimeAlarmStr_V
GetInsideLowTempAlarm_V GetInsideHiTempAlarm_V GetOutsideLowTempAlarm_V GetOutsideHiTempAlarm_V
GetLowInsideHumAlarm_V GetHiInsideHumAlarm_V GetLowOutsideHumAlarm_V GetHiOutsideHumAlarm_V
GetLowWindChillAlarm_V GetLowDewPtAlarm_V GetHiDewPtAlarm_V GetHiSolarRadAlarm_V
GetHiWindSpeedAlarm_V GetHi10MinWindSpeedAlarm_V GetHiHeatIndexAlarm_V GetHiTHSWAlarm_V
GetHiRainRateAlarm_V GetHiDailyRainAlarm_V GetHiRainStormAlarm_V GetFlashFloodAlarm_V
GetHiUVAlarm_V GetHiUVMedAlarm_V
GetLowExtraTempAlarm_V GetHiExtraTempAlarm_V GetLowExtraHumAlarm_V GetHiExtraHumAlarm_V
GetLowSoilTempAlarm_V GetHiSoilTempAlarm_V GetLowSoilMoistureAlarm_V GetHiSoilMoistureAlarm_V
GetLowLeafTempAlarm_V GetHiLeafTempAlarm_V GetLowLeafWetAlarm_V GetHiLeafWetAlarm_V

PutBarRiseAlarm_V PutBarFallAlarm_V PutTimeAlarm_V
PutInsideLowTempAlarm_V PutInsideHiTempAlarm_V PutOutsideLowTempAlarm_V PutOutsideHiTempAlarm_V
PutLowInsideHumAlarm_V PutHiInsideHumAlarm_V PutLowOutsideHumAlarm_V PutHiOutsideHumAlarm_V
PutLowWindChillAlarm_V PutLowDewPtAlarm_V PutHiDewPtAlarm_V PutHiSolarRadAlarm_V
PutHiWindSpeedAlarm_V PutHi10MinWindSpeedAlarm_V PutHiHeatIndexAlarm_V PutHiTHSWAlarm_V
PutHiRainFloodAlarm_V PutRainPerDayAlarm_V PutRainStormAlarm_V PutRainRateAlarm_V
PutHiUVAlarm_V PutHiUVMedAlarm_V
PutLowExtraTempAlarm_V PutHiExtraTempAlarm_V PutLowExtraHumAlarm_V PutHiExtraHumAlarm_V
PutLowSoilTempAlarm_V PutHiSoilTempAlarm_V PutLowSoilMoistureAlarm_V PutHiSoilMoistureAlarm_V
PutLowLeafTempAlarm_V PutHiLeafTempAlarm_V PutLowLeafWetAlarm_V PutHiLeafWetAlarm_V

ClearBarRiseAlarm_V ClearBarFallAlarm_V ClearTimeAlarm_V
ClearInsideLowTempAlarm_V ClearInsideHiTempAlarm_V ClearOutsideLowTempAlarm_V ClearOutsideHiTempAlarm_V
ClearLowInsideHumAlarm_V ClearHiInsideHumAlarm_V ClearLowOutsideHumAlarm_V ClearHiOutsideHumAlarm_V
ClearLowWindChillAlarm_V ClearLowDewPtAlarm_V ClearHiDewPtAlarm_V ClearHiSolarRadAlarm_V
ClearHiWindSpeedAlarm_V ClearHi10MinWindSpeedAlarm_V ClearHiHeatIndexAlarm_V ClearHiTHSWAlarm_V
ClearHiRainFloodAlarm_V ClearHiRainPerDayAlarm_V ClearRainStormAlarm_V ClearRainRateAlarm_V
ClearHiUVAlarm_V ClearHiUVMedAlarm_V
ClearLowExtraTempAlarm_V ClearHiExtraTempAlarm_V ClearLowExtraHumAlarm_V ClearHiExtraHumAlarm_V
ClearLowSoilTempAlarm_V ClearHiSoilTempAlarm_V ClearLowSoilMoistureAlarm_V ClearHiSoilMoistureAlarm_V
ClearLowLeafTempAlarm_V ClearHiLeafTempAlarm_V ClearLowLeafWetAlarm_V ClearHiLeafWetAlarm_V

High Low Functions

LoadVantageHiLows_V
GetHiOutsideTemp_V GetLowOutsideTemp_V GetHiLowTimesOutTemp_V
GetHiInsideTemp_V GetLowInsideTemp_V GetHiLowTimesInTemp_V
GetHiOutsideHum_V GetLowOutsideHum_V GetHiLowTimesOutHum_V
GetHiInsideHum_V GetLowInsideHum_V GetHiLowTimesInHum_V
GetHiDewPt_V GetLowDewPt_V GetHiLowTimesDewPt_V
GetLowWindChill_V GetLowTimesWindChill_V
GetHiWindSpeed_V GetHiTimesWindSpeed_V
GetHiLowDataByID_V GetHiLowDataStrByID_V
GetHiLowTimeByID_V GetHiLowTimeStrByID_V

Calibrate Functions

LoadVantageCalibration_V PutOutsideTempCalibrationValue_V PutOutsideTempCalibrationValueEx_V
PutInsideTempCalibrationValue_V
PutOutsideHumCalibrationValue_V PutOutsideHumCalibrationValueEx_V PutInsideHumCalibrationValue_V PutOutsideTempCalibrationOffset_V
PutOutsideTempCalibrationOffsetEx_V
PutInsideTempCalibrationOffset_V PutOutsideHumCalibrationOffset_V PutOutsideHumCalibrationOffsetEx_V PutInsideHumCalibrationOffset_V
SetVantageCalibration_V GetWindDirCalibrationOffset_V PutWindDirCalibrationOffset_V
PutBarometer_V

Download Functions

Clear Functions

DLL Structures

All of the structures used in the 2.0 version of the DLL have the same definitions in this version. There is an additional structure, WeatherRecordStructEx, that is used to retrieve additional data values not available in the earlier DLL version.

Note: The application programmer is responsible for allocating space for any of the following structures and passing a pointer or reference to the structure to the DLL function that will use the data. They are also responsible for de-allocating the memory when the structure is no longer needed.

DateTime

Holds a date and time, but not the year..

```
struct  DateTime
{
   short int month;
   short int day;
   short int hour;
   short int min;
};
```

DateTimeStamp

Holds a date and time, including the year..

```
struct  DateTimeStamp
{
   int minute;
   int hour;
   int day;
   int month;
   int year;
};
```

WeatherUnits

Holds the unit selections for the DLL. Used by GetUnits_V and SetUnits_V.

```
struct  WeatherUnits
{
   char TempUnit;
   char RainUnit;
   char BaromUnit;
   char WindUnit;
   char elevUnit;
};
```

WeatherRecordStruct

Holds a data record from the VantagePro archive memory. The data is stored in the units currently selected in the DLL. The data values are filled in by GetArchiveRecord_V. This structure is primarily for backward compatibility with previous DLL versions.

```
struct  WeatherRecordStruct
{
   short year;
   char month;
   char day;
   short packedTime;
   char dateStr[16];
   char timeStr[16];
   float heatIndex;
   float windChill;
   float hiOutsideTemp;
   float lowOutsideTemp;
   float dewPoint;
   float windSpeed;
   short windDirection;
   char windDirectionStr[5];
   float hiWindSpeed;
   float rain;
   float Barometer;
   float insideTemp;
   float outsideTemp;
   float insideHum;
   float outsideHum;
   short archivePeriod;
   short solarRad;
   float uv;
   float et;
   short hiWindDirection;
   char hiWindDirectionStr[5];
};
```

CurrentVantageCalibration

Holds the current temperature and humidity readings and calibration offsets as read off of the VantagePro console. The data values are filled in by LoadVantageCalibration_V.

```
struct  CurrentVantageCalibration
{
   //console readings
   float tempIn;
   float tempOut;
   BYTE humIn;
   BYTE humOut;

   //calibration offsets
   float tempInOffset;
   float tempOutOffset;
   char humInOffset;
   char humOutOffset;
};
```

WeatherRecordStructEx

Holds a data record from the VantagePro archive memory. The data is stored in the units currently selected in the DLL. This structure contains more data fields than the WeatherRecordStruct used by previous versions of the DLL and is filled in by GetArchiveRecordEx_V.

```
struct  WeatherRecordStructEx
{
   short year;
   char month;
   char day;
   short packedTime;
   char dateStr[16];
   char timeStr[16];
   short archivePeriod;

   float outsideTemp;
```

```
    float hiOutsideTemp;
    float lowOutsideTemp;
    float insideTemp;

    float barometer;
    short barometerTrend;

    float outsideHum;
    float insideHum;

    float rain;
    float hiRainRate;

    float windSpeed;
    float hiWindSpeed;
    short windDirection;
    char windDirectionStr[5];
    short hiWindDirection;
    char hiWindDirectionStr[5];

    short numWindSamples;
    short numExpectedSamples;

    short solarRad;
    short hiSolarRad;
    float UV;
    float hiUV;

    float et;

    float extraTemp[3];
    float extraHum[2];
    float soilTemp[4];
    float leafTemp[2];

    float soilMoisture[4];
    float leafWetness[2];

    float heatIndex;
    float THWIndex;
    float THSWIndex;
    float windChill;
    float dewPoint;

    float insideDewPoint;
    float insideHeatIndex;
};
```

ReceptionStats

Holds the reception statistics for the ISS or wireless anemometer station on the VantagePro console since midnight or since they were cleared manualy on the console. The data is filled in by GetReceptionData_V.

```
struct  ReceptionStats
{
    long totalPacketsReceived;
    long totalPacketsMissed;
    long numberOfResynchs;
    long maxInARow;
    long numCRCerrors;
};
```

LatLonValue

Holds a latitude or longitude value. They can be expressed either as a floating point number that holds degrees and fractions or as integer degrees, minutes, and seconds. Both set of data fields are filled in when reading a value from the VantagePro. The "bUseFranctionalDegrees" field is used to select which set of data the DLL should use when writing the data to the weather station (0 = use the integer degrees/minutes/seconds fields; 1 = use the fractionalDegrees field).

Positive degree values are used for North latitude and East longitude. Negitive degrees are used for South latitude and West longitude. When using the

integer degrees/minutes/seconds fields, only the degrees value is negative. The minutes and seconds fields are always positive numbers between 0 and 60.

The data is filled in by GetVantageLat_V and GetVantageLon_V. The data structure is used to write new values to the VantagePro by SetVantageLat_V and SetVantageLon_V.

```
struct  LatLonValue
{
   short bUseFractionalDegrees;
   float fractionalDegrees;
   short degrees;
   short minutes;
   short seconds;
};
```

TxConfiguration

Holds the transmiter configuration data read from the VantagePro console, or to be written to it. The data fields are filled in by GetVantageTxConfig_V and written to the weather station by SetVantageTxConfig_V.

For each of the 8 transmitter ID's the corresponding txType entry indicates the selected Weather Transmitter Type, and the repeater entry indicates the selected VantagePro 2 repeater. For the repeater entry, use 0 for no repeater (or for VantagePro 1 systems), or a value from 1 to 8 to select repeater A through H.

Note: The txType values used by the DLL do not match the ones specified in the Vantage Programmers reference. The DLL will determine the weather station firmware version and write the correct values to the station.

```
struct  TxConfiguration
{
   short txType[8];
   short repeater[8];
};
```

BarCalData

Holds information about the current sea-level correction for the barometer sensor on the VantagePro console. The data values are filled in by GetBarometerData_V.

Barometer, temperature, and elevation values are given in the current DLL units. The correntBarometer value is the most recently measured barometer reading (normally updated every 15 minutes) corrected to sea-level. To determine the raw reading, subtract the barCalibrationOffset from it and divide the result by the barCalibrationRatio.

```
struct  BarCalData
{
   float currentBarometer;
   float elevation;
   float dewPoint;
   float virtualTemp;
   short int humCfactor;
   float barCalibrationRatio;
   float barCalibrationOffset;
};
```

TimeZoneSettings

Holds information about the Time Zone and Daylight Savings settings on the VantagePro console. This structure is used by the functions GetTimeZoneSettings_V and SetTimeZoneSettings_V.

The time zone can either be specified from a list (with bUseTimeZoneList = 1 and timeZone = a constant from the Time Zone List) or by a GMT/UTC offset (with bUseTimeZoneList = 0 and GMToffset = number of hours that clocks should be set ahead or behind GMT/UTC). The GMTofset value is rounded to the nearest 15 minute (0.25 hour) time. The time zone set should be based on the "Standard" time zone. Adjustments for daylight savings are taken into consideration by the two daylight savings settings.

Set bAutoDaylightSavings = 1 to have the console automatically switch Daylight Savings on or off. The exact dates that this occurs vary depending on the latitude/longitude settings. See the WeatherLink on-line help file for more details.
Set bAutoDaylightSavings = 0 to control daylight savings mode manually. (i.e. your location does not observe daylight savings time, or does not use the same starting and ending dates that the VantagePro console uses.)

bDaylightSavingsOnNow will indicate whether daylight savings time is currently in effect.
If automatic daylight savings mode is selected, GetTimeZoneSettings_V will return the current status, and the value is ignored by SetTimeZoneSettings_V.
If manual daylight savings mode is selected, the value is used by both GetTimeZoneSettings_V and SetTimeZoneSettings_V.

Note: Older versions (prior to Nov 20, 2005 for VP1 or Nov 28, 2005 for VP2) of the VantagePro firmware do not account for the changes in the daylight savings starting and ending dates in the US that are scheduled to take effect fall 2007. This change should not affect European and Australian weather stations.

```
struct  TimeZoneSettings
{
   short bUseTimeZoneList;
   short timeZone;
   float GMToffset;
   short bAutoDaylightSavings;
   short bDaylightSavingsOnNow;
};
```

DLL Constants

Units Constants

INCHES	0
MM	1
MB	2
HECTO_PASCAL	3
FAHRENHEIT	0
CELSIUS	1
MPH	0
KNOTS	1
KPH	2
METERS_PER_SECOND	3
KM	1
MILES	0
FT	1
M	0

DLL Return values (error codes)

COM_ERROR	-101	Indicates that there was a problem in communication with the VantagePro station. Check your com port settings, and, if necessary, increase the value of timeout using SetCommTimeOutVal_V(), or use SetVantageTimeoutVal_V() instead.
MEMORY_ERROR	-102	
COM_OPEN_ERROR	-103	
NOT_LOADED_ERROR	-104	Indicates that the appropriate "Load___" function has not been called.
BAD_DATA	-32768 = 0x80000	Indicates that either the sensor is not connected, or sensor is connected but does not have a valid value.
---	0	Usually indicates success.
---	-1	Usually indicates failure.

The TimeOutType could be one of those defined constants

`TO_STANDARD`	0 // Standard
`TO_DUMP_AFTER`	1
`TO_MODEM`	2
`TO_LOOPBACK`	3
`TO_LOOP`	4
`TO_FLUSH`	5
`TO_DONE`	6
`TO_STANDARD_MODEM`	7
`TO_STANDARD_MONITOR`	8
`TO_STANDARD_MONITOR_MODEM`	9
`TO_AUTO_DETECT`	10

Rain Collector types

ENGLISH_10	0 // one 10'th of an inch
ENGLISH_100	1 // one 100'th of an inch
METRIC_5	2 // one 5'th of a mm
METRIC_1	3 // 1 mm
ENGLISH_OTHER	4 // other inch
METRIC_OTHER	5 // other mm

Wind Direction Values:

"N" "NNE" "NE" "ENE" "E" "ESE" "SE" "SSE" "S" "SSW" "SW" "WSW" "W" "WNW" "NW" "NNW"
"---" if Wind Direction is not available

Weather Transmitter Types:

Note: The numerical values of the transmitter types listed below do not match those documented in the VantagePro Technical Reference.

The VantagePro DLL uses the same constants for all VantagePro firmware versions and takes care of translating and differences. If a requested station type is not available on the VantagePro console, either an appropriate alternitive will be selected, or the transmitter will not be enabled. The WeatherLink software documentation contains more information about which stations are available with different Vantage firmware versions.

NO_TX	No transmiter on this ID
ISS_TX	An ISS station configured as an ISS. Only one allowed per VantagePro.
TEMP_ONLY_TX	Temperature only station
TEMP_HUM_TX	Temperature/humidity station
WIND_TX	Wireless Anemometer station
LEAF_TX	Leaf Wetness/Temperature station
SOIL_TX	Soil Moisture/Temperature station
LEAF_SOIL_TX	Combined Leaf and Soil station
RETRANSMIT_TX	VantagePro will retransmit ISS data on this ID

Rain and ET Types:

Use these constants when selecting the rain or ET value to clear with the ClearVantageRainET_V function.

DAY_RAIN_VALUE	Day Rain Total
STORM_RAIN_VALUE	Rain Storm Total
MONTH_RAIN_VALUE	Month Rain Total
YEAR_RAIN_VALUE	Year Rain Total
DAY_ET_VALUE	Day ET Total
MONTH_ET_VALUE	Month ET Total
YEAR_ET_VALUE	Year ET Total

Wind Cup Sizes:

Use these constants when setting or examining the wind cup size with the functions GetWindCupSize_V() or SetWindCupSize_V(). The large wind cup size is what is shipped with the weather station. The small cup size should only be used in applications where large wind speeds are expected.

LARGE_WIND_CUPS	The default wind cup size. They have a lower start up threshold than the small cups.
SMALL_WIND_CUPS	These shipped with our older line of weather stations and should only be used where large wind speeds are expected.
OTHER_WIND_CUPS	This is new for firmware 1.80 or later, and the Vue station.

Temperature Averaging Settings:

Use these constants when setting or examining the temperature averageing setting with the functions GetTempAvg_V() or SetTempAvg_V()(). It controls how temperature values logged in the archive memory are recorded. This feathure is only available on VantagePro stations with firmware dates on or after July 18, 2001.

TEMPERATURE_SAMPLED	The default setting. The temperature value at the end of the archive period is recorded.
TEMPERATURE_AVERAGED	The average teperature over the archive period is recorded.

Time Zone List:

Use these constants in the timezone field of the TimeZoneSetting structure to select a timezone from the pre-defined list. This list matches the one in the VantagePro console's setup screen.

Symbolic Name	Value	GMT / UTC Offset	Time Zone Name
TZ_ENIWETOK	0	-12:00	Eniwetok, Kwajalein
TZ_MIDWAY	1	-11:00	Midway Island, Samoa
TZ_HAWAII	2	-10:00	Hawaii
TZ_ALASKA	3	-09:00	Alaska
TZ_PACIFIC	4	-08:00	Pacific Time, Tijuana
TZ_MOUNTAIN	5	-07:00	Mountain Time
TZ_CENTRAL	6	-06:00	Central Time
TZ_MEXICO_CITY	7	-06:00	Mexico City
TZ_CENTRAL_AMERICA	8	-06:00	Central America
TZ_BOGOTA	9	-05:00	Bogota, Lima, Quito
TZ_EASTERN	10	-05:00	Eastern Time
TZ_ATLANTIC	11	-04:00	Atlantic Time
TZ_CARACAS	12	-04:00	Caracas, La Paz, Santiago
TZ_NEWFOUNDLAND	13	-03:30	Newfoundland
TZ_BRASILIA	14	-03:00	Brasilia
TZ_BUENOS_AIRES	15	-03:00	Buenos Aires, Georgetown, Greenland
TZ_MID_ATLANTIC	16	-02:00	Mid-Atlantic
TZ_AZORES	17	-01:00	Azores, Cape Verde Is.
TZ_GMT	18	00:00	Greenwich Mean Time, Dublin, Edinburgh, Lisbon, London
TZ_MONROVIA	19	00:00	Monrovia, Casablanca
TZ_BERLIN	20	+01:00	Berlin, Rome, Amsterdam, Bern, Stockholm, Vienna
TZ_PARIS	21	+01:00	Paris, Madrid, Brussels, Copenhagen, W Central Africa
TZ_PRAGUE	22	+01:00	Prague, Belgrade, Bratislava, Budapest, Ljubljana
TZ_ATHENS	23	+02:00	Athens, Helsinki, Istanbul, Minsk, Riga, Tallinn
TZ_CAIRO	24	+02:00	Cairo
TZ_EAST_EUROPE	25	+02:00	Eastern Europe, Bucharest
TZ_PRETORIA	26	+02:00	Harare, Pretoria
TZ_ISRAEL	27	+02:00	Israel, Jerusalem
TZ_BAGHDAD	28	+03:00	Baghdad, Kuwait, Nairobi, Riyadh
TZ_MOSCOW	29	+03:00	Moscow, St. Petersburg, Volgograd
TZ_TEHRAN	30	+03:30	Tehran
TZ_ABU_DHABI	31	+04:00	Abu Dhabi, Muscat, Baku, Tblisi, Yerevan, Kazan
TZ_KABUL	32	+04:30	Kabul
TZ_ISLAMABAD	33	+05:00	Islamabad, Karachi, Ekaterinburg, Tashkent
TZ_BOMBAY	34	+05:30	Bombay, Calcutta, Madras, New Delhi, Chennai
TZ_COLOMBO	35	+06:00	Almaty, Dhaka, Colombo, Novosibirsk, Astana
TZ_BANGKOK	36	+07:00	Bangkok, Jakarta, Hanoi, Krasnoyarsk
TZ_BEIJING	37	+08:00	Beijing, Chongqing, Urumqi, Irkutsk, Ulaan Bataar
TZ_HONG_KONG	38	+08:00	Hong Kong, Perth, Singapore, Taipei, Kuala Lumpur
TZ_TOKYO	39	+09:00	Tokyo, Osaka, Sapporo, Seoul, Yakutsk
TZ_ADELAIDE	40	+09:30	Adelaide
TZ_DARWIN	41	+09:30	Darwin
TZ_BRISBANE	42	+10:00	Brisbane, Melbourne, Sydney, Canberra
TZ_GUAM	43	+10:00	Hobart, Guam, Port Moresby, Vladivostok
TZ_SOLOMON_ISLANDS	44	+11:00	Magadan, Solomon Is, New Caledonia
TZ_FIJI	45	+12:00	Fiji, Kamchatka, Marshall Is.
TZ_WELLINGTOM	46	+12:00	Wellington, Auckland

WeatherDataID Values:

The following constants are used to select particular weather data parameters to retrieve with the GetCurrentDataByID_V(), GetCurrentDataStrByID_V(),

GetHiLowDataByID_V(), GetHiLowDataByID_V(), GetHiLowTimeByID_V(), and GetHiLowTimeByID_V() functions.

These constants are defined in the header file WeatherDataID.h, and are mostly the same values as used in the WeatherLink Expansion Module SDK. The entries marked with "**" are ones where there are differences.

List of Weather Data ID Tables

Normal Weather Parameters (Temperature, Humidity, etc) Current and Day values

The weather data ID's for month and year high/low values are formed by replacing "DAY_" with "MONTH_" or "YEAR_" in the corresponding day high/low value, if it is available.

Weather Parameter	Current Value	Day High Value	Day Low Value	Month High Value	Month Low Value	Year High Value	Year Low Value
Outside Temperature	CUR_OUT_TEMP_WDID	DAY_HI_OUT_TEMP_WDID	DAY_LOW_OUT_TEMP_WDID	X	X	X	X
Inside Temperature	CUR_IN_TEMP_WDID	DAY_HI_IN_TEMP_WDID	DAY_LOW_IN_TEMP_WDID	X	X	X	X
Outside Humidity	CUR_OUT_HUM_WDID	DAY_HI_OUT_HUM_WDID	DAY_LOW_OUT_HUM_WDID	X	X	X	X
Inside Humidity	CUR_IN_HUM_WDID	DAY_HI_IN_HUM_WDID	DAY_LOW_IN_HUM_WDID	X	X	X	X
Outside Dew Point	CUR_OUT_DEW_WDID	DAY_HI_OUT_DEW_WDID	DAY_LOW_OUT_DEW_WDID	X	X	X	X
Wind Chill	CUR_WIND_CHILL_WDID	---	DAY_LOW_WIND_CHILL_WDID	---	X	---	X
Outside Heat Index	CUR_OUT_HEAT_WDID	DAY_HI_OUT_HEAT_WDID	DAY_LOW_OUT_HEAT_WDID	X	---	X	---
Outside THW Index	CUR_OUT_THW_WDID	---	---	---	---	---	---
Outside THSW Index	CUR_OUT_THSW_WDID	DAY_HI_OUT_THSW_WDID	---	X	---	X	---
Inside Dew Point	CUR_IN_DEW_WDID	---	---	---	---	---	---
Wind Speed	CUR_SPEED_WDID	DAY_HI_SPEED_WDID	---	X	---	X	---
2 min. Avg. Speed	**WIND_2MIN_AVG_WDID**	---	---	---	---	---	---
10 min. Avg. Speed	CUR_AVG_SPEED_WDID	---	---	---	---	---	---
10 min. Wind Gust	**WIND_GUST_10MIN_WDID**	---	---	---	---	---	---
Wind Direction	CUR_WIND_DIR_WDID	---	---	---	---	---	---
Wind Direction Sector	CUR_WIND_DIR_SECTOR_WDID	---	---	---	---	---	---
Barometer	CUR_SEA_LEV_BAR_WDID	DAY_HI_SEA_LEV_BAR_WDID	DAY_LOW_SEA_LEV_BAR_WDID	X	X	X	X
Barometer Trend	CUR_SEA_LEV_BAR_TREND_WDID	---	---	---	---	---	---
Bar Altimeter	**BAR_ALTIMETER_WDID**	---	---	---	---	---	---
Rain Rate	CUR_HI_RATE_WDID	DAY_HI_RATE_WDID	---	X	---	X	---
Solar Radiation	CUR_SOLAR_RAD_WDID	DAY_HI_SOLAR_RAD_WDID	---	X	---	X	---
UV	CUR_UV_WDID	DAY_HI_UV_WDID	---	X	---	X	---
Temperature 2	CUR_TEMP_2_WDID	DAY_HI_TEMP_2_WDID	DAY_LOW_TEMP_2_WDID	X	X	X	X
Temperature 3	CUR_TEMP_3_WDID	DAY_HI_TEMP_3_WDID	DAY_LOW_TEMP_3_WDID	X	X	X	X
Temperature 4	CUR_TEMP_4_WDID	DAY_HI_TEMP_4_WDID	DAY_LOW_TEMP_4_WDID	X	X	X	X
Temperature 5	CUR_TEMP_5_WDID	DAY_HI_TEMP_5_WDID	DAY_LOW_TEMP_5_WDID	X	X	X	X
Temperature 6	CUR_TEMP_6_WDID	DAY_HI_TEMP_6_WDID	DAY_LOW_TEMP_6_WDID	X	X	X	X
Temperature 7	CUR_TEMP_7_WDID	DAY_HI_TEMP_7_WDID	DAY_LOW_TEMP_7_WDID	X	X	X	X
Temperature 8	CUR_TEMP_8_WDID	DAY_HI_TEMP_8_WDID	DAY_LOW_TEMP_8_WDID	X	X	X	X
Humidity 2	CUR_HUM_2_WDID	DAY_HI_HUM_2_WDID	DAY_LOW_HUM_2_WDID	X	X	X	X
Humidity 3	CUR_HUM_3_WDID	DAY_HI_HUM_3_WDID	DAY_LOW_HUM_3_WDID	X	X	X	X
Humidity 4	CUR_HUM_4_WDID	DAY_HI_HUM_4_WDID	DAY_LOW_HUM_4_WDID	X	X	X	X

Humidity 5	CUR_HUM_5_WDID	DAY_HI_HUM_5_WDID	DAY_LOW_HUM_5_WDID	X	X	X	X
Humidity 6	CUR_HUM_6_WDID	DAY_HI_HUM_6_WDID	DAY_LOW_HUM_6_WDID	X	X	X	X
Humidity 7	CUR_HUM_7_WDID	DAY_HI_HUM_7_WDID	DAY_LOW_HUM_7_WDID	X	X	X	X
Humidity 8	CUR_HUM_8_WDID	DAY_HI_HUM_8_WDID	DAY_LOW_HUM_8_WDID	X	X	X	X
Soil Moisture 1	CUR_SOIL_M_1_WDID	DAY_HI_SOIL_M_1_WDID	DAY_LOW_SOIL_M_1_WDID	X	X	X	X
Soil Moisture 2	CUR_SOIL_M_2_WDID	DAY_HI_SOIL_M_2_WDID	DAY_LOW_SOIL_M_2_WDID	X	X	X	X
Soil Moisture 3	CUR_SOIL_M_3_WDID	DAY_HI_SOIL_M_3_WDID	DAY_LOW_SOIL_M_3_WDID	X	X	X	X
Soil Moisture 4	CUR_SOIL_M_4_WDID	DAY_HI_SOIL_M_4_WDID	DAY_LOW_SOIL_M_4_WDID	X	X	X	X
Soil Temp 1	CUR_SOIL_T_1_WDID	DAY_HI_SOIL_T_1_WDID	DAY_LOW_SOIL_T_1_WDID	X	X	X	X
Soil Temp 2	CUR_SOIL_T_2_WDID	DAY_HI_SOIL_T_2_WDID	DAY_LOW_SOIL_T_2_WDID	X	X	X	X
Soil Temp 3	CUR_SOIL_T_3_WDID	DAY_HI_SOIL_T_3_WDID	DAY_LOW_SOIL_T_3_WDID	X	X	X	X
Soil Temp 4	CUR_SOIL_T_4_WDID	DAY_HI_SOIL_T_4_WDID	DAY_LOW_SOIL_T_4_WDID	X	X	X	X
Leaf Wetness 1	CUR_LEAF_W_1_WDID	DAY_HI_LEAF_W_1_WDID	DAY_LOW_LEAF_W_1_WDID	X	X	X	X
Leaf Wetness 2	CUR_LEAF_W_2_WDID	DAY_HI_LEAF_W_2_WDID	DAY_LOW_LEAF_W_2_WDID	X	X	X	X
Leaf Temp 1	CUR_LEAF_T_1_WDID	DAY_HI_LEAF_T_1_WDID	DAY_LOW_LEAF_T_1_WDID	X	X	X	X
Leaf Temp 2	CUR_LEAF_T_2_WDID	DAY_HI_LEAF_T_2_WDID	DAY_LOW_LEAF_T_2_WDID	X	X	X	X

Accumulation Parameters (rain, ET, etc)

Weather Parameter	Day Value	Storm Value	Month Value	Year Value	Last Hour	Last 24 Hour
Rain	DAY_RAIN_WDID	STORM_RAIN_WDID	MONTH_RAIN_WDID	YEAR_RAIN_WDID	LAST_HOUR_RAIN_WDID	LAST_24HOUR_RAIN_WDID
ET	DAY_ET_WDID	---	MONTH_ET_WDID	YEAR_ET_WDID		

Other Parameters

Current Battery Voltage	CONSOLE_BATT_WDID
Transmitter Battery Status	TX_BATT_WDID
Repeater Battery Status (VP2 only)	REPEATER_BATT_WDID
Current Weather Forecast	FORECAST_WDID
Time of Sunrise	SUNRISE_WDID
Time of Sunset	SUNSET_WDID

DLL Function Details

Initialization Functions

The following functions are used to open/close communication with Vantage and to read or modify settings in the DLL. These functions do not communicate with the VantagePro console.

Opening and Closing the Communication Port
Use the funtion OpenCommPort_V (or OpenUSBPort_V for USB WeatherLink device, or OpenTCPIPPort_V for TCPIP WeatherIP) to open the communication channel to the VantagePro console before calling any function that communicates with the station (all functions that are not part of the Initialization Functions group). For USB device, GetUSBDevSerialNumber_V() sould be called to the get serial number parameter for calling the OpenUSBPort_V command.

After calling OpenCommPort(or CloseUSBPort_V for USB WeatherLink device, or CloseTCPIPPort_V for TCPIP Weather IP device), the function InitStation_V should be called to verify that a station is connected to the serial port and to configure the DLL operations for the firmware version detected.

Use the function CloseCommPort_V (or CloseUSBPort_V, or CloseTCPIPPort_V) to close the channel when you are done communicating with the station. While the DLL holds the channel open, other programs (such as WeatherLink) can not talk to the weather station.

Units
Use the function SetUnits_V to select which units of measurements the DLL should use to report weather data. All weather data values read from or written to DLL function are in the selected units.

Rain Collector Model
Use the function SetRainCollectorModel_V or the function GetAndSetRainCollectorModel_V to configure the rain collector type that is being used by your weather station. The rain collector model is used to calculate all rain and rain rate data values and you will get incorrect values if you are using the wrong setting.
The default setting is the 0.01 inches rain collector.

Note: The rain collector model is independant from the rain display units. You can display the rain totals from both a 0.01 inch rain collecter or a 0.2 mm rain collector in either inches or mm.

Note: The function GetAndSetRainCollectorModel_V reads the rain collector model from the weather station, so technically it is NOT an Initialization Function.

---

float GetDllVersion_V ()
Description
Gets the Dll version number.
Return Values
the DLL version number, e.g., 1.0

Dll Functions
Constants

---

short int OpenCommPort_V (short int comPort, int baudRate)
Description
Use this function to open the com port. e.g., OpenCommPort_V(1,19200).
Use CloseCommPort_V to close the port.when you are done.
Parameters
comPort - number of the port to be opened, depending upon the connection with Vantage.
baudRate - This can be 9600 or 19200.

This function must be called before any other function in the Dll that talks to the VantagePro console (i.e. all functions that are not "Initialization Functions"). .

Return Values
0 if successful
MEMORY_ERROR if system is low on memory
COM_ERROR if communication error
Dll Functions
Constants

---

short int OpenDefaultCommPort_V( )
Description
Use this function to open the default com port as 1, and default baud rate as 19200.
Use CloseCommPort_V to close the port.when you are done.
Return Values
0 if successful
COM_OPEN_ERROR if com is already open
MEMORY_ERROR if system is low on memory
COM_ERROR if communication error

Dll Functions
Constants

---

short int OpenUSBPort_V (int usbSerialNumber)
Description
Use this function to open the port of the USB WeatherLink device.
Use CloseUSBPort_V or CloseCommPort_V to close the port when you are done.
Parameters
usbSerialNumber - serial number of the USB WeatherLink device. Use the function GetUSBSerialNumber to determine the Serial number of a USB Datalogger. Only one USB Datalogger should be connected during the detection.

This function must be called before any other functions in the Dll that talk to the USB VantagePro console (i.e. all functions that are not "Initialization Functions").

Return Values
0 if successful
MEMORY_ERROR if system is low on memory
COM_ERROR if communication error
Dll Functions
Constants

---

short int GetUSBSerialNumber()
Description
Use this function to get the serial number for the 1st USB WeatherLink device the software detects. The serial number is to be used for opening the USB WeatherLink device.

Return Values
USB serial number if successful
0 if not successful
Dll Functions
Constants

---

short int OpenTCPIPPort_V (const char *TCPPort, const char *IPAddr)
Description
Use this function to open the port of the TCPIP WeatherLink IP device. Example:

OpenTCPIPPort_V("22222", "00:1D:0A:00:00:4A");

OpenTCPIPPort_V("22222", "172.16.24.12").

The software automatically detects the 2nd parameter as either a Device ID or an IP Address by looking at the format of the string.

Use CloseTCPIPPort_V or CloseCommPort_V to close the port when you are done.
Parameters
TCPPort - This is the port number, usually "22222".
IPAddr - This can be wither the WeatherLink Device ID or its IP Address.

This function must be called before any other function in the Dll that talks to the WeatherLink IP VantagePro console (i.e. all functions that are not "Initialization Functions"). .

Return Values
0 if successful
COM_OPEN_ERROR if com is already open
MEMORY_ERROR if system is low on memory
COM_ERROR if communication error
Dll Functions
Constants

---

short int CloseCommPort_V()

Description
Closes the com-port opened by OpenCommPort_V / OpenDefaultCommPort_V. This function should only be used after opening the communication port. Starting with DLL 2.40, this function also close the USB WeatherLink port or the TCPIP port.
Once CloseCommPort_V is called, the port must be reopened before you can communicate with the VantagePro console.
Return Values
0 if successful
COM_ERROR if com was already closed

Dll Functions
Constants

---

short int CloseUSBPort_V()
Description
Closes the USB-port opened by OpenUSBPort_V. This function should only be used after opening the com port.
Once CloseUSBPort_V is called, the port must be reopened before you can communicate with the VantagePro console.
Return Values.
0 if successful
COM_ERROR if com was already closed

Dll Functions
Constants

---

short int CloseTCPIPPort_V()
Description
Closes the TCPIP-port opened by OpenTCPIPPort_V. This function should only be used after opening the TCPIP port.
Once CloseTCPIPPort_V is called, the port must be re-opened before you can communicate with the VantagePro console.
Return Values
0 if successful
COM_ERROR if com was already closed

Dll Functions
Constants

---

short int SetCommTimeoutVal_V(short int ReadTimeout, short int WriteTimeout)
Parameters
ReadTimeout in milliseconds
WriteTimeout in milliseconds
Description
In version 2.4, this is routed to SetVantageTimeoutVal_V (TO_STANDARD). The signature remains, and users are encourage to use SetVantageTimeoutVal_V() in place of this.
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

short int SetVantageTimeoutVal_V(short int TimeOutType)
Parameters
TypeOutType

The TimeOutType could be one of those defined constants

```
#define  TO_STANDARD       (0)
#define  TO_DUMP_AFTER     (1)
#define  TO_MODEM          (2)
```

```
#define  TO_LOOPBACK       (3)
#define  TO_LOOP           (4)
#define  TO_FLUSH          (5)
#define  TO_DONE           (6)
#define  TO_STANDARD_MODEM (7)
#define  TO_STANDARD_MONITOR  (8)
#define  TO_STANDARD_MONITOR_MODEM  (9)
#define  TO_AUTO_DETECT       (10)
```

Description
The default values for sending a character and receiving a character are set to 4 seconds in the Dll. Use this function to change the time out values.

This function should only be used after opening the communication port. This is the recommended method to use over SetCommTimeoutVal_V()
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short void GetUnits_V(WeatherUnits *units)
Description
Fills the structure WeatherUnits with the units currently being used in the Dll. For DLL version 2.42 and later, this value is actually obtained from the console firmware settings.
All weather data will be read and reported by the DLL in these units.
The default DLL units are:

Temperature	FAHRENHEIT
Barometer	INCHES
Rain	INCHES
Wind	MPH
Elevation	FEET

Dll Functions
Constants

---

short int SetUnits_V(WeatherUnits *units)
Description
The Dll sets the units used to read and report weather data to the values specified in the WeatherUnits structure. For DLL version 2.42 and later, this actually set the console firmware settings.

Parameters
units - WeatherUnits structure to set the weather units.
Return Values
0 if successful
-1 if invalid data

Dll Functions
Constants

---

short int SetRainCollectorModel_V(char rainCModel)
Description
This function sets the rain collector model used by the DLL to the specified type.
The default value is ENGLISH_100

Note: SetRainCollectorModel_V does NOT set the rain collector type setting on the VantagePro console. Use

SetRainCollectorModelOnStation_V to configure the station's rain collecter setting.

Return Values
0 if successful
-1 if invalid data

Dll Functions
Constants

---

short int GetRainCollectorModel_V()
Description
Returns the rain collector model value currently being used in the Dll.

Note: GetRainCollectorModel_V returns the rain collector type setting from the DLL, NOT the setting on the VantagePro console. Use GetRainCollectorModelOnStation_V to read the the station's rain collecter setting or GetAndSetRainCollectorModelOnStation_V to set the DLL's rain collecter seting to match the station's value.

Return Values
BAD_DATA if data is not available

Dll Functions
Constants

Lowlevel Functions

Normally the Dll functions should be sufficient to communicate with the Weather station and obtain data. The low level functions provide additional flexibility which lets a user send commands directly and retrieve data. The communication port has to be opened first before using any of these functions.

Dll Functions
Constants

---

short int GetSerialChar_V ()
Description
Retrieves a character from the serial port.
Return Values
0 if successful
-1 if error

Dll Functions
Constants

---

short int PutSerialStr_V (char *s)
Description
Sends the character string s to the serial port.
Return Values
0 if successful
-1 if error

Dll Functions
Constants

---

short int PutSerialChar_V (unsigned char c)
Description
Sends a character to the serial port
Return Values
0 if successful

-1 if error

Dll Functions
Constants

Station Configuration Functions

The functions listed below are used to read and write settings on the VantagePro console. Thes functions do not require the function LoadCurrentVantageData_V to be called first.

The communication port must already be opened with the OpenCommPort_V, OpenDefaultCommPort_V, OpenUSBPort_V, or OpenTCPIPPort_V functions. In addition, some functions require calling InitStation_V in order to correctly accommodate variations in the data format of different Vantage firmware versions.

---

short int InitStation_V()
Description
Reads the firmware version date from the station and configures the DLL
Return Values
Firmware date.
COM_ERROR if error in getting Firmware Date.

Dll Functions
Constants

---

short int GetModelNo_V()
Description
Returns the model type of the Weather Station.
Return Values
16 is the model number of VantagePro, VantagePro2, Envoy, and Envoy2 stations. 17 is the model number of the Vantage Vue station.
COM_ERROR if error in getting Model number

Dll Functions
Constants

---

short int GetStationTime_V (DateTimeStamp &timeStamp)
Description
Fills the timeStamp struct with current weather station time. The range for the hour is 0-23.
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int SetStationTime_V (DateTimeStamp &timeStamp)
Description
Sets the station time with the values of timeStamp struct.
Return Values
COM_ERROR if error in setting station time
0 if successful

Dll Functions
Constants

---

short int GetArchivePeriod_V ( )
Description
Gets the archive interval in minutes on the station.The possible values are 1,5,10,15,30,60,120 .
Return Values
The archive period in minutes, if successful
COM_ERROR if error in getting archive period

Dll Functions
Constants

---

short int SetArchivePeriod_V (int intervalCode)
Description
Sets the archive interval in minutes on the station.
Parameters
Valid Values 1, 5, 10, 15, 30, 60, 120
Return Values
-1 upon invalid intervalCode
COM_ERROR if error occurs in setting
0 if successful

Dll Functions
Constants

---

short int PutTotalRain_V (short totalRain)
Description
Sets the total rain of the Weather Station.
Return Values
COM_ERROR if error ocuurs in setting.
0 if successful

Dll Functions
Constants

---

short int GetStationFirmwareDate_V(DateTimeStamp &timeStamp)
Description
Fills the timeStamp parameter with the curent firmware date read from the VantagePro console. Only the date fields will be set. The time fields are set to midnight (00:00).
The functions available on the VantagePro depend on the firmware dates. See the VantagePro Programmer's Reference for more details.
Return Values

The date of the firmware.
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetStationFirmwareVersion_V(char *versionSt)
Description
Fills the timeStamp parameter with the curent firmware version read from the VantagePro console.

Return Values
The version of the firmware.
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetReceptionData_V(ReceptionStats &receptionStats)
Description
Fills in the receptionStats parameter with the current ISS or Wireless Anemometer reception data. These values represent the statistics since midnight, since the last time the console was put into the setup screens, or since the reception data was cleared on the console (by holding the clear button when the diagnostics screen was being shown) whichever is most recent.
Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetVantageLat_V(LatLonValue &latitude)
Description
Fills the LatLonValue structure with the current lattitude value that is currently set on the Console.
Both the floating point fractionalDegrees and the integer degree/minute/seconds fields are set.
Positive values represent Northern lattitudes, and negative calues represent Sothern latitudes. For the integral degree/minute/seconds fields, only the degree number is negative, the minutes and seconds fields are positive numbers (i.e. 90°30'00" S = -90 degrees, 30 minutes, 00 seconds, -90.5 fractionalDegrees)

The Vantage console only stores latitude to 1/10 of a degree (6 minute) resolution.

Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int SetVantageLat_V(LatLonValue &latitude)
Description
Sets the latitude on the VantagePro console to the value specified in the latitude parameter. The value of the bUseFractionalDegrees field determines whether the latitude value is taken from the fractionalDegrees field or from the degree/minute/seconds fields.

The Vantage console only stores latitude to 1/10 of a degree (6 minute) resolution.

Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetVantageLon_V(LatLonValue &longitude)
Description
Fills the LatLonValue structure with the current longitude value that is currently set on the Console.
Both the floating point fractionalDegrees and the integer degree/minute/seconds fields are set.
Positive values represent Eastern longitudes, and negative calues represent Western latitudes. For the integral degree/minute/seconds fields, only the degree number is negative, the minutes and seconds fields are positive numbers (i.e. 90°30'00" W = -90 degrees, 30 minutes, 00 seconds, -90.5 fractionalDegrees)

The Vantage console only stores longitude to 1/10 of a degree (6 minute) resolution.

Return Values
BAD_DATA if data is not available

Dll Functions

Constants

---

short int SetVantageLon_V(LatLonValue &longitude)
Description
Sets the longitude on the VantagePro console to the value specified in the longitude parameter. The value of the bUseFractionalDegrees field determines whether the longitude value is taken from the fractionalDegrees field or from the degree/minute/seconds fields.

The Vantage console only stores longitude to 1/10 of a degree (6 minute) resolution.

Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int SetVantageLamp_V(short int lampState)
Description
Turns the console lamp ON (lampState = 1) or OFF (lampState = 0).
Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int SetNewBaud_V(short int baud)
Description
Changes the baud rate of VantagePro 1 consoles and all Envoy versions. This is the only way to change the baud rate of an Envoy since it does not have a keyboard/LCD interface. The serial port should be configured with the baud rate that the station is currently set to. If the baud rate change fails, then either the DLL or the application should reset the baud rate back to the previous value. This command is not supported on the VantagePro 2 console.
Parameters
baud - The new target baud rate. Valid values are: 1200, 2400, 4800, 9600, 14400, and 19200.

Return Values
COM_ERROR error communicating with station

Dll Functions
Constants

---

short int GetVantageTxConfig_V(TxConfiguration *txConfig)
Description
Fills in the TxConfiguration structure with the current transmitter configuration and repeater configuration (VantagePro 2 only) from the VantagePro console.

Note: The function InitStation_V must be called before calling GetVantageTxConfig_V in order for the DLL to know which set of Weather transmitters are supported by the VantagePro console.

Return Values
PARAMETER_ERROR - txConfig is a NULL pointer
COM_ERROR - error communicating with station
NOT_LOADED_ERROR - InitStation_V has not been called

Dll Functions
Constants

---

short int SetVantageTxConfig_V(TxConfiguration &txConfig)

Description
Uses the values in the txConfig parameter to configure the transmitter and repeater settings on the VantagePro console.

Note: The function InitStation_V must be called before calling GetVantageTxConfig_V in order for the DLL to know which set of Weather transmitters are supported by the VantagePro console.

Return Values
PARAMETER_ERROR txConfig is a NULL pointer, or if the requested transmiter types are not valid.
COM_ERROR - error communicating with station
NOT_LOADED_ERROR - InitStation_V has not been called

Dll Functions
Constants

---

short int GetBarometerData_V(BarCalData &barCalData)
Description
Returns information about the current vaules used to perform the sea-level correction of the barometer sensor.
Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int SetRainCollectorModelOnStation_V(char rainCModel)
Description
This function sets the rain collector model on the VantagePro console to the specified type.

Note: SetRainCollectorModelOnStation_V does not change the setting used by the DLL. Use SetRainCollectorModel_V to configure the DLL to a particular rain collector, or GetAndSetRainCollectorModelOnStation_V to have the DLL use the same setting used by the VantagePro console.

Return Values
0 if successful
-1 if invalid data

Dll Functions
Constants

---

short int GetRainCollectorModelOnStation_V()
Description
Returns the rain collector model value currently configures on the VantagePro console.

Note: The value returned by GetRainCollectorModelOnStation_V may not be the setting used by the DLL. Use GetRainCollectorModel_V to examine the rain collector setting currently in use by the DLL.

Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetAndSetRainCollectorModelOnStation_V()
Description
Reads the rain collector model currently being used by the VantagePro console and configures the Dll to use the same setting.

Return Values
BAD_DATA if data is not available

Dll Functions

Constants

---

short int GetWindCupSize_V()
Description
Reads the Wind Cup Size setting off of the VantagePro console.

Return Values
LARGE_WIND_CUPS
SMALL_WIND_CUPS
OTHER_WIND_CUPS
BAD_DATA if data is not available

Dll Functions
Constants

---

short int SetWindCupSize_V(short int windCupSize)
Description
Sets the Wind Cup Size setting on of the VantagePro console.

Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetTempAvg_V()
Description
Returns whether the archive temperature values are averaged over the archive interval or if the temperature at the end of the interval is recorded.

Return Values
TEMPERATURE_SAMPLED
TEMPERATURE_AVERAGED
BAD_DATA if data is not available

Dll Functions
Constants

---

short int SetTempAvg_V(short int tempAverageSetting)
Description
Configures whether the archive temperature values are averaged over the archive interval or if the temperature at the end of the interval is recorded.

Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetTimeZoneSettings_V(TimeZoneSettings *tzSettings)
Description
Reads the current Time Zone and Daylight Savings settings from the console and stores the values into the TimeZoneSettings data structure.

Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int SetTimeZoneSettings_V(TimeZoneSettings *tzSettings)
Description
Configures the Time Zone and Daylight Savings settings on the console according to the values in the TimeZoneSettings data structure.

Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int IsNewLoopPackageSupported_V()
Description
Check with the console support the new loop package, which gets the following data: Last 2 minutes wind average, last 10 minutes wind gust, last hour rain, last 24 hour rain, and bar-altimeter.

Return Values
FALSE if data is new loop is not supported, otherwise, returns TRUE.

Dll Functions
Constants

Current Data Functions

To retrieve current weather data from the VantagePro console, first call the function LoadVantageData_V to capture a snapshot of all of the current weather conditions, then call one or more of the other "Current Data Functions" to retrieve individual data values. Each call to a Current Data Function will retrieve data captured at the time when the last call to LoadVantageData_V was made.

There are two ways to retrieve current data after calling LoadVantageData_V. All of the functions from previous versions of the DLL are still available for backwards compatibility. This provides a separate function to retreive each of the supported data values.

Note: Not all data values have separate functions. To retrieve a value that does not have a separate function, the weather data ID interface must be used.

The new functions GetCurrentDataByID_V and GetCurrentDataStrByID_V can be used to retreive any current weather data value by using one of the Weather Data ID numbers. This is similar to the approach used in the WeatherLink Expansion Module SDK to specify data values to retrieve from a database record. There are slight differences because the data available from the current data set is different from the data available in a weather archive record.

For example
If you are interested in current values of barometer, inside temperature, outside temperature, and inside humidity, you can use one of the sequences of steps as below.

LoadVantageData_V( )
GetBarometer_V( )
GetInsideTemp_V( )
GetOutsideTemp_V( )
GetInsideHumidity_V( )

Or

LoadVantageData_V( )
GetCurrentDataByID_V(CUR_SEA_LEV_BAR_WDID)
GetCurrentDataByID_V(CUR_IN_TEMP_WDID)
GetCurrentDataByID_V(CUR_OUT_TEMP_WDID)
GetCurrentDataByID_V(CUR_IN_HUM_WDID)

In the above sequence of steps, all the current VantagePro data is loaded into a local cache in step 1, and in the subsequent steps the values are returned from the cache. The DLL only communicates with Vantage in the first step.

Dll Functions
Constants

---

float GetBarometer_V ()
Description
Returns the calibrated barometric pressure.
Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

float GetOutsideTemp_V()
Description
Returns the calibrated outside temperature.
Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

float GetDewPt_V ();
Description
Returns the Dew point.
Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

float GetWindChill_V ()
Description
Returns the wind chill.
Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

float GetInsideTemp_V ()
Description
Returns the calibrated inside temperature.
Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetInsideHumidity_V()
Description
Returns the inside humidity.
Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetOutsideHumidity_V()
Description
Returns the calibrated outside humidity.
Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

float GetTotalRain_V ()
Description
Returns the total rain.
Return Values
total rain.
Dll Functions
Constants

---

float GetDailyRain_V ()
Description
Returns the daily rain.
Return Values
daily rain.

Dll Functions
Constants

---

float GetMonthlyRain_V ()
Description
Returns the monthly rain.
Return Values
monthly rain.

Dll Functions
Constants

---

float GetStormRain_V ()
Description
Returns the Storm rain.
Return Values
Storm rain.

Dll Functions
Constants

---

short int LoadCurrentVantageData_V ()

Description
Loads the all current values from Vantage and stores them into a local cache in the DLL.
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int PutBarometer_V(float bar, short elev)
Parameters
bar - barometer value to set
elev - elevation value
Description
Valid values for bar --- between 20 and 32.5 inches.
Valid values for elevation --- between -2000 and 15000 feet.
If bar value is != 0 and is valid, Sets the current Barometric reading on the station to the bar value. If bar value is 0, and elevation is valid, sets the current Barometric reading by calculating according to the elevation value.
Return Values
0 if successful
-1 if invalid data
COM_ERROR if error

Dll Functions
Constants

---

float GetWindSpeed_V()
Description
Gets the current wind speed.
Return Values
current wind speed.

Dll Functions
Constants

---

short int GetWindDir_V ( )
Description
Gets the wind direction in degrees.
Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

char* GetWindDirStr_V (char* dirStr)
Description
This function gets the currenct wind direction in string representation.
Return Values
current wind direction
"---" represents no data available.

Dll Functions
Constants

---

float GetRainRate_V ( )
Description

This function gets the rain rate value.
Return Values
Rainrate value

Dll Functions
Constants

---

float GetET_V ( )
Description
This function gets the daily ET value.
Return Values
Daily ET value

Dll Functions
Constants

---

float GetMonthlyET_V ( )
Description
This function gets the monthly ET value.
Return Values
Monthly ET value

Dll Functions
Constants

---

float GetYearlyET_V ( )
Description
This function gets the yearly ET value.
Return Values
Yearly ET value

Dll Functions
Constants

---

short int GetSolarRad_V ( )
Description
This function gets the Solar radiation in W/m^2.
Return Values
current solar radiation

Dll Functions
Constants

---

short int GetUV_V ( )
Description
This function returns the current UV value.
Return Values
the current UV value

Dll Functions
Constants

---

short int GetHeatIndex_V ( )
Description
This function gets the current Heat Index value.

Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetActiveAlarms_V ( )
Description
Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetCurrentDataByID_V (long int weatherDataID)
Description
Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

char * GetCurrentDataStrByID_V (long int weatherDataID, char *buffer, short int bufferLength)
Description
Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetStartOfCurrentStorm_V(DateTimeStamp &timeStamp)
Description
Fills the timeStamp parameter with the date that the current rain storm started, if one is currently in progress. Otherwise, the date fields will be set to all zeros. Only the date fields will be set. The time fields are set to midnight (00:00).

Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetSunriseTime_V(DateTimeStamp &timeStamp)
Description
Fills the timeStamp parameter with the sunrise time from the VantagePro console.

Note: After 6:00pm (local time) the sunrise time returned is the one for the next day. Otherwise you would have to wait until after midnight to find out the time of the next day's sunrise.

The Date fields are set to either the current date on the PC, or to the next day's date between 6:00 pm and midnight.

Return Values
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetSunsetTime_V(DateTimeStamp &timeStamp)
Description
Fills the timeStamp parameter with the sunset time from the VantagePro console. The Date fields are set to the current date on the PC

Return Values
BAD_DATA if data is not available

Dll Functions
Constants

Alarm Functions

The following functions are used to manipulate Alarms in Vantage.

Similar to the Current Data functions above, LoadVantageAlarms_V downloads all the current alarm threshold values from the VantragePro console and stores them in a local cache. This function must be called before any of the Get___Alarm, Put___Alarm, or Clear___Alarm functions are called.

Once the current alarm thresholds have been loaded, the Get___Alarm functions can be used to examine the current settings, the Put___Alarm functions can be used to set a new value, and the Clear___Alarm functions can be used to turn off the alarm (i.e. set the threshold to "dashes").

The Put and Clear functions do not change the alarm thresholds on the VantagePro console directly. Instead, they modify the values in the internal cache. The function SetVantageAlarms_V is used to write the values in the current cache (as read from the VantagePro console and modified with the Put and Clear functions) into the weather station.

Note: The ClearVantageAlarms_V function does clear all of the alarm thresholds directly on the console. It does not clear the values in the internal cache. You should call LoadVantageAlarms_V after calling ClearVantageAlarms_V if you intend to make further modifications to the alarm settings.

For example, to find out what the current bar trend (both rising and falling) and the time alarm settings are:

LoadVantageAlarms_V( ) // loads all the Alarms from the VantagePro console
GetBarRiseAlarm_V ( )
GetBarFallAlarm_V ( )
GetTimeAlarm_V ( )

In the above sequence of steps, all the vantage alarm values are loaded into a local cache (a structure) in step 1, and in the subsequent steps the values are returned from the cache.

To set or modify an alarm threshold:

LoadVantageAlarms_V( ) // loads all the Alarms from the VantagePro console
PutHiRainFloodAlarm_V (...) // Set new values for the three Rain totals alarms
PutRainPerDayAlarm_V (...)
PutRainStormAlarm_V (...)
ClearRainRateAlarm_V () // Disable the Rain Rate alarm
SetVantageAlarms_V ( ) // sets all the above alarms on the Vantage

In the above sequence of steps, all the vantage alarm values are stored in a local cache (a structure) in step 1. These values are modified in steps 2 through 5, and in the last step the values are written to the Vantage.

Note: Attempting to set an alarm variable with invalid data results in clearing of that alarm variable.

All alarm threshold values are read and written using the units currently selected for the DLL.

Dll Functions
Constants

---

short int SetVantageAlarms_V()

Description
Sets the alarms for the vantage.
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int LoadVantageAlarms_V ()
Description
This function reads Vantage Alarm values from the device and fills the DLL structure with those values.
Return Values
0 if successful
OM_ERROR if error
Dll Functions
Constants

---

float GetBarRiseAlarm_V ()
Description
Returns the hi barometric alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the hi barometric alarm.
NOT_SET if alarm is not set
Dll Functions
Constants

---

float GetBarFallAlarm_V ()
Description
Returns the low barometric alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the low barometric alarm.
NOT_SET if alarm is not set
Dll Functions
Constants

---

int GetTimeAlarm_V ()
Description
Returns the time alarm value from the DLL alarm structure as int.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the time alarm on the station as int.
For e.g., if alarm is set to 2:30am return value = 230;
if alarm is set to 2:30pm return value = 1430;
NOT_SET if alarm is not set
Dll Functions
Constants

---

char* GetTimeAlarmStr_V ()
Description
Returns the time alarm value from the DLL alarm structure as string.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the time alarm on the station as a string in am/pm format, eg., 2:30am.

"---" if alarm not set
Dll Functions
Constants

---

float GetInsideLowTempAlarm_V()
Description
Returns the inside low temperature alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the low inside temperature alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

float GetInsideHiTempAlarm_V ()
Description
Returns the inside hi temparature alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the high inside temperature alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

float GetOutsideLowTempAlarm_V ()
Description
Returns the outside low temparature alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the low outside temperature alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

float GetOutsideHiTempAlarm_V()
Description
Returns the outside hi temparature alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the high outside temperature alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

short int GetLowInsideHumAlarm_V ()
Description
Returns the low inside humidity alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Returns the low inside humidity alarm
NOT_SET if alarm is not set

Dll Functions
Constants

---

short int GetHiInsideHumAlarm_V ()
Description
Returns the hi inside humidity alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Returns the high inside humidity alarm
NOT_SET if alarm is not set

Dll Functions
Constants

---

short int GetLowOutsideHumAlarm_V ()
Description
Returns the low outside humidity alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Returns the low outside humidity alarm
NOT_SET if alarm is not set

Dll Functions
Constants

---

short int GetHiOutsideHumAlarm_V ()
Description
Returns the hi outside humidity alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Returns the high outside humidity alarm
NOT_SET if alarm is not set

Dll Functions
Constants

---

float GetLowWindChillAlarm_V ()
Description
Returns the low wind chilll alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the low wind chill alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

float GetLowDewPtAlarm_V ()
Description
Returns the low dew point alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the low dew point alarm.
NOT_SET if alarm is not set

Dll Functions

Constants

---

float GetHiDewPtAlarm_V ()
Description
Returns the hi dew point alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the hi dew point alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

float GetHiSolarRadAlarm_V ()
Description
Returns the hi solar radiation alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the hi solar radiation alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

short int GetHiWindSpeedAlarm_V ()
Description
Returns the hi wind speed alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the high wind speed alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

float GetHi10MinWindSpeedAlarm_V ()
Description
Returns the hi 10 minute wind speed alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the hi 10 minute wind speed alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

float GetHiHeatIndexAlarm_V ()
Description
Returns the hi heat index alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the hi heat index alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

float GetHiTHSWAlarm_V ()
Description
Returns the hi THSW alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the hi THSW alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

float GetHiRainRateAlarm_V ()
Description
Returns the hi rain rate alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the hi rain rate alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

float GetHiDailyRainAlarm_V ()
Description
Returns the hi dialy rain alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the hi dialy rain alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

float GetHiRainStormAlarm_V ()
Description
Returns the hi rain storm alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the hi rain storm alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

float GetFlashFloodAlarm_V ()
Description
Returns the flood flash alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the flood flash alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

float GetHiUVAlarm_V ()
Description
Returns the hi UV alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the hi UV alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

float GetHiUVMedAlarm_V ()
Description
Returns the hi UV Med alarm value from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Return Values
Gets the hi UV Med alarm.
NOT_SET if alarm is not set

Dll Functions
Constants

---

float GetLowExtraTempAlarm_V (short int sensorNumber)
Description
Returns the low alarm value for the given extra temperature sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Parameter
sensorNumber - The number of the extra temperature sensor. Valid values are 2 - 8.
Return Values
Returns the specified alarm threshold value in the current DLL units, or
BAD_DATA if alarm is not set

Dll Functions
Constants

---

float GetHiExtraTempAlarm_V (short int sensorNumber)
Description
Returns the high alarm value for the given extra temperature sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Parameter
sensorNumber - The number of the extra temperature sensor. Valid values are 2 - 8.
Return Values
Returns the specified alarm threshold value in the current DLL units, or
BAD_DATA if alarm is not set

Dll Functions
Constants

---

float GetLowExtraHumAlarm_V (short int sensorNumber)
Description
Returns the low alarm value for the given extra humidity sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Parameter
sensorNumber - The number of the extra temperature sensor. Valid values are 2 - 8.
Return Values
Returns the specified alarm threshold value in the current DLL units, or
BAD_DATA if alarm is not set

Dll Functions
Constants

---

float GetHiExtraHumAlarm_V (short int sensorNumber)
Description
Returns the high alarm value for the given extra humidity sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Parameter
sensorNumber - The number of the extra temperature sensor. Valid values are 2 - 8.
Return Values
Returns the specified alarm threshold value in the current DLL units, or
BAD_DATA if alarm is not set

Dll Functions
Constants

---

float GetLowSoilTempAlarm_V (short int sensorNumber)
Description
Returns the low alarm value for the given soil temperature sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Parameter
sensorNumber - The number of the soil temperature sensor. Valid values are 1 - 4.
Return Values
Returns the specified alarm threshold value in the current DLL units, or
BAD_DATA if alarm is not set

Dll Functions
Constants

---

float GetHiSoilTempAlarm_V (short int sensorNumber)
Description
Returns the high alarm value for the given soil temperature sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Parameter
sensorNumber - The number of the soil temperature sensor. Valid values are 1 - 4.
Return Values
Returns the specified alarm threshold value in the current DLL units, or
BAD_DATA if alarm is not set

Dll Functions
Constants

---

float GetLowSoilMoistureAlarm_V (short int sensorNumber)
Description
Returns the low alarm value for the given soil moisture sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Parameter
sensorNumber - The number of the soil moisture sensor. Valid values are 1 - 4.
Return Values
Returns the specified alarm threshold value in the current DLL units, or
BAD_DATA if alarm is not set

Dll Functions
Constants

---

float GetHiSoilMoistureAlarm_V (short int sensorNumber)
Description
Returns the high alarm value for the given soil moisture sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Parameter
sensorNumber - The number of the soil moisture sensor. Valid values are 1 - 4.
Return Values
Returns the specified alarm threshold value in the current DLL units, or
BAD_DATA if alarm is not set

Dll Functions
Constants

---

float GetLowLeafTempAlarm_V (short int sensorNumber)
Description
Returns the low alarm value for the given leaf temperature sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Parameter
sensorNumber - The number of the leaf temperature sensor. Valid values are 1 - 2.
Return Values
Returns the specified alarm threshold value in the current DLL units, or
BAD_DATA if alarm is not set

Dll Functions
Constants

---

float GetHiLeafTempAlarm_V (short int sensorNumber)
Description
Returns the high alarm value for the given leaf temperature sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Parameter
sensorNumber - The number of the leaf temperature sensor. Valid values are 1 - 2.
Return Values
Returns the specified alarm threshold value in the current DLL units, or
BAD_DATA if alarm is not set

Dll Functions
Constants

---

float GetLowLeafWetAlarm_V (short int sensorNumber)
Description
Returns the low alarm value for the given leaf wetness sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Parameter
sensorNumber - The number of the leaf wetness sensor. Valid values are 1 - 2.
Return Values
Returns the specified alarm threshold value in the current DLL units, or
BAD_DATA if alarm is not set

Dll Functions
Constants

---

float GetHiLeafWetAlarm_V (short int sensorNumber)
Description
Returns the high alarm value for the given leaf wetness sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Parameter

sensorNumber - The number of the leaf wetness sensor. Valid values are 1 - 2.
Return Values
Returns the specified alarm threshold value in the current DLL units, or
BAD_DATA if alarm is not set

Dll Functions
Constants

---

short int PutBarRiseAlarm_V (float barRiseAlarm)
Parameter
It should be in the range of 0.00 to 0.25
Description
Sets the barometer rise alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutBarFallAlarm_V (float barFallAlarm)
Parameter
It should be in the range of 0.00 to 0.25
Description
Sets the barometer fall alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutTimeAlarm_V (char* timeAlarm)
Parameter
Pass the parameter for this function in time format, e.g., "5:30a", "1:25p".
Description
Sets the time alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutInsideLowTempAlarm_V (float lowtempAlarm)
Parameter
It should be in the range of -90F to 164F
Description
Sets the low inside temperature alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutInsideHiTempAlarm_V (float hitempAlarm)
Parameter
It should be in the range of -90F to 164F
Description
Sets the high inside temperature alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutOutsideLowTempAlarm_V (float lowtempAlarm)
Parameter
It should be in the range of -90F to 164F
Description
Sets the low outside temperature alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

float PutOutsideHiTempAlarm_V (float hitempAlarm)
Parameter
It should be in the range of -90F to 164F
Description
Sets the high outside temperature alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutLowInsideHumAlarm_V (short int lowInsideAlarm)
Parameter
It should be in the range of 0 to 100
Description
Sets the low inside humidity alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutHiInsideHumAlarm_V (short int hiInsideAlarm)

Parameter
It should be in the range of 0 to 100
Description
Sets the high inside humidity alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutLowOutsideHumAlarm_V (short int lowOutsideAlarm)
Parameter
It should be in the range of 0 to 100
Description
Sets the low outside humidity alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutHiOutsideHumAlarm_V (short int hiOutsideAlarm)
Parameter
It should be in the range of 0 to 100
Description
Sets the high outside humidity alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutLowWindChillAlarm_V (float lowWindChillAlarm)
Parameters
It should be in the range of -120F to 134F
Description
Sets the low wind chill alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutLowDewPtAlarm_V (int lowDewPoint)
Parameter
It should be in the range of -120F to 134F
Description
Sets the low dew point alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.

Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutHiDewPtAlarm_V (int hiDewPoint)
Parameter
It should be in the range of -120F to 134F
Description
Sets the hi dew point alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutHiSolarRadAlarm_V (short hiAlarm)
Parameter
It should be in the range of 0 to 1800
Description
Sets the hi solar radiation alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutHiWindSpeedAlarm_V (float hiAlarm)
Parameter
It should be in the range of 0 to 254mph
Description
Sets the high wind speed alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutHi10MinWindSpeedAlarm_V (float hiAlarm)
Parameter
It should be in the range of 0 to 254mph
Description
Sets the high 10 minute average high wind speed alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions

Constants

---

short int PutHiHeatIndexAlarm_V (float heatAlarm)
Parameters
It should be in the range of -120F to 134F
Description
Sets the hi heat index alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutHiTHSWAlarm_V (float hiAlarm)
Parameter
It should be in the range of -90F to 164F
Description
Sets the hi THSW alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutHiRainFloodAlarm_V (float hiAlarm)
Parameter
It should be in the range of 0.00in to 327.66in
Description
Sets the hi rain flood alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutRainPerDayAlarm_V (float hiAlarm)
Parameter
It should be in the range of 0.00in to 327.66in
Description
Sets the rain per day alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutRainStormAlarm_V (float hiAlarm)
Parameter

It should be in the range of 0.00in to 327.66in
Description
Sets the rain storm alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutRainRateAlarm_V (float hiAlarm)
Parameter
It should be in the range of 0.00in/hr to 655.34in/hr
Description
Sets the rain rate alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutHiUVAlarm_V (float hiAlarm)
Parameter
It should be in the range of 0 to 16
Description
Sets the hi UV alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutHiUVMedAlarm_V (float hiAlarm)
Parameter
It should be in the range of 0 to 16
Description
Sets the hi UV Med alarm variable in the DLL struct. Invalid data clears the alarm variable
After setting all the alarm variables in the DLL struct, you need to call SetVantageAlarms_V function to set in Vantage.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutLowExtraTempAlarm_V (short int sensorNumber, float lowAlarm)
Description
Sets the low alarm value for the given extra temperature sensor in the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the extra temperature sensor. Valid values are 2 - 8.

lowAlarm - The new alarm threshold value to set.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutHiExtraTempAlarm_V (short int sensorNumber, float hiAlarm)
Description
Sets the high alarm value for the given extra temperature sensor in the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the extra temperature sensor. Valid values are 2 - 8.
hiAlarm - The new alarm threshold value to set.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutLowExtraHumAlarm_V (short int sensorNumber, float lowAlarm)
Description
Sets the low alarm value for the given extra humidity sensor in the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the extra temperature sensor. Valid values are 2 - 8.
lowAlarm - The new alarm threshold value to set.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutHiExtraHumAlarm_V (short int sensorNumber, float hiAlarm)
Description
Sets the high alarm value for the given extra humidity sensor in the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the extra temperature sensor. Valid values are 2 - 8.
hiAlarm - The new alarm threshold value to set.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutLowSoilTempAlarm_V (short int sensorNumber, float lowAlarm)
Description
Sets the low alarm value for the given soil temperature sensor in the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.

Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the soil temperature sensor. Valid values are 1 - 4.
lowAlarm - The new alarm threshold value to set.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutHiSoilTempAlarm_V (short int sensorNumber, float hiAlarm)
Description
Sets the high alarm value for the given soil temperature sensor in the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the soil temperature sensor. Valid values are 1 - 4.
hiAlarm - The new alarm threshold value to set.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutLowSoilMoistureAlarm_V (short int sensorNumber, float lowAlarm)
Description
Sets the low alarm value for the given soil moisture sensor in the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the soil moisture sensor. Valid values are 1 - 4.
lowAlarm - The new alarm threshold value to set.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutHiSoilMoistureAlarm_V (short int sensorNumber, float hiAlarm)
Description
Sets the high alarm value for the given soil moisture sensor in the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the soil moisture sensor. Valid values are 1 - 4.
hiAlarm - The new alarm threshold value to set.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutLowLeafTempAlarm_V (short int sensorNumber, float lowAlarm)

Description
Sets the low alarm value for the given leaf temperature sensor in the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the leaf temperature sensor. Valid values are 1 - 2.
lowAlarm - The new alarm threshold value to set.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutHiLeafTempAlarm_V (short int sensorNumber, float hiAlarm)
Description
Sets the high alarm value for the given leaf temperature sensor in the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the leaf temperature sensor. Valid values are 1 - 2.
hiAlarm - The new alarm threshold value to set.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutLowLeafWetAlarm_V (short int sensorNumber, float lowAlarm)
Description
Sets the low alarm value for the given leaf wetness sensor in the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the leaf wetness sensor. Valid values are 1 - 2.
lowAlarm - The new alarm threshold value to set.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int PutHiLeafWetAlarm_V (short int sensorNumber, float hiAlarm)
Description
Sets the high alarm value for the given leaf wetness sensor in the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the leaf wetness sensor. Valid values are 1 - 2.
hiAlarm - The new alarm threshold value to set.
Return Values
-1 if invalid data
0 if successful

Dll Functions
Constants

---

short int ClearBarRiseAlarm_V ()
Description
This function clears the barometer rise alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearBarFallAlarm_V ()
Description
This function clears the barometer fall alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearTimeAlarm_V ()
Description
This function clears the time alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearInsideLowTempAlarm_V ()
Description
This function clears the low inside temperature alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearInsideHiTempAlarm_V ()
Description
This function clears the hi inside temperature alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearOutsideLowTempAlarm_V ()
Description
This function clears the low outside temperature alarm in the weather station
Return Values

0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearOutsideHiTempAlarm_V ()
Description
This function clears the hi outside temperature alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearLowInsideHumAlarm_V ()
Description
This function clears the low inside humidity alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearHiInsideHumAlarm_V ()
Description
This function clears the hi inside humidity alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearLowOutsideHumAlarm_V ()
Description
This function clears the low outside humidity alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearHiOutsideHumAlarm_V ()
Description
This function clears the hi outside humidity alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearLowWindChillAlarm_V ()
Description
This function clears the low wind chill alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearLowDewPtAlarm_V ()
Description
This function clears the low dew point alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearHiDewPtAlarm_V ()
Description
This function clears the hi dew point alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearHiSolarRadAlarm_V ()
Description
This function clears the hi solar radiation alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearHiWindSpeedAlarm_V ()
Description
This function clears the hi wind speed alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearHi10MinWindSpeedAlarm_V ()
Description
This function clears the hi 10 minute wind speed alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearHiHeatIndexAlarm_V ()
Description
This function clears the hi heat index alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearHiTHSWAlarm_V ()
Description
This function clears the hi THSW alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearHiRainFloodAlarm_V ()
Description
This function clears the hirain flood alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearHiRainPerDayAlarm_V ()
Description
This function clears the hi rain per day alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearRainStormAlarm_V ()
Description
This function clears the rain storm alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearRainRateAlarm_V ()

Description
This function clears the rain rate alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearHiUVAlarm_V ()
Description
This function clears the UV alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

short int ClearHiUVMedAlarm_V ()
Description
This function clears the UV Med alarm in the weather station
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

float ClearLowExtraTempAlarm_V (short int sensorNumber)
Description
This function clears the low alarm value for the given extra temperature sensor in the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the extra temperature sensor. Valid values are 2 - 8.
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

float ClearHiExtraTempAlarm_V (short int sensorNumber)
Description
This function clears the high alarm value for the given extra temperature sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the extra temperature sensor. Valid values are 2 - 8.
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

float ClearLowExtraHumAlarm_V (short int sensorNumber)
Description
This function clears the low alarm value for the given extra humidity sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the extra temperature sensor. Valid values are 2 - 8.
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

float ClearHiExtraHumAlarm_V (short int sensorNumber)
Description
This function clears the high alarm value for the given extra humidity sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the extra temperature sensor. Valid values are 2 - 8.
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

float ClearLowSoilTempAlarm_V (short int sensorNumber)
Description
This function clears the low alarm value for the given soil temperature sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the soil temperature sensor. Valid values are 1 - 4.
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

float ClearHiSoilTempAlarm_V (short int sensorNumber)
Description
This function clears the high alarm value for the given soil temperature sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the soil temperature sensor. Valid values are 1 - 4.
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

float ClearLowSoilMoistureAlarm_V (short int sensorNumber)
Description
This function clears the low alarm value for the given soil moisture sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the soil moisture sensor. Valid values are 1 - 4.
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

float ClearHiSoilMoistureAlarm_V (short int sensorNumber)
Description
This function clears the high alarm value for the given soil moisture sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the soil moisture sensor. Valid values are 1 - 4.
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

float ClearLowLeafTempAlarm_V (short int sensorNumber)
Description
This function clears the low alarm value for the given leaf temperature sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the leaf temperature sensor. Valid values are 1 - 2.
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

float ClearHiLeafTempAlarm_V (short int sensorNumber)
Description
This function clears the high alarm value for the given leaf temperature sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the leaf temperature sensor. Valid values are 1 - 2.
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

float ClearLowLeafWetAlarm_V (short int sensorNumber)

Description
This function clears the low alarm value for the given leaf wetness sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the leaf wetness sensor. Valid values are 1 - 2.
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

---

float ClearHiLeafWetAlarm_V (short int sensorNumber)
Description
This function clears the high alarm value for the given leaf wetness sensor from the DLL alarm structure.
Call LoadVantageAlarms_V fuction before calling this function.
Call SetVantageAlarms_V fuction to write the new value into the weather station.
Parameter
sensorNumber - The number of the leaf wetness sensor. Valid values are 1 - 2.
Return Values
0 if successful
COM_ERROR if error

Dll Functions
Constants

High Low Functions

The following functions are used to manipulate the High/Low data in Vantage.

Similar to the Current Data functions above, LoadVantageHiLows_V reads in all of the High and Low data values stored on the Vantage console into a cache in the DLL. The GetHi___ and GetLow___ functions return the selected high or low value out of the cache.

Also like the Current Data functions, there are both individual functions for specific values (such as GetHiOutsideTemp which returns the high outside temperature since midnight) and general functions that will retrieve the value selected by its input parameter (GetHiLowDataByID_V). The individual functions are primerily for backward compatibility with previous DLL versions. The Get___ByID_V functions are the prefered method for accessing high/low data. See the VantgePro Programmer's Reference or the Vantage User Guide for more information on what high and low values are available on the Vantage console.

To retrieve the time that a daily high or low occured, you can either use an "individual" function (such as GetHiLowTimesDewPt_V), or the general function (GetHiLowTimeByID_V). The individual function will retrieve the time of both the high and low for the specified weathe data. GetHiLowTimeByID_V will retrieve the time of the selected high/low value using the same ID as GetHiLowDataByID_V. (But only Daily high and low value will return valid time data.)

For example:
LoadVantageHiLows_V( ) // loads all the Hi/Lows from Vantage
GetHiInsideTemp_V ( )
GetLowInsideTemp_V ( )
GetHiLowTimesInTemp_V( )
GetHiOutsideTemp_V ( )
GetLowOutsideTemp_V ( )
GetHiLowTimesOutTemp_V( )

In the above sequence of steps, all the vantage Hi/Low values are loaded into a local cache (a structure) in step 1, and in the subsequent steps the values are returned from the cache..

Example using ___ByID_V functions:

LoadVantageHiLows_V( ) // loads all the Hi/Lows from Vantage
GetHiLowDataByID_V (DAY_HI_OUT_DEW_WDID)
GetHiLowTimeByID_V (DAY_HI_OUT_DEW_WDID)
GetHiLowDataByID_V (MONTH_HI_OUT_DEW_WDID)
GetHiLowDataByID_V (YEAR_HI_OUT_DEW_WDID)

In this example we retrieve the value and time of the daily high dew point and the value of the month and year high dew point.

Dll Functions
Constants

---

int LoadVantageHiLows_V ()
Description
This function reads Vantage Hi/Low values from the device and fills the DLL structure with those values.
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

float GetHiOutsideTemp_V()
Description
Returns the hi outside temperature value from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Return Values
Gets the high outside temperature.
BAD_DATA if data is not available

Dll Functions
Constants

---

float GetLowOutsideTemp_V()
Description
Returns the low outside temperature value from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Return Values
Gets the low outside temperature.
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetHiLowTimesOutTemp_V (DateTimeStruct* DateTimeHiOutTemp,DateTimeStruct* DateTimeLowOutTemp)
Description
Returns the Hi/Low outside temperature times from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Fills the DateTimeStruct structure with the high and low times.
Return Values
BAD_DATA if data is not available
0 if successful

Dll Functions
Constants

---

float GetHiInsideTemp_V()

Description
Returns the hi inside temperature value from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Return Values
Gets the high inside temperature.
BAD_DATA if data is not available

Dll Functions
Constants

---

float GetLowInsideTemp_V()
Description
Returns the low inside temperature value from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Return Values
Gets the low inside temperature.
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetHiLowTimesInTemp_V (DateTimeStruct* DateTimeHiInTemp,DateTimeStruct* DateTimeLowInTemp)
Description
Returns the Hi/Low inside temperature times from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Fills the DateTimeStruct structure with the high and low times.
Return Values
BAD_DATA if data is not available
0 if successful

Dll Functions
Constants

---

short int GetHiOutsideHum_V()
Description
Returns the hi outside humidity value from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Return Values
Gets the high outside humidity.
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetLowOutsideHum_V()
Description
Returns the low outside humidity value from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Return Values
Gets the low outside humidity.
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetHiLowTimesOutHum_V (DateTimeStruct* DateTimeHiOutHum,DateTimeStruct* DateTimeLowOutHum)

Description
Returns the Hi/Low outside humidity times from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Fills the DateTimeStruct structure with the high and low times.
Return Values
BAD_DATA if data is not available
0 if successful

Dll Functions
Constants

---

short int GetHiInsideHum_V()
Description
Returns the hi inside humidity value from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Return Values
Gets the high inside humidity.
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetLowInsideHum_V()
Description
Returns the low inside humidity value from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Return Values
Gets the low inside humidity.
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetHiLowTimesInHum_V (DateTimeStruct* DateTimeHiInHum,DateTimeStruct* DateTimeLowInHum)
Description
Returns the Hi/Low inside humidity times from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Fills the DateTimeStruct structure with the high and low times.
Return Values
BAD_DATA if data is not available
0 if successful

Dll Functions
Constants

---

float GetHiDewPt_V()
Description
Returns the hi dew point value from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Return Values
Gets the high dew point value.
BAD_DATA if data is not available

Dll Functions
Constants

---

float GetLowDewPt_V()
Description
Returns the low dew point value from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Return Values
Gets the low dew point value.
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetHiLowTimesDewPt_V (DateTimeStruct* DateTimeHiDewPt,DateTimeStruct* DateTimeLowDewPt)
Description
Returns the Hi/Low dew point times from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Fills the DateTimeStruct structure with the high and low times.
Return Values
BAD_DATA if data is not available
0 if successful

Dll Functions
Constants

---

float GetLowWindChill_V()
Description
Returns the low wind chill value from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Return Values
Gets the low wind chill value.
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetLowTimesWindChill_V (DateTimeStruct* DateTimeLowWindChill)
Description
Returns the low wind chill time from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Fills the DateTimeStruct structure with the time for low wind chill.
Return Values
BAD_DATA if data is not available
0 if successful

Dll Functions
Constants

---

float GetHiWindSpeed_V()
Description
Returns the Hi wind speed value from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Return Values
Gets the high wind speed.
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetHiTimesWindSpeed_V (DateTimeStruct* DateTimeHiWindSpeed)
Description
Returns the Hi wind speed time from the DLL alarm structure.
Call LoadVantageHiLows_V function before calling this function.
Fills the DateTimeStruct structure with the time for hi wind speed.
Return Values
BAD_DATA if data is not available
0 if successful

Dll Functions
Constants

---

float GetHiLowDataByID_V(long int weatherDataID)
Description
Call LoadVantageHiLows_V function before calling this function.
Return Values
Gets the value of the selected high/low parameter
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetHiLowTimeByID_V(long int weatherDataID, DateTimeStamp * dateTimeValue)
Description
Call LoadVantageHiLows_V function before calling this function.
Only Day high and low values have times accociated with them. If you try to get the time of a Month or Year High/Low value you will get an error.
Return Values
BAD_DATA if data is not available
0 if successful

---

float GetHiLowDataStrByID_V(long int weatherDataID, char *s, short int bufferLength)
Description
Call LoadVantageHiLows_V function before calling this function.
Return Values
Gets the text of the value of the selected high/low parameter and put it in the the “s” parameter
BAD_DATA if data is not available

Dll Functions
Constants

---

short int GetHiLowTimeStrByID_V(long int weatherDataID, char *s, short int bufferLength)
Description
Call LoadVantageHiLows_V function before calling this function.
Get the text of the time associated with the value and put it in the the “s” parameter. Only Day high and low values have times associated with them. If you try to get the time of a Month or Year High/Low value you will get an error.
Return Values
BAD_DATA if data is not available
0 if successful

Dll Functions
Constants

Calibrate Functions

These functions are used to get and set the calibration values for various Vantage weather data parameters.

LoadVantageCalibration_V(CurrentVantageCalibration&) **downloads all the current Calibration values from Vantage and copies into CurrentVantageCalibration structure.**

After calibrating any Vantage variable , you need to call SetVantageCalibration_V to calibrate. SetVantageCalibration_V is the one which calibrates the Vantage variables.

Here's an sample scenario:
PutOutsideTempCalibrationValue_V(..);
PutInsideTempCalibrationValue_V(..);
PutOutsideHumCalibrationValue_V(..);
PutInsideHumCalibrationValue_V(..);
SetVantageCalibration_V ( ) //this calibrates all the above variables in the Vantage

In the above sequence of steps, all the vantage calibration values are stored in a local cache (a structure) in steps 1,2,3,4, and in the last step the values are calibrated in Vantage. This way you only need to communicate with Vantage only once.

Dll Functions
Constants

---

int LoadVantageCalibration_V(CurrentVantageCalibration)
Description
Downloads all the current Calibration values from Vantage and copies into CurrentVantageCalibration structure.
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int PutOutsideTempCalibrationValue_V(float calValue)
Description
Sets the outside temperature variable in the DLL struct.
You need to call SetVantageCalibration_V function to calibrate into Vantage.
Puts the calValue into CurrentVantageCalibrationstruct
Parameters
Valid values are -12.8 to 12.7 degrees from outside temperature raw value.
e.g., if
CurrentVantageCalibration.tempOut equals 76 F, and
CurrentVantageCalibration.tempOutOffset equals .5,
then raw outside temperature is caliculated as
rawOutsideTemp = 75.5 (CurrentVantageCalibration.tempOut - CurrentVantageCalibration.tempOutOffset);
If you enter calValue as 90 (which is 90-75.5=14.5 (>12.7)) its invalid
a value of say 85 (85-75.5 = 9.5(with in -12.8 to +12.7 range)) is valid.
Return Values
0 if successful
-1 if invalid data
COM_ERROR if error in communicating with Vantage

Dll Functions
Constants

---

int PutOutsideTempCalibrationValueEx_V(int sensorNumber, float calValue)
Description
Sets the extra outside temperature variables in the DLL struct.

You need to call SetVantageCalibration_V function to calibrate into Vantage.
Puts the calValue into CurrentVantageCalibrationstruct
Parameters
Valid values are -12.8 to 12.7 degrees from outside temperature raw value.
e.g., if
CurrentVantageCalibration.tempOut equals 76 F, and
CurrentVantageCalibration.tempOutOffset equals .5,
then raw outside temperature is caliculated as
rawOutsideTemp = 75.5 (CurrentVantageCalibration.tempOut - CurrentVantageCalibration.tempOutOffset);
If you enter calValue as 90 (which is 90-75.5=14.5 (>12.7)) its invalid
a value of say 85 (85-75.5 = 9.5(with in -12.8 to +12.7 range)) is valid.
Return Values
0 if successful
-1 if invalid data
COM_ERROR if error in communicating with Vantage

Dll Functions
Constants

---

int PutOutsideTempCalibrationOffset_V(float calValue)
Description
Sets the outside temperature offset value in the DLL struct.
You need to call SetVantageCalibration_V function to calibrate into Vantage.
Parameters
Valid values are -12.8 to 12.7 degrees.
e.g., if
CurrentVantageCalibration.tempOut equals 76 F, and
CurrentVantageCalibration.tempOutOffset equals .5,
then raw outside temperature is caliculated as
rawOutsideTemp = 75.5 (CurrentVantageCalibration.tempOut - CurrentVantageCalibration.tempOutOffset);
If you enter calValue as 3, the value that is displayed on the Vantage console is:
75.5 + 3 = 78.5 ;rawValue + offset)
Return Values
0 if successful
-1 if invalid data
COM_ERROR if error in communicating with Vantage

Dll Functions
Constants

---

int PutOutsideTempCalibrationOffsetEx_V(int sensorNumber, float calValue)
Description
Sets the extra outside temperature offset values in the DLL struct.
You need to call SetVantageCalibration_V function to calibrate into Vantage.
Parameters
Valid values are -12.8 to 12.7 degrees.
e.g., if
CurrentVantageCalibration.tempOut equals 76 F, and
CurrentVantageCalibration.tempOutOffset equals .5,
then raw outside temperature is caliculated as
rawOutsideTemp = 75.5 (CurrentVantageCalibration.tempOut - CurrentVantageCalibration.tempOutOffset);
If you enter calValue as 3, the value that is displayed on the Vantage console is:
75.5 + 3 = 78.5 ;rawValue + offset)
Return Values
0 if successful
-1 if invalid data
COM_ERROR if error in communicating with Vantage

Dll Functions

Constants

---

int PutInsideTempCalibrationValue_V(float calValue)
Description
Sets the inside temperature variable in the DLL struct.
You need to call SetVantageCalibration_V function to calibrate into Vantage.
Puts the calValue into CurrentVantageCalibrationstruct
Parameters
Valid values are -12.8 to 12.7 degrees from inside temperature raw value.
e.g., if
CurrentVantageCalibration.tempIn equals 70 F, and
CurrentVantageCalibration.tempInOffset equals .5,
then raw outside temperature is caliculated as
rawInsideTemp = 69.5 (CurrentVantageCalibration.tempIn - CurrentVantageCalibration.tempInOffset);
If you enter calValue as 85 (which is 85-69.5=15.5 (>12.7)) its invalid
a value of say 75 (75 - 69.5 = 5.5(with in -12.8 to +12.7 range)) is valid.
Return Values
0 if successful
-1 if invalid data
COM_ERROR if error in communicating with Vantage

Dll Functions
Constants

---

int PutInsideTempCalibrationOffset_V(float calValue)
Description
Sets the inside temperature offset value in the DLL struct.
You need to call SetVantageCalibration_V function to calibrate into Vantage.
Parameters
Valid values are -12.8 to 12.7 degrees.
e.g., if
CurrentVantageCalibration.tempIn equals 76 F, and
CurrentVantageCalibration.tempInOffset equals .5,
then raw inside temperature is caliculated as
rawInsideTemp = 75.5 (CurrentVantageCalibration.tempIn - CurrentVantageCalibration.tempInOffset);
If you enter calValue as 3, the value that is displayed on the Vantage console is:
75.5 + 3 = 78.5 ; rawValue + offset)
Return Values
0 if successful
-1 if invalid data
COM_ERROR if error in communicating with Vantage

Dll Functions
Constants

---

int PutOutsideHumCalibrationValue_V(float calValue)
Description
Sets the outside humidity variable in the DLL struct.
You need to call SetVantageCalibration_V function to calibrate into Vantage.
Puts the calValue into CurrentVantageCalibrationstruct
Parameters
Valid values are 0 to 100.
Return Values
0 if successful
-1 if invalid data
COM_ERROR if error in communicating with Vantage

Dll Functions

Constants

---

int PutOutsideHumCalibrationValueEx_V(int sensorNumber, float calValue)
Description
Sets the extra outside humidity variables in the DLL struct.
You need to call SetVantageCalibration_V function to calibrate into Vantage.
Puts the calValue into CurrentVantageCalibrationstruct
Parameters
Valid values are 0 to 100.
Return Values
0 if successful
-1 if invalid data
COM_ERROR if error in communicating with Vantage

Dll Functions
Constants

---

int PutOutsideHumCalibrationOffset_V(float calValue)
Description
Sets the outside humidity offset in the DLL struct.
You need to call SetVantageCalibration_V function to calibrate into Vantage.
Puts the calValue into CurrentVantageCalibrationstruct
Parameters
calValue(offset) + rawValue should be within the range 0 to 100.
For Eg.,
CurrentVantageCalibration.humOut equals 60 F, and
CurrentVantageCalibration.humOutOffset equals 2,
then raw outside humidity is caliculated as
rawOutsideHum = 58 (CurrentVantageCalibration.humOut - CurrentVantageCalibration.humOutOffset);
If you enter calValue as 3, the value that is displayed on the Vantage console is:
58 + 3 = 61 (rawValue + offset with in 0 to 100)
Return Values
0 if successful
-1 if invalid data
COM_ERROR if error in communicating with Vantage

Dll Functions
Constants

---

int PutOutsideHumCalibrationOffsetEx_V(int sensorNumber,float calValue)
Description
Sets the extra outside humidity offsets in the DLL struct.
You need to call SetVantageCalibration_V function to calibrate into Vantage.
Puts the calValue into CurrentVantageCalibrationstruct
Parameters
calValue(offset) + rawValue should be within the range 0 to 100.
For Eg.,
CurrentVantageCalibration.humOut equals 60 F, and
CurrentVantageCalibration.humOutOffset equals 2,
then raw outside humidity is caliculated as
rawOutsideHum = 58 (CurrentVantageCalibration.humOut - CurrentVantageCalibration.humOutOffset);
If you enter calValue as 3, the value that is displayed on the Vantage console is:
58 + 3 = 61 (rawValue + offset with in 0 to 100)
Return Values
0 if successful
-1 if invalid data
COM_ERROR if error in communicating with Vantage

Dll Functions
Constants

---

int PutInsideHumCalibrationValue_V(float calValue)
Description
Sets the inside humidity variable in the DLL struct.
You need to call SetVantageCalibration_V function to calibrate into Vantage.
Puts the calValue into CurrentVantageCalibrationstruct
Parameters
Valid values are 0 to 100.
Return Values
0 if successful
-1 if invalid data
COM_ERROR if error in communicating with Vantage

Dll Functions
Constants

---

int PutInsideHumCalibrationOffset_V(float calValue)
Description
Sets the inside humidity offset in the DLL struct.
You need to call SetVantageCalibration_V function to calibrate into Vantage.
Puts the calValue into CurrentVantageCalibrationstruct
Parameters
calValue(offset) + rawValue should be within the range 0 to 100.
For Eg.,
CurrentVantageCalibration.humIn equals 60 F, and
CurrentVantageCalibration.humInOffset equals 2,
then raw inside humidity is caliculated as
rawInsideHum = 58 (CurrentVantageCalibration.humIn - CurrentVantageCalibration.humInOffset);
If you enter calValue as 3, the value that is displayed on the Vantage console is:
58 + 3 = 61 (rawValue + offset with in 0 to 100)
Return Values
0 if successful
-1 if invalid data
COM_ERROR if error in communicating with Vantage

Dll Functions
Constants

---

int SetVantageCalibration_V()
Description
Sets the calibration data from the union VantageCalData into the vantage
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int GetWindDirCalibrationOffset_V(short &windDirCal)
Description
Returns the wind direction calibration offset stored in the Vantage Console
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int PutWindDirCalibrationOffset_V(short windDirCal)
Description
Sets the wind calibration offset. Valid range of values (-360 to +360)
Return Values
COM_OPEN_ERROR if comm port not open
COM_ERROR if error
PARAMETER_ERROR if windDirCal is an invalid number.
0 if successful

Dll Functions
Constants

Download Functions

The Following functions allow you to download and manipulate the archive records stored in the VantagePro datalogger.

Use the function DownloadData_V to download the archive records from a given time point (or the whole archive memory) into a local cache in the DLL.

Use the function DownloadWeData_V to download the archive records from a given time point (or the whole archive memory) into a local cache in the DLL from the Davis Instrument WeatherServer.

Use the function GetMemoryArchiveRecordCount_V to determine how many blocks of new data are available for download from the VantagePro station. Each block holds 5 data records, but both the first and last blocks usually also contain old data records as well as new ones.

Use the function GetMemoryArchiveCountAfterDate_V to return the number of download blocks available after the timestamp.

Use the function GetNumberOfArchiveRecords_V after downloading to determine the number of new archive records that were transfered by DownloadData_V.

Use either GetArchiveRecord_V or GetArchiveRecordEx_V to extract the data from a particular archive record and copy it into a data structure. GetArchiveRecord_V uses the same WeatherRecordStruct used in previous DLL versions. GetArchiveRecordEx_V uses the WeatherRecordStructEx structure with additional data fields.

For example:
```
DateTimeStamp dateTimeStamp;
WeatherRecord record;
WeatherRecordEx recordEx;

// Add code to initialize the time stamp to download after a particular time
DownloadData_V( timestamp ); // download the archive records
int n = GetNumberOfArchiveRecords_V( );

for (int i=0; i<n; i++)
  {
 &nbspGetArchiveRecord_V(&record, i);
 &nbspGetArchiveRecordEx_V(&recordEx, i);
  // Add code to do something with the record such as store it in a database etc.
  }
```

Dll Functions
Constants

---

int DownloadData_V(DateTimeStamp dateTimeStamp)
Description

Starts the download of the archive data, after the date-time passed. Returns the number of records transferred.
Parameter
dateTimeStamp contains the time stamp after which you are interested to download the records.
Return Values
COM_ERROR if an error occurs.
0 if no new records to download.
# of records transferred, if successful.

Dll Functions
Constants

---

int DownloadWebData_V(DateTimeStamp dateTimeStamp, char *userName, char *password)
Description
Starts the download of the archive data from WeatherLink.com. If the DateTimeStamp is found, only data after the date-time are archived. Otherwise, all records are archived. Returns the number of records transferred.
Parameter
dateTimeStamp contains the time stamp after which you are interested to download the records.
userName contains the user name of the WeatherLink account.
password contains the password of the WeatherLink account.
Return Values
COM_ERROR if an error occurs.
0 if no new records for that user to download.
# of records transferred, if successful.

Dll Functions
Constants

---

short int GetMemoryArchiveRecordCount_V()
Description
Returns the number of records in the archive memory.
Return Values
COM_ERROR if error

Dll Functions
Constants

---

short int GetMemoryArchiveCountAfterDate_V(DateTimeStamp *dateTimeStamp)
Description
Returns the number of records in the archive memory after the date timestamp
Return Values
COM_ERROR if error

Dll Functions
Constants

---

short int GetNumberOfArchiveRecords_V()
Description
Returns the number of records.
Return Values
COM_ERROR if error

Dll Functions
Constants

---

short int GetArchiveRecord_V(WeatherRecordStruct * newRecordStruct, short int i)
Description

Fills the WeatherRecordStruct structure with the data for the record number i in the data just downloaded from the archive. The records are numbered from 0 .This function must be used immediately after the DownloadData_V function. If another Dllfunction other than the ones in the download group is called after DownloadData_V then the Dll frees the memory used for storing the downloaded data.
Parameter
First parameter is reference to WeatherRecordStruct.
Second parameter is record number.
Return Values
returns 0 if successful

Dll Functions
Constants

---

short int GetArchiveRecordEx_V(WeatherRecordStructEx * newRecordStruct, short int i)
Description
Fills the WeatherRecordStructEx structure with the data for the record number i in the data just downloaded from the archive. The records are numbered from 0. This function must be used immediately after the DownloadData_V function. If another DLL function other than the ones in the download group is called after DownloadData_V then the Dll frees the memory used for storing the downloaded data.
Parameter
newRecordStruct - A pointer to the WeatherRecordStructEx.where the data will be stored
i - The record number.to read. Valid values are 0 to (GetNumberOfArchiveRecords_V -1)
Return Values
0 if successful

Dll Functions
Constants

Clear Functions

The following functions clear various values directly on the VantagePro console.

The ClearVantageLows_V function combines the functions of the ClearVantageDayLows_V, ClearVantageMonthLows_V, and ClearVantageYearLows_V functions. ClearVantageHighs_V likewise combines the operation of the ClearVantageDayHighs_V, ClearVantageMonthHighs_V, and ClearVantageYearHighs_V functions.

Note: In all of the clear high/low functions, the values for all weather data parameters are cleared. There is no way to clear an individual high or low value (such as Month High Outside Humidity).

Dll Functions
Constants

---

int ClearVantageLows_V()
Description
Clear the day, month, and year low values for all weather parameters on the vantage.
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int ClearVantageHighs_V()
Description

Clear the day, month, and year high values for all weather parameters on the vantage.
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int ClearVantageAlarms_V()
Description
Clear the alarm threshold values on the vantage.
Does not clear them in the internal data cache read in by LoadVantageAlarms_V.
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int ClearVantageCalNums_V()
Description
Clear the calibration numbers, except for the Barometer calibration settings.
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int ClearCurrentData_V()
Description
Clears the current data on the Vantage.
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int ClearStoredData_V()
Description
Clears the stored archived data on the Vantage.
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int ClearVantageDayLows_V()
Description
Clear the day low values for all weather parameters on the vantage.
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int ClearVantageDayHighs_V()
Description
Clear the day high values for all weather parameters on the vantage.
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int ClearVantageMonthLows_V()
Description
Clear the month low values for all weather parameters on the vantage.
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int ClearVantageMonthHighs_V()
Description
Clear the month high values for all weather parameters on the vantage.
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int ClearVantageYearLows_V()
Description
Clear the year low values for all weather parameters on the vantage.
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int ClearVantageYearHighs_V()
Description
Clear the year high values for all weather parameters on the vantage.
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int ClearVantageRainET_V(short int rainETValue)
Description

Clears the selected rain or ET data value..
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int ClearVantageGraphs_V()
Description
Clears all of the graph points on the Vantage console.
Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

---

int ClearVantageAlarmBits_V()
Description
Clears the status of all of the current alarms on the VantagePro console.

Note: This does not clear the alarm threshold values. If any thresholds are set so that they are activated by the current conditions, the alarm will sound as soon as the weather station receives the next data packet containing the appropriate data.

For example, if the high outside temperature alarm is set to 65F when it is currently 70F, then the alarm will sound as soon as the next temperature data is received (nominaly every 10 seconds).

Return Values
COM_ERROR if error
0 if successful

Dll Functions
Constants

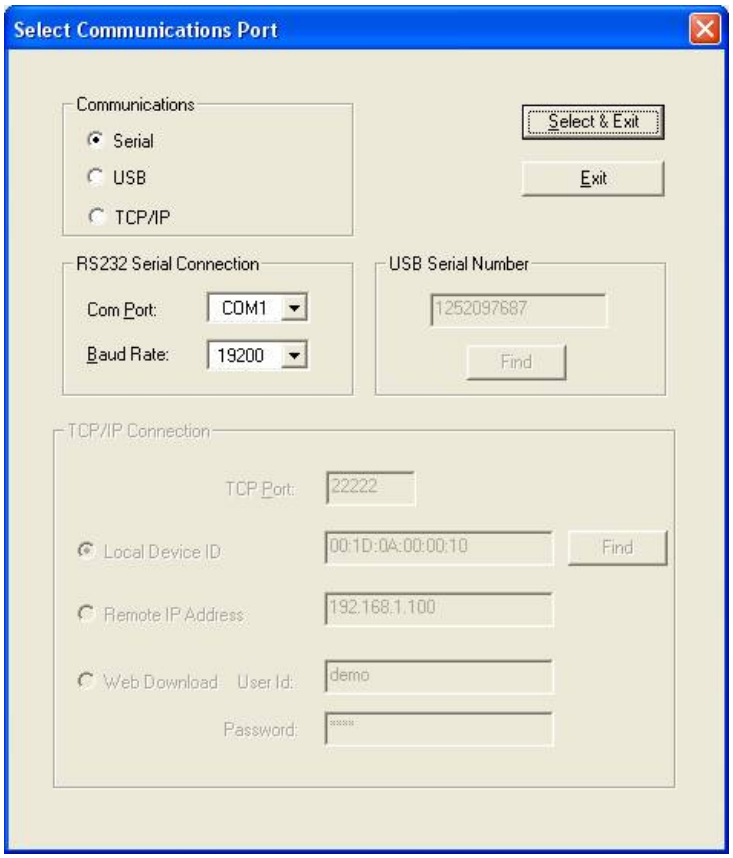
Appendixes: DLL Usage Examples

To illustrate how the VantagePro DLL could be used, two sample programs are included with the DLL 2.40/2.41 package. The original Visual C++ 6.0 example program has been updated to work with Visual Studio 2005 in order for to easily support WinINet (The Windows Internet application programming interface) the same way as our WeatherLink software does. The Visual Basic Interface has been added along with a new Visual Basic sample program. Please be aware that the example program is written with the focus illustrating the usage of VantageProDll. In a real application, we would handle things more gracefully -- gray out commands that are not valid for the current settings, and close the communication port when we are done using it.

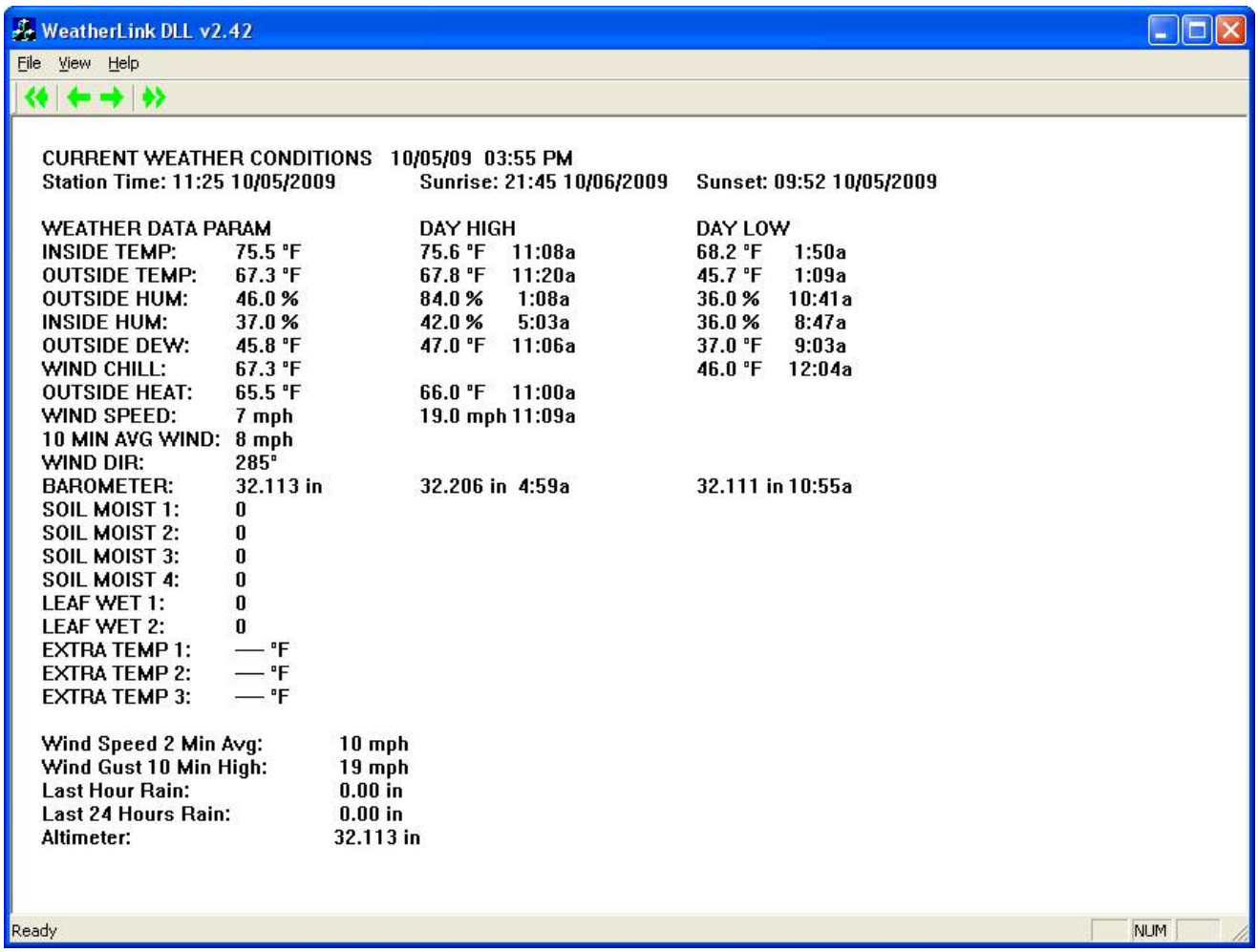
1. Visual C++ Example Program

As the DLL is being updated, the original Example Application is also being expanded to show usage of the support of USB WeatherLink device, TCPIP WeatherLink IP device and Web-Download. The communication data/selection is now stored in the “comm.dat” file instead of the original “example.dat” file.

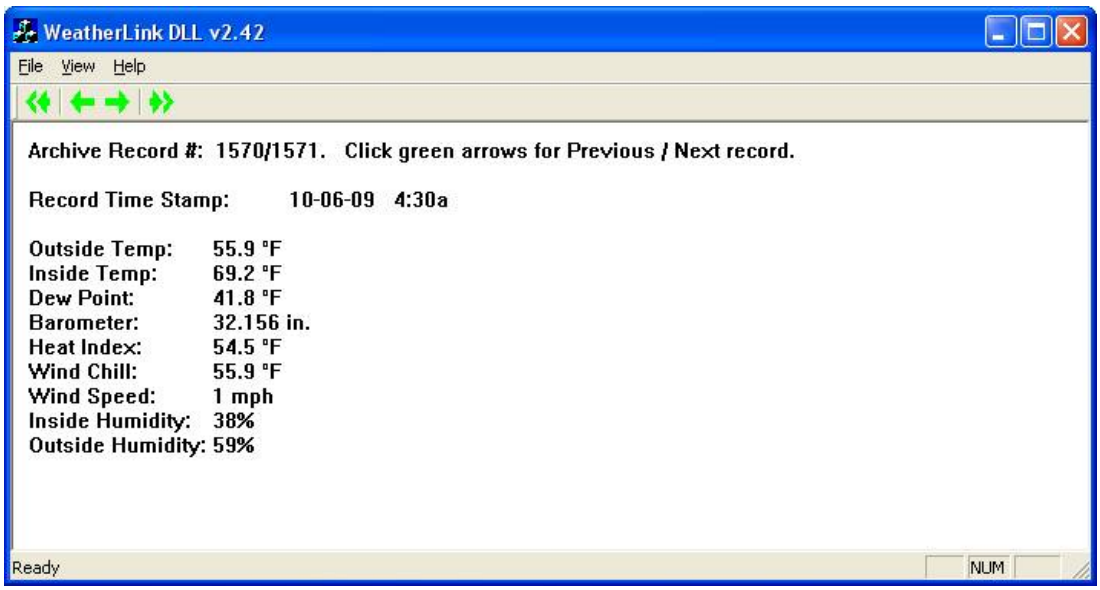
The Communications Port Setup screen let user select from various communication options:



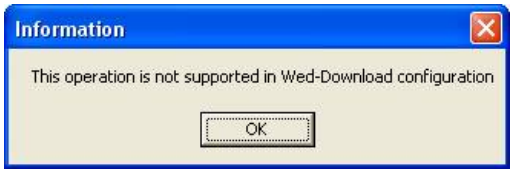
A Current Data screen.



A Download Result screen. Here we may actually use the Green Arrows to scroll to other records.



For Web-Download configuration, we will not be able to get firmware version, current data or Vantage Alarms. The following message shall appear if we attempt to get those data.



2. Visual Basic Support

The Visual Basic DLL Interface

W just added a Visual Studio 2008 VB Interface to our DLL version 2.4 and later. We also provide a VB Example program with Visual Studio 2008 VB. However, our commitment to Visual Basic support is limited to that -- Davis instrument is not committed to further user support on the Visual Basic. Users of earlier versions of Visual Basic may need to make modifications on their own, in accordance with VB DLL usage guideline for that versions.

Many Interface functions and structure have not been tested either. We put in some effort to provide accurate data type conversions, but we don't guarantee the accuracy. In case there are errors, please compare this VB DLL header file against the DLL Interface file for C/C++ and consult a proper VB book for the corresponding VB syntax for using C/C++ DLL. Also, please let us know if you find any errors so we can correct them and share with other users.

Important NOTE about data sizes:
"int" or "long int" refers to a 4 byte integer
"short int" refers to a 2 byte integer
"float" refers to a 4 byte floating point
"char" refers to a single byte.

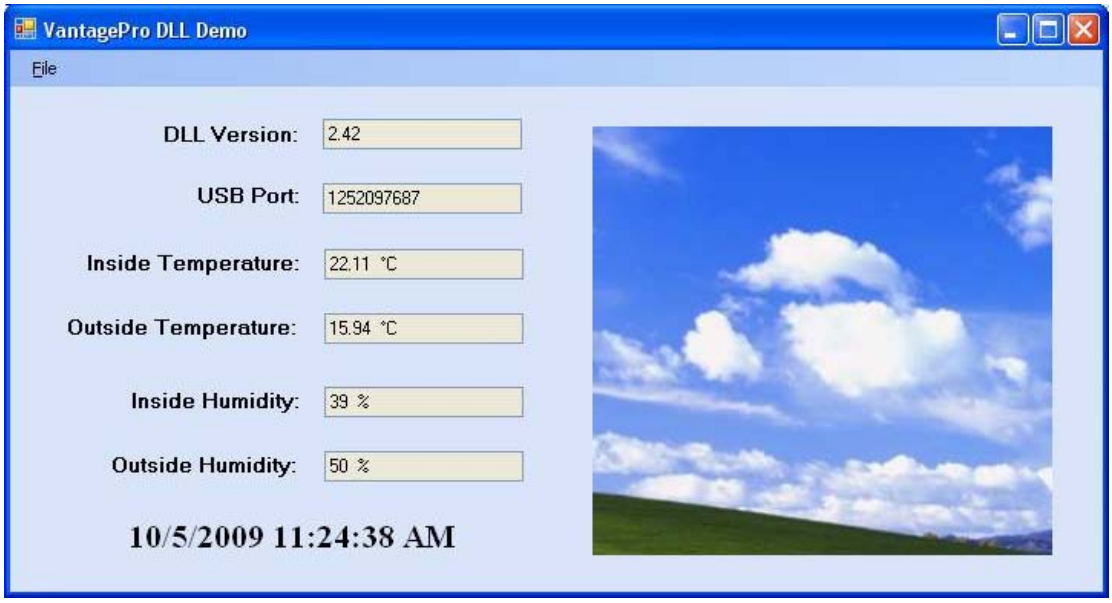
If you are accessing the DLL through Visual Basic, care must be taken in choosing data types of the parameters to DLL functions.

For example, 'int' in C++ occupies 4 bytes, while in Visual Basic, 'Integer' occupies 2 bytes. In this case you have to choose 'Long' (if you are using Visual Basic). The DLL functions may not function propery if you pass mismatched data types.

Given the VantageProDll C/C++ header file, Visual Basic (VB) programmers could add VB support for VantageProDLL by writing the corresponding Visual Basic style header file themselves. In fact, numerous developers did just that to a limited extent for VantageProDLL DLL 2.3 or earlier. To use VantageProDLL, the supplied "VantageProDllDemo.vb" needs to be included in your Visual Basic Project/Solution

The Example Program.

This program is written with the .Net version of Visual Studio 2008 Visual Basic. It is not an equivalence of the Visual C++ example, which demonstrates a broader DLL usage than this little VB example. VB users should probably look at the C++ example too for further information as VB DLL usage is very similar that of C++. This VB example program just gets and displays the DLL version, USB WeatherLink device's serial Number at the start of the program, and displays a sample bitmap. Then it gets and displays the Inside Temperature, Outside Temperature, Inside Humidity, Outside Humidity and the Time. The unit for temperature could be in Celcius, or Farenheit, depending on the firmware settings.


VantagePro DLL Demo
File
DLL Version:
2.42
USB Port:
1252097687
Inside Temperature:
22.11 °C
Outside Temperature:
15.94 °C
Inside Humidity:
39 %
Outside Humidity:
50 %
10/5/2009 11:24:38 AM