

## 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.
<b>VantageProDll.vb</b>	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 3<sup>rd</sup> 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

### [Functions](#)

### [Structures](#)

### [Constants](#)

### [Appendixes](#)

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](#)

<a href="#">GetDIIVersion_V</a>	<a href="#">OpenCommPort_V</a>	<a href="#">OpenDefaultCommPort_V</a>	
<a href="#">CloseCommPort_V</a>	<a href="#">SetCommTimeoutVal_V</a>	<a href="#">GetUnits_V</a>	<a href="#">SetUnits_V</a>
<a href="#">SetRainCollectorModel_V</a>	<a href="#">GetRainCollectorModel_V</a>	<a href="#">OpenUSBPort_V</a>	<a href="#">OpenTCPIPPort_V</a>
<a href="#">CloseUSBPort_V</a>	<a href="#">CloseTCPIPPort_V</a>	<a href="#">SetVantageTimeoutVal_V</a>	<a href="#">SetVantageTimeoutVal_V</a>
<a href="#">GetUSBDevSerialNumber_V</a>			

## Lowlevel Functions

<a href="#">GetSerialChar_V</a>	<a href="#">PutSerialStr_V</a>	<a href="#">PutSerialChar_V</a>
---------------------------------	--------------------------------	---------------------------------

## Station Configuration Functions

<a href="#">InitStation_V</a>			
<a href="#">GetModelNo_V</a>	<a href="#">GetStationTime_V</a>	<a href="#">SetStationTime_V</a>	<a href="#">GetArchivePeriod_V</a>
<a href="#">SetArchivePeriod_V</a>	<a href="#">GetStationFirmwareDate_V</a>	<a href="#">GetStationFirmwareVersion_V</a>	<a href="#">GetReceptionData_V</a>
<a href="#">GetVantageLat_V</a>	<a href="#">SetVantageLat_V</a>	<a href="#">GetVantageLon_V</a>	<a href="#">SetVantageLon_V</a>
<a href="#">SetVantageLamp_V</a>	<a href="#">SetNewBaud_V</a>	<a href="#">GetVantageTxConfig_V</a>	<a href="#">SetVantageTxConfig_V</a>
<a href="#">GetBarometerData_V</a>	<a href="#">PutBarometer_V</a>	<a href="#">PutTotalRain_V</a>	
<a href="#">GetRainCollectorModelOnStation_V</a>		<a href="#">GetAndSetRainCollectorModelOnStation_V</a>	
<a href="#">SetRainCollectorModelOnStation_V</a>			
<a href="#">GetWindCupSize_V</a>	<a href="#">SetWindCupSize_V</a>	<a href="#">GetTempAvg_V</a>	<a href="#">SetTempAvg_V</a>
<a href="#">GetTimeZoneSettings_V</a>	<a href="#">SetTimeZoneSettings_V</a>	<a href="#">IsNewLoopPackageSupported_V</a>	

## Current Data Functions

<a href="#">LoadCurrentVantageData_V</a>			
<a href="#">GetBarometer_V</a>	<a href="#">GetOutsideTemp_V</a>	<a href="#">GetDewPt_V</a>	<a href="#">GetWindChill_V</a>
<a href="#">GetInsideTemp_V</a>	<a href="#">GetInsideHumidity_V</a>	<a href="#">GetOutsideHumidity_V</a>	<a href="#">GetTotalRain_V</a>
<a href="#">GetDailyRain_V</a>	<a href="#">GetMonthlyRain_V</a>	<a href="#">GetStormRain_V</a>	
<a href="#">GetWindSpeed_V</a>	<a href="#">GetWindDir_V</a>	<a href="#">GetWindDirStr_V</a>	<a href="#">GetRainRate_V</a>
<a href="#">GetET_V</a>	<a href="#">GetMonthlyET_V</a>	<a href="#">GetYearlyET_V</a>	<a href="#">GetSolarRad_V</a>
<a href="#">GetUV_V</a>	<a href="#">GetHeatIndex_V</a>	<a href="#">GetActiveAlarms_V</a>	<a href="#">GetCurrentDataByID_V</a>
<a href="#">GetCurrentDataStrByID_V</a>	<a href="#">GetStartOfCurrentStorm_V</a>	<a href="#">GetSunriseTime_V</a>	<a href="#">GetSunsetTime_V</a>

## Alarm Functions

<a href="#">LoadVantageAlarms_V</a>	<a href="#">SetVantageAlarms_V</a>	<a href="#">ClearVantageAlarms_V</a>	
<a href="#">GetBarRiseAlarm_V</a>	<a href="#">GetBarFallAlarm_V</a>	<a href="#">GetTimeAlarm_V</a>	<a href="#">GetTimeAlarmStr_V</a>
<a href="#">GetInsideLowTempAlarm_V</a>	<a href="#">GetInsideHiTempAlarm_V</a>	<a href="#">GetOutsideLowTempAlarm_V</a>	<a href="#">GetOutsideHiTempAlarm_V</a>
<a href="#">GetLowInsideHumAlarm_V</a>	<a href="#">GetHiInsideHumAlarm_V</a>	<a href="#">GetLowOutsideHumAlarm_V</a>	<a href="#">GetHiOutsideHumAlarm_V</a>
<a href="#">GetLowWindChillAlarm_V</a>	<a href="#">GetLowDewPtAlarm_V</a>	<a href="#">GetHiDewPtAlarm_V</a>	<a href="#">GetHiSolarRadAlarm_V</a>
<a href="#">GetHiWindSpeedAlarm_V</a>	<a href="#">GetHi10MinWindSpeedAlarm_V</a>	<a href="#">GetHiHeatIndexAlarm_V</a>	<a href="#">GetHiTHSWAlarm_V</a>
<a href="#">GetHiRainRateAlarm_V</a>	<a href="#">GetHiDailyRainAlarm_V</a>	<a href="#">GetHiRainStormAlarm_V</a>	<a href="#">GetFlashFloodAlarm_V</a>
<a href="#">GetHiUVAAlarm_V</a>	<a href="#">GetHiUVMedAlarm_V</a>		
<a href="#">GetLowExtraTempAlarm_V</a>	<a href="#">GetHiExtraTempAlarm_V</a>	<a href="#">GetLowExtraHumAlarm_V</a>	<a href="#">GetHiExtraHumAlarm_V</a>
<a href="#">GetLowSoilTempAlarm_V</a>	<a href="#">GetHiSoilTempAlarm_V</a>	<a href="#">GetLowSoilMoistureAlarm_V</a>	<a href="#">GetHiSoilMoistureAlarm_V</a>
<a href="#">GetLowLeafTempAlarm_V</a>	<a href="#">GetHiLeafTempAlarm_V</a>	<a href="#">GetLowLeafWetAlarm_V</a>	<a href="#">GetHiLeafWetAlarm_V</a>
<a href="#">PutBarRiseAlarm_V</a>	<a href="#">PutBarFallAlarm_V</a>	<a href="#">PutTimeAlarm_V</a>	
<a href="#">PutInsideLowTempAlarm_V</a>	<a href="#">PutInsideHiTempAlarm_V</a>	<a href="#">PutOutsideLowTempAlarm_V</a>	<a href="#">PutOutsideHiTempAlarm_V</a>
<a href="#">PutLowInsideHumAlarm_V</a>	<a href="#">PutHiInsideHumAlarm_V</a>	<a href="#">PutLowOutsideHumAlarm_V</a>	<a href="#">PutHiOutsideHumAlarm_V</a>
<a href="#">PutLowWindChillAlarm_V</a>	<a href="#">PutLowDewPtAlarm_V</a>	<a href="#">PutHiDewPtAlarm_V</a>	<a href="#">PutHiSolarRadAlarm_V</a>
<a href="#">PutHiWindSpeedAlarm_V</a>	<a href="#">PutHi10MinWindSpeedAlarm_V</a>	<a href="#">PutHiHeatIndexAlarm_V</a>	<a href="#">PutHiTHSWAlarm_V</a>
<a href="#">PutHiRainFloodAlarm_V</a>	<a href="#">PutRainPerDayAlarm_V</a>	<a href="#">PutRainStormAlarm_V</a>	<a href="#">PutRainRateAlarm_V</a>
<a href="#">PutHiUVAAlarm_V</a>	<a href="#">PutHiUVMedAlarm_V</a>		
<a href="#">PutLowExtraTempAlarm_V</a>	<a href="#">PutHiExtraTempAlarm_V</a>	<a href="#">PutLowExtraHumAlarm_V</a>	<a href="#">PutHiExtraHumAlarm_V</a>
<a href="#">PutLowSoilTempAlarm_V</a>	<a href="#">PutHiSoilTempAlarm_V</a>	<a href="#">PutLowSoilMoistureAlarm_V</a>	<a href="#">PutHiSoilMoistureAlarm_V</a>
<a href="#">PutLowLeafTempAlarm_V</a>	<a href="#">PutHiLeafTempAlarm_V</a>	<a href="#">PutLowLeafWetAlarm_V</a>	<a href="#">PutHiLeafWetAlarm_V</a>
<a href="#">ClearBarRiseAlarm_V</a>	<a href="#">ClearBarFallAlarm_V</a>	<a href="#">ClearTimeAlarm_V</a>	
<a href="#">ClearInsideLowTempAlarm_V</a>	<a href="#">ClearInsideHiTempAlarm_V</a>	<a href="#">ClearOutsideLowTempAlarm_V</a>	<a href="#">ClearOutsideHiTempAlarm_V</a>
<a href="#">ClearLowInsideHumAlarm_V</a>	<a href="#">ClearHiInsideHumAlarm_V</a>	<a href="#">ClearLowOutsideHumAlarm_V</a>	<a href="#">ClearHiOutsideHumAlarm_V</a>
<a href="#">ClearLowWindChillAlarm_V</a>	<a href="#">ClearLowDewPtAlarm_V</a>	<a href="#">ClearHiDewPtAlarm_V</a>	<a href="#">ClearHiSolarRadAlarm_V</a>
<a href="#">ClearHiWindSpeedAlarm_V</a>	<a href="#">ClearHi10MinWindSpeedAlarm_V</a>	<a href="#">ClearHiHeatIndexAlarm_V</a>	<a href="#">ClearHiTHSWAlarm_V</a>
<a href="#">ClearHiRainFloodAlarm_V</a>	<a href="#">ClearHiRainPerDayAlarm_V</a>	<a href="#">ClearRainStormAlarm_V</a>	<a href="#">ClearRainRateAlarm_V</a>
<a href="#">ClearHiUVAAlarm_V</a>	<a href="#">ClearHiUVMedAlarm_V</a>		
<a href="#">ClearLowExtraTempAlarm_V</a>	<a href="#">ClearHiExtraTempAlarm_V</a>	<a href="#">ClearLowExtraHumAlarm_V</a>	<a href="#">ClearHiExtraHumAlarm_V</a>

[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

[DownloadData\\_V](#) [DownloadWebData\\_V](#) [GetMemoryArchiveCountAfterDate\\_V](#) [GetNumberOfArchiveRecords\\_V](#)  
[GetMemoryArchiveRecordCount\\_V](#) [GetArchiveRecord\\_V](#) [GetArchiveRecordEx\\_V](#)

## Clear Functions

[ClearVantageLows\\_V](#) [ClearVantageHighs\\_V](#) [ClearVantageAlarms\\_V](#) [ClearVantageCalNums\\_V](#)  
[ClearCurrentData\\_V](#) [ClearStoredData\\_V](#) [ClearVantageDayLows\\_V](#) [ClearVantageDayHighs\\_V](#)  
[ClearVantageMonthLows\\_V](#) [ClearVantageMonthHighs\\_V](#) [ClearVantageYearLows\\_V](#) [ClearVantageYearHighs\\_V](#)  
[ClearVantageRainET\\_V](#) [ClearVantageGraphs\\_V](#) [ClearVantageAlarmBits\\_V](#)

## 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 manually 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 "bUseFractionalDegrees" 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. Negative 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 transmitter 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 GMTOffset 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 alternative 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 transmitter 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 averaging setting with the functions [GetTempAvg\\_V\(\)](#) or [SetTempAvg\\_V\(\)](#). It controls how temperature values logged in the archive memory are recorded. This feature 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_WELLINGTON	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 Parameters \(Temperature, Humidity, etc\)](#)
- [Accumulation Parameters \(rain, ET, UV Dose, etc\)](#)
- [Other Parameters](#)

**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	---
<b>2 min. Avg. Speed</b>	<b>WIND_2MIN_AVG_WDID</b>	---	---	---	---	---	---
10 min. Avg. Speed	CUR_AVG_SPEED_WDID	---	---	---	---	---	---
<b>10 min. Wind Gust</b>	<b>WIND_GUST_10MIN_WDID</b>	---	---	---	---	---	---
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	---	---	---	---	---	---
<b>Bar Altimeter</b>	<b>BAR_ALTIMETER_WDID</b>	---	---	---	---	---	---
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 function [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\(\)](#) could be called to get the serial number parameter for calling the [OpenUSBPort\\_V](#) command.

After calling [OpenCommPort\\_V](#) (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 collector 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 1<sup>st</sup> 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 2<sup>nd</sup> 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 collector 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 collector setting or [GetAndSetRainCollectorModelOnStation\\_V](#) to set the DLL's rain collector 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 occurs in setting.  
0 if successful

[Dll Functions](#)  
[Constants](#)

---

short int GetStationFirmwareDate\_V( [DateTimeStamp](#) &timeStamp)

**Description**

Fills the timeStamp parameter with the current 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 current 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 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 latitude 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 latitudes, and negative values represent Southern 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 values represent Western longitudes. 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 transmitter 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 retrieve 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 retrieve 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<sup>2</sup>.

**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 VantagePro 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 temperature 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 temperature 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 temperature 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 chill 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 function before calling this function.

Call SetVantageAlarms\_V function 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 function before calling this function.

Call SetVantageAlarms\_V function 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 function before calling this function.

Call SetVantageAlarms\_V function 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 function before calling this function.

Call SetVantageAlarms\_V function 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 primarily for backward compatibility with previous DLL versions. The [Get\\_\\_\\_ByID\\_V](#) functions are the preferred method for accessing high/low data. See the VantagePro 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 occurred, 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 weather 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 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

---

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 calculated as  
 $\text{rawOutsideTemp} = 75.5 (\text{CurrentVantageCalibration.tempOut} - \text{CurrentVantageCalibration.tempOutOffset});$   
 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 calculated as  
 $\text{rawOutsideTemp} = 75.5 (\text{CurrentVantageCalibration.tempOut} - \text{CurrentVantageCalibration.tempOutOffset});$   
 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 calculated as  
 $\text{rawOutsideTemp} = 75.5 (\text{CurrentVantageCalibration.tempOut} - \text{CurrentVantageCalibration.tempOutOffset});$   
 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 calculated as  
 $\text{rawOutsideTemp} = 75.5 (\text{CurrentVantageCalibration.tempOut} - \text{CurrentVantageCalibration.tempOutOffset});$   
 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 calculated as

$\text{rawInsideTemp} = 69.5$  (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 calculated as

$\text{rawInsideTemp} = 75.5$  (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 [CurrentVantageCalibration](#)struct

**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 calculated 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 [CurrentVantageCalibration](#)struct

**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 calculated 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 [CurrentVantageCalibration](#)struct

**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 [CurrentVantageCalibration](#)struct

**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 calculated 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 transferred 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++)
{
    &nbsp;&nbsp;&nbsp;GetArchiveRecord_V(&record, i);
    &nbsp;&nbsp;&nbsp;GetArchiveRecordEx_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 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**

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 (nominally every 10 seconds).

**Return Values**  
**COM\_ERROR** if error  
**0** if successful

[Dll Functions](#)  
[Constants](#)

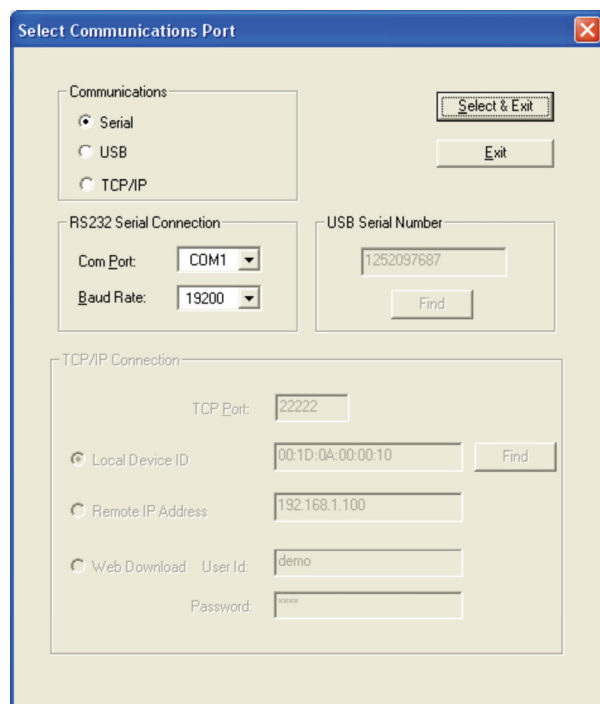
**Appendix: DLL Usage Examples**

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.

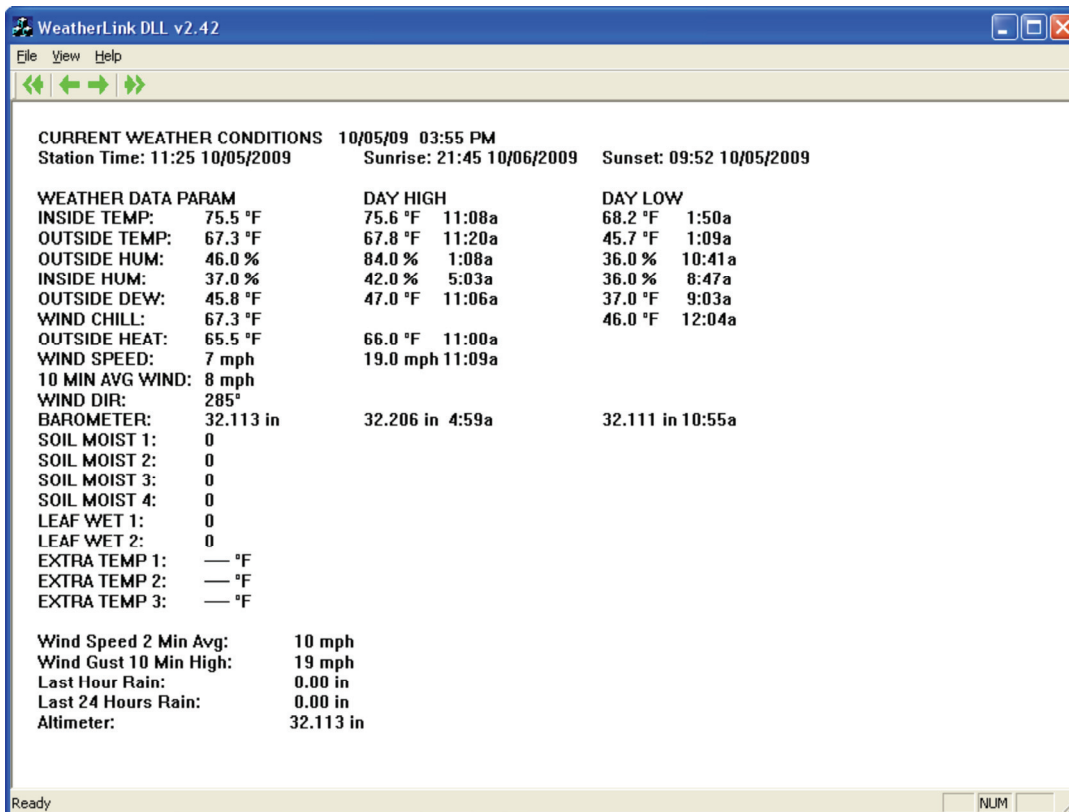
**1. Visual C++ Example Program**

As the DLL is being updated, the original Example Application is also being expanded to show usage of the support of USB WeatherLink device, TCP/IP 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.

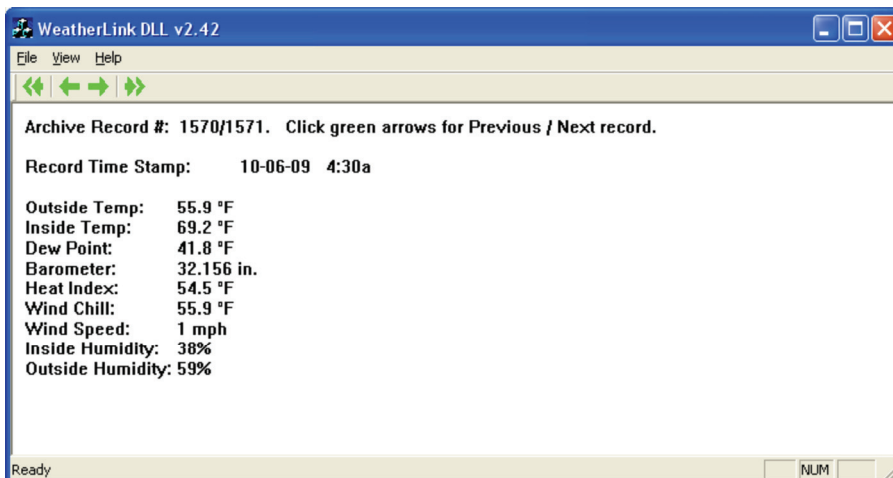
The Communications Port Setup screen let user select from various communication options:



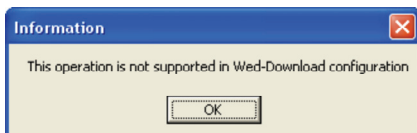
**A Current Data screen.**



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

We 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 properly 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.

