AeroShield Class

AeroShield.DroneTracker

[Visual Basic]
```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Public Class DroneTracker
Inherits UserControl
```

[C#]
```
[Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()]
public class DroneTracker : UserControl
```

[C++]
```
[Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()]
```

Requirements: Namespace: AeroShieldAPI

Platforms: Windows Phone 8.1, Windows Phone 8, Windows 8.1, Windows Server 2012 R2, Windows 8, Windows Server 2012, Windows 7, Windows Vista SP2, Windows Server 2008 (Server Core Role not supported), Windows Server 2008 R2 (Server Core Role supported with SP1 or later; Itanium not supported)

Assembly: AeroShieldAPI (in AeroShieldAPI.dll)

```
public ref class DroneTracker : public UserControl^
```

Properties: BandEnabled, Configurationfile, EmailAddress, EmailAuthenticate, EmailHostname, EmailPassword, EmailPortNumber, EmailUsername, FilterTransientEvents, MapFileName, MaskOffset, MaskTraceCount, Monitoring, OpenReports, RecordStationaryEvents, ReportFolder, SendEmail, ShowMap, ShowStatusbar, ShowToolBar

Methods: ClearIgnoreList, CreateMasks, DoHealthCheck, GetMonitorBand, LatencyTest, LoadConfiguration, Monitor, MonitorAdd, MonitorFind, MonitorListClear, MonitorRemove, MonitorUpdate, SaveConfiguration, SendRSMCommand, SetMonitorBand, StartTCPServer, StopTracking, TrackAt, UIAction

Events: Hovering, MaskViolation, MouseOverToolbar, StatusMessage, TrackingEvent, TrackingStarted, TrackingStopped

Structures: DroneListInfo, MonitorBand, RemoteMonitor

Enumerations: MaskLevels, TrackingModes, TransientFilterLevel

DroneTracker.AlwaysGenerateReport Property

Description: Sets/Gets a Boolean value that determines if a report is automatically generated after the conclusion of each tracking event. If true, the report default folder is the user's Documents folder. The report folder can be set with the DroneTracker.

[Visual Basic]
```
Public Property AlwaysGenerateReport() As Boolean
```

[C#]
```
public bool AlwaysGenerateReport {get; set;}
```

[C++]
```
public:
property bool AlwaysGenerateReport {
bool get( );
void set(
bool value
);
}
```

DroneTracker.EmailSenderAddress Property

Description: Sets/Gets the return email address used when sending notification emails.

[Visual Basic]
```
Public Property EmailSenderAddress() As String
```

[C#]
```
public string EmailSenderAddress {get; set;}
```

[C++]
```
public:
```

```
property String^ EmailSenderAddress {
String^ get( );
void set(
String^ value
);
}
```

DroneTracker.LatencyFactor Property

Description Gets/Sets the latency factor. This is an integer value between 1 and 10. It is automatically set when doing DroneTracker.LatencyTest. This property allows you to get the set value, or to set it manually.

[Visual Basic]
```
Public Property LatencyFactor() As Integer
```

[C#]
```
public int LatencyFactor {get; set;}
```

[C++]
```
public:
property int LatencyFactor {
int get( );
void set(
int value
);
}
```

DroneTracker.BandEnabled Property

Description Sets/Gets the current status of each RF band. There are 5 bands defined. The first is the 2.4 GHz ISM band. The second is the 5.0 GHz ISM band. The other three are custom bands and can be set to any hardware supported frequency range. This property allows the user to set the bands in use. If only the first band is enabled, for instance, then only the 2.4 GHz band will be monitored.

[Visual Basic]
```
Public Property BandEnabled( _
ByVal Index As Integer _
) As Boolean
```

[C#]
```
[System.Runtime.CompilerServices.IndexerName("BandEnabled")]
public bool this[
int Index
] {get; set;}
```

[C++]
```
public:
property bool BandEnabled[int] {
bool get(
int Index
);
void set(
int Index
bool value
);
}
```

Parameters Index

DroneTracker.CheckIfConfigurationHasChanged Method

Description This method returns true if the current configuration differs from that saved in the current configuration file. This allows the controlling program to determine if the configuration file needs to be saved, or whether to prompt the user regarding

saving the configuration file. If the AeroShield user control is being used, this prompt is taken care of in the UI.

[Visual Basic] Public Sub CheckIfConfigurationHasChanged()

[C#] public System.Void CheckIfConfigurationHasChanged()

[C++] public:

System::Void CheckIfConfigurationHasChanged()

DroneTracker.ClearIgnoreList Method

Description Removes all entries in the signal Ignore List.

[Visual Basic] Public Sub ClearIgnoreList()

[C#] public System.Void ClearIgnoreList()

[C++] public:

System::Void ClearIgnoreList()

DroneTracker.Configurationfile Property

Description Gets/Sets the configuration file name. The Get property returns the name of the current configuration file. Setting the property value causes the specified file to be loaded and the program settings stored in the file to be applied.

[Visual Basic] Public Property Configurationfile() As String

[C#] public string Configurationfile {get; set;}

[C++] public:

property String^ Configurationfile {

String^ get( );

void set(

String^ value

);

}

DroneTracker.CreateMasks Method

Description Causes DroneTracker to talk with each RSM in the list and to collect max hold traces for each active band. The traces are used to create mask, the violation of which triggers a tracking event. The Mask trace count and Offset values should be set first, as well as activating and configuring the sweep bands that will be monitored.

[Visual Basic] Public Sub CreateMasks()

[C#] public System.Void CreateMasks()

[C++] public:

System::Void CreateMasks()

DroneTracker.DoHealthCheck Method

Description Communicate with each RSM in the list. Verifies that the RSMs respond as expected.

Return value is a string containing the model, installed options, firmware version, GPS Fix, and Ping time for each RSM.

[Visual Basic] Public Function DoHealthCheck() As String

[C#] public string DoHealthCheck()

[C++] public:

String^ DoHealthCheck()

DroneTracker.EmailAddress Property

Description The DroneTracker Class can send an email at the end of each tracking event. This property sets/gets the email address that the notification will be sent to.

[Visual Basic]
```
Public Property EmailAddress() As String
```

[C#]
```
public string EmailAddress {get; set;}
```

[C++]
```
public:
property String^ EmailAddress {
String^ get( );
void set(
String^ value
);
}
```

DroneTracker.EmailAuthenticate Property

Description DroneTracker uses an outgoing email server to send event notifications. The server you use is up to you, and probably depends on your corporate IT infrastructure. Many email servers require user authentification. This setting should match the requirements of the server you chose to use. If you choose to use Authentification, then you must provide a username and password for an account on the email server.

[Visual Basic]
```
Public Property EmailAuthenticate() As Boolean
```

[C#]
```
public bool EmailAuthenticate {get; set;}
```

[C++]
```
public:
property bool EmailAuthenticate {
bool get( );
void set(
bool value
);
}
```

DroneTracker.EmailHostname Property

Description Sets/Gets the current email hostname.

[Visual Basic]
```
Public Property EmailHostname() As String
```

[C#]
```
public string EmailHostname {get; set;}
```

[C++]
```
public:
property String^ EmailHostname {
String^ get( );
void set(
String^ value
);
}
```

DroneTracker.EmailPassword Property

Description Gets/Sets the email server account password.

[Visual Basic]
```
Public Property EmailPassword() As String
```

[C#]
```
public string EmailPassword {get; set;}
```

[C++]
```
public:
property String^ EmailPassword {
String^ get( );
void set(
```

```
String^ value
);
}
```

DroneTracker.EmailPortNumber Property

Description Gets/Sets the email server port number. This is a value provided by the administrator of the server.

[Visual Basic]
```
Public Property EmailPortNumber() As Integer
```

[C#]
```
public int EmailPortNumber {get; set;}
```

[C++]
```
public:
property int EmailPortNumber {
int get( );
void set(
int value
);
}
```

DroneTracker.EmailUsername Property

Description Gets/Sets the username of the account being used to send email notifications.

[Visual Basic]
```
Public Property EmailUsername() As String
```

[C#]
```
public string EmailUsername {get; set;}
```

[C++]
```
public:
property String^ EmailUsername {
String^ get( );
void set(
String^ value
);
}
```

DroneTracker.FilterTransientEvents Property

Description Gets/Sets the use of Transient Event filtering in the detection algorithm. DroneTracker detects a possible drone event by a violation of a sweep trace mask. The public bands used for drone control are often noisy, and there can be many signals that are not drone related that have to be looked at by the software to determine if it is in fact a drone. Setting this property requires the DroneTracker API software to verify that a signal is present in 3 consecutive trace sweeps before it register and looks to see if it is a drone. These will often remove a number of mask violations that otherwise trigger positive signals that have to be evaluated. This comes at a cost, as it takes longer to trigger a detection event, so the notification is delayed.

[Visual Basic]
```
Public Property FilterTransientEvents() As Boolean
```

[C#]
```
public bool FilterTransientEvents {get; set;}
```

[C++]
```
public:
property bool FilterTransientEvents {
bool get( );
void set(
bool value
);
}
```

DroneTracker.GetMonitorBand Method

Description Returns the configuration of the sweep band requested. Bands are numbered 1 to 5.

[Visual Basic]
```
Public Function GetMonitorBand( _
ByVal Index As Integer _
) As MonitorBand
```

[C#]
```
public
MonitorBand GetMonitorBand(
int Index
)
```

[C++]
```
public:
MonitorBand GetMonitorBand(
int Index
)
```

Parameters Index

Input parameter: Band number to return (1-5)

Return Value: A structure of type MonitorBand

DroneTracker.GetTrackList Property

Description This returns a set of GPS coordinates for an active track. Tracks are only available while AeroShield is actively tracking a drone. One the drone has left the area, the track is stored in the tracking file and cleared from active memory. Use this property to get an active track. Use the AeroShield Track Viewer to get a track that is no longer active.

[Visual Basic]
```
Public ReadOnly Property GetTrackList( _
ByVal Index As Integer, _
ByVal Averaging As Integer _
) As PointD()
```

[C#]
```
[System.Runtime.CompilerServices.IndexerName("GetTrackList")]
public PointD[] this[
int Index,
int Averaging
] {get;}
```

[C++]
```
public:
property array< PointD >^ GetTrackList[int, int] {
array< PointD >^ get(
int Index,
int Averaging
);
}
```

Parameters Index

Averaging

DroneTracker.LatencyTest Method

Description Performs a test on each RSM to determine the average network latency for command responses. Sets the Latency factor used to compensate for high latency networks.

[Visual Basic]
```
Public Function LatencyTest() As Boolean
```

[C#]
```
public bool LatencyTest()
```

[C++] public:

bool LatencyTest()

Return value: Boolean value if test completed successfully.

DroneTracker.LoadConfiguration Method

Description Load a configuration file saved on disk. The configuration file contains all of the DroneTracker settings, such as the RSM list, sweep band settings, email configurations, etc. Depending on the display options, it may make changes that are not at all obvious to the end user.

[Visual Basic] Public Sub LoadConfiguration(_

ByVal Filename As String _

)

[C#] public System.Void LoadConfiguration(

string Filename

)

[C++] public:

System::Void LoadConfiguration(

String^ Filename

)

Parameters Filename

DroneTracker.MapFileName Property

Description If you have selected to use the DroneTracker map display, this is the filename of the map file to load and display. There are separate instruction for the preparation of a map file.

[Visual Basic] Public Property MapFileName() As String

[C#] public string MapFileName {get; set;}

[C++] public:

property String^ MapFileName {

String^ get( );

void set(

String^ value

);

}

DroneTracker.MaskDepth Property

Description Gets/Sets the MaskDepth. When collecting sweep data for traces, how long you collect can be a critical factor determining the performance of AeroShield. If you collect for too short a time, you will miss a lot of intermittent signals that may routinely pop up in the ISM bands. These signals will trigger a lot of positive tracking events. While AeroShield is determining a positive can be ignored (which can take many minutes), a true drone event may be missed. So a longer capture time when creating masks is generally preferable. However, collecting for too long also raises the noise floor, meaning that it is harder to detect weak signals. So the best tradeoff is to collect long enough to see most transient signals that regularly occur, but not longer than necessary to keep the noise floor as low as possible. The best setting, therefore, depends on the location. IF the bands are noisy, then choose a longer MaskDepth. IF the bands are very quiet, then choose a lower MaskDepth.

The MaskDepth takes on 4 possible values. Low, Medium, High, Very High.

When you set the mask depth, you also set the dB offset used to create the mask. The dB offset is a simple offset value added to each mask point to lift it above noise. Longer acquisition (Very High mask Depth) corresponds to a lower dB

offset, since the noise floor is higher. Setting MaskDepth sets both the length of time used to collect sweeps for the mask, as well as setting the dB offset for each band.

[Visual Basic]
```
Public Property MaskDepth() As MaskLevels
```

[C#]
```
public MaskLevels MaskDepth {get; set;}
```

[C++]
```
public:
property MaskLevels MaskDepth {
MaskLevels get( );
void set(
MaskLevels value
);
}
```

DroneTracker.MonitorAdd Method

Description: Adds a new RSM to the RSM list.

[Visual Basic]
```
Public Sub MonitorAdd( _
ByVal Name As String, _
ByVal URL As String, _
ByVal Port As Integer, _
ByVal Latitude As Double, _
ByVal Longitude As Double _
)
```

[C#]
```
public System.Void MonitorAdd(
string Name,
string URL,
int Port,
double Latitude,
double Longitude
)
```

[C++]
```
public:
System::Void MonitorAdd(
String^ Name,
String^ URL,
int Port,
double Latitude,
double Longitude
)
```

Parameters:
Name as String
URL as String
Port as Integer
Latitude as Double
Longitude as Double

DroneTracker.MonitorCount Property

Description: Returns the number of RSMs in the current RSM list.

[Visual Basic]
```
Public ReadOnly Property MonitorCount() As Integer
```

[C#]
```
public int MonitorCount {get;}
```

[C++] public:
property int MonitorCount {
int get( );
}

DroneTracker.MonitorFind Method

Description The method returns the probe number based on the probe name. The number is the index number according to the display order in the Drone list of the user control. If the user control is not being used, then it will be the ordinal number of the drone in the configuration file.

[Visual Basic] Public Function MonitorFind(_
ByVal Name As String _
) As Integer

[C#] public int MonitorFind(
string Name
)

[C++] public:
int MonitorFind(
String^ Name
)

Parameters
Name

DroneTracker.MonitorListClear Method

Description Clears the RSM list.

[Visual Basic] Public Sub MonitorListClear()

[C#] public System.Void MonitorListClear()

[C++] public: System::Void MonitorListClear()

DroneTracker.Monitor Method

Description Returns the details of the requested RSM.

[Visual Basic] Public Function Monitor(_
ByVal Index As Integer _) As RemoteMonitor

[C#] public RemoteMonitor Monitor(
int Index
)

[C++] public: RemoteMonitor Monitor(int Index)

Parameters Index of the RSM to return details on. This is the ordinal value of the RSM in the RSM list.

Return value: A structure of type RemoteMonitor

DroneTracker.Monitoring Property

Description Sets/Gets the current monitoring state. Set this property to True to initiate monitoring for drones on the RSM network.

[Visual Basic] Public Property Monitoring() As Boolean

[C#] public bool Monitoring {get; set;}

[C++] public:
property bool Monitoring {

bool get( );
void set(
bool value
);
}

DroneTracker.MonitorRemove Method

Description Removes a particular RSM, by name, form the active RSM list.

[Visual Basic]
```
Public Sub MonitorRemove( _
ByVal Name As String _
)
```

[C#]
```
public System.Void MonitorRemove(
string Name
)
```

[C++]
```
public:
System::Void MonitorRemove(
String^ Name
)
```

Parameters Name

DroneTracker.MonitorStatus Property

Description Returns the status of AeroShield Monitoring. This will have one these values: Idle, Tracking, Masks, LatencyTest. It informs of the current activity going on with AeroShield.

[Visual Basic]
```
Public ReadOnly Property MonitorStatus() As String
```

[C#]
```
public string MonitorStatus {get;}
```

[C++]
```
public:
property String^ MonitorStatus {
String^ get( );
}
```

DroneTracker.MonitorUpdate Method

Description Updates details for a specific RSM.

[Visual Basic]
```
Public Sub MonitorUpdate( _
ByVal Index As Integer, _
ByVal Name As String, _
ByVal URL As String, _
ByVal Port As Integer, _
ByVal Latitude As Double, _
ByVal Longitude As Double _
)
```

[C#]
```
public System.Void MonitorUpdate(
int Index,
string Name,
string URL,
int Port,
double Latitude,
double Longitude
```

)

[C++] public:
System::Void MonitorUpdate(
int Index,
String^ Name,
String^ URL,
int Port,
double Latitude,
double Longitude
)

Parameters Index
Name
URL
Port
Latitude
Longitude

DroneTracker.OpenReports Property

Description Sets/Gets a value to determine whether tracking reports are automatically opened in the default web browser at the end of each tracking event. This is useful if you are at the terminal watching tracking events. However, it may be annoying if the detection is running automatically. There is no reason to open all of the reports if there is no operator present to view them.

[Visual Basic] Public Property OpenReports() As Boolean

[C#] public bool OpenReports {get; set;}

[C++] public:
property bool OpenReports {
bool get( );
void set(
bool value
);
}

DroneTracker.ProbeList Property

Description Returns the list of RSMs in the RSM list.

[Visual Basic] Public ReadOnly Property ProbeList() As String

[C#] public string ProbeList {get;}

[C++] public:
property String^ ProbeList {
String^ get( );
}

DroneTracker.RecordStationaryEvents Property

Description DroneTracker monitors and detects any mask violation events that exceed the threshold and width requirements. If a source does not move at all, it is considered a non-drone event, and normally not recorded in the tracking event list. (In practice, the event shows up in the list when first detected, but then is removed if it proves to be a stationary event.) If The property is True, the event stays in the Event List, otherwise it is removed.

[Visual Basic] Public Property RecordStationaryEvents() As Boolean

[C#] public bool RecordStationaryEvents {get; set;}

[C++]
```
public:
property bool RecordStationaryEvents {
bool get( );
void set(
bool value
);
}
```

DroneTracker.ReportFolder Property

Description At the end of each tracking event, a report may be automatically created and stored on the local disk space. This property Sets/Gets the report folder full path name. The default is [MyDocuments]\Anritsu.

[Visual Basic] Public Property ReportFolder() As Object

[C#] public object ReportFolder {get; set;}

[C++]
```
public:
property object^ ReportFolder {
object^ get( );
void set(
object^ value
);
}
```

DroneTracker.SaveConfiguration Method

Description Saves the current control configuration to disk file.

[Visual Basic]
```
Public Sub SaveConfiguration( _
ByVal Filename As String _
)
```

[C#]
```
public System.Void SaveConfiguration(
string Filename
)
```

[C++]
```
public:
System::Void SaveConfiguration(
String^ Filename
)
```

Parameters Filename

DroneTracker.SaveLogs Property

Description Gets/Sets a value indicating if the event log is automatically written to file. The value defaults to false as there is normally no reason to write the event log to disk. If this value is false, you can still save manually by right-clicking for the event log context menu.

[Visual Basic] Public Property SaveLogs() As Boolean

[C#] public bool SaveLogs {get; set;}

[C++]
```
public:
property bool SaveLogs {
bool get( );
void set(
```

bool value

);

}

DroneTracker.SendEmail Property

Description Sets/Gets a value determining whether tracking event email events are sent. If this is true, the email server properties (hostname, port number, authentification) must be set as well. Note: This is for email notification only. The API continues to raise events to the calling program regardless.

[Visual Basic] Public Property SendEmail() As Boolean

[C#] public bool SendEmail {get; set;}

[C++] public:

property bool SendEmail {

bool get( );

void set(

bool value

);

}

DroneTracker.SendRSMCommand Method

Description Sends a SCPI command to a specified RSM. Returns the SCPI response string if the command is a query.

[Visual Basic] Public Function SendRSMCommand(_

ByVal ProbeNumber As Integer, _

ByVal Cmd As String _

) As String

[C#] public string SendRSMCommand(

int ProbeNumber,

string Cmd

)

[C++] public:

String^ SendRSMCommand(

int ProbeNumber,

String^ Cmd)

Input Parameters Probe Number: The number of the RSM in the RSM list. If this value is 0, then the command will be sent to all RSMS.

Cmd: The SCPI command to send. Please refer to the Remote Spectrum Monitor Programming Manual for details on supported SCPI commands.

Return value The response of the command, if it is a query. Otherwise 'Ok' if the command was sent properly.

DroneTracker.SetMonitorBand Method

Description Sets the particular sweep parameters for a particular RF sweep band. This includes the start and stop frequencies, along with the RBW and reference level to be used.

[Visual Basic] Public Sub SetMonitorBand(_

ByVal Index As Integer, _

ByVal BandInfo As MonitorBand _

)

[C#] public System.Void SetMonitorBand(
int Index,
MonitorBand BandInfo
)

[C++] public:
System::Void SetMonitorBand(
int Index,
MonitorBand BandInfo
)

Parameters Index
BandInfo

DroneTracker.ShowMap Property

Description Gets/Sets whether the user choses to view the internal mapping display. If this property it , the mapping display is hidden. In many use cases, DroneTracker is an integral part of a larger system that already provides mapping abilities. In a stand-alone application, however, visual mapping may be desired through the DroneTracker API.

[Visual Basic] Public Property ShowMap() As Boolean

[C#] public bool ShowMap {get; set;}

[C++] public:
property bool ShowMap {
bool get( );
void set(
bool value
);
}

DroneTracker.ShowStatusbar Property

Description Gets/Sets a property determining if the DroneTracker control status bar is visible. When the DroneTracker control is used in the context of a larger application, if is often desirable to hide the status bar, and display the status information in the larger applications status area.

[Visual Basic] Public Property ShowStatusbar() As Boolean

[C#] public bool ShowStatusbar {get; set;}

[C++] public:
property bool ShowStatusbar {
bool get( );
void set(
bool value
);
}

DroneTracker.ShowToolBar Property

Description Gets/Sets a property determining if the DroneTracker control toolbar is visible. When the DroneTracker control is used in the context of a larger application, if is often desirable to hide the toolbar, and provide control through the host applications user interface.

[Visual Basic] Public Property ShowToolBar() As Boolean

[C#] public bool ShowToolBar {get; set;}

[C++]
```
public:
property bool ShowToolBar {
bool get( );
void set(
bool value
);
}
```

DroneTracker.StartTCPServer Method

Description This function starts the TCP/IP server. This must be called by the host program if it intends to receive and respond to a remote PC through this interface.

[Visual Basic]
```
Public Sub StartTCPServer()
```

[C#]
```
public System.Void StartTCPServer()
```

[C++]
```
public:
System::Void StartTCPServer()
```

DroneTracker.StopTracking Method

Description Stops tracking on the specified track. If tracking not active, nothing happens.

[Visual Basic]
```
Public Sub StopTracking( _
ByVal TrackIndex As Integer _
)
```

[C#]
```
public System.Void StopTracking(
int TrackIndex
)
```

[C++]
```
public:
System::Void StopTracking(
int TrackIndex
)
```

Parameters TrackIndex

DroneTracker.TimeToIgnore Property

Description Sets/Gets the number of seconds a signal is on the ignore list. If AeroShield detects a mask violation that turns out not to be a drone, it places the signal on the ignore list so it doesn't keep looking at the same signal over and over and potentially miss real drone signals. We do not want to leave a signal on the Ignore list. A signal that triggers AeroShield must be above the mask, so it is probably intermittent, or of a single occurrence nature. It is likely at a frequency that will be available when it goes away, and could very well be chosen by a drone controller at a later time. So we do not want to ignore signals at this frequency for a long time. So it needs to expire from the Ignore List. By default, signals expire on the ignore list in 90 seconds. But this is user settable.

[Visual Basic]
```
Public Property TimeToIgnore() As Integer
```

[C#]
```
public int TimeToIgnore {get; set;}
```

[C++]
```
public:
property int TimeToIgnore {
int get( );
void set(
int value
);
}
```

DroneTracker.TrackAt Method

Description Normally DroneTracker is set to automatically sweep and monitor. A tracking event begins with a mask violation. This function allows the user to initial a tracking event without waiting for a mask violation. The user must provide a frequency, RSM and antenna channel.

[Visual Basic]
```
Public Sub TrackAt( _
ByVal Probe As Integer, _
ByVal Channel As Integer, _
ByVal Freq As Long _
)
```

[C#]
```
public System.Void TrackAt(
int Probe,
int Channel,
long Freq
)
```

[C++]
```
public:
System::Void TrackAt(
int Probe,
int Channel,
long Freq
)
```

Parameters
Probe
Channel
Freq

DroneTracker.TrackingMode Property

[Visual Basic]
```
Public Property TrackingMode() As TrackingModes
```

[C#]
```
public TrackingModes TrackingMode {get; set;}
```

[C++]
```
public:
property TrackingModes TrackingMode {
TrackingModes get( );
void set(
TrackingModes value
);
}
```

DroneTracker.TransientFilter Property

Description Off | Low | High. Off means a single mask violation triggers an AeroShield attempt to track and determine if a signal is really a drone. Low means a signal must persist for 3 consecutive sweeps (typically about 3 seconds). High means the signal must persist for 5 consecutive sweeps (typically about 5 seconds).

[Visual Basic]
```
Public Property TransientFilter() As TransientFilterLevel
```

[C#]
```
public TransientFilterLevel TransientFilter {get; set;}
```

[C++]
```
public:
property TransientFilterLevel TransientFilter {
TransientFilterLevel get( );
void set(
TransientFilterLevel value
```

```
);
}
```

DroneTracker.VerboseMode Property

Description Sets/Gets a value determining how much information is displayed in the tracking log. Verbose set to true causes a lot more information to be displayed. This is useful when setting up a system and wanting to know more about what it going on under the hood. It is not useful when things are operating normally, as it can be too much information to parse easily.

Verbose set to True also provides a warning message if the trace data exceeds the mask at more than 50% of the data points. This is an indication that the mask is probably inappropriate and needs to be redone.

[Visual Basic]
```
Public Property VerboseMode() As Boolean
```

[C#]
```
public bool VerboseMode {get; set;}
```

[C++]
```
public:
property bool VerboseMode {
bool get( );
void set(
bool value
);
}
```

MaskViolation Event

Description This event is raise each time there is a detected mask violation. Parameters include the frequency, and bandwidth of the infringing signal. It also provides the power over the mask and the probe number that detected the violation.

[Visual Basic]
```
Public Event MaskViolation( _
ByVal Frequency As Long, _
ByVal Bandwidth As Long, _
ByVal PowerOver As Single, _
ByVal DetectionProbe As Integer _
)
```

[C#]
```
public event MaskViolationEventHandler MaskViolation
delegate void MaskViolationEventHandler(
long Frquency,
long Bandwidth,
float PowerOver,
int DetectionProbe
)
```

[C++]
```
public:
event MaskViolationEventHandler^ MaskViolation {
void add (MaskViolationEventHandler^ value);
void remove (MaskViolationEventHandler^ value);
}
delegate void MaskViolationEventHandler(
long Frquency,
long Bandwidth,
float PowerOver,
int DetectionProbe
```

)

Parameters

Frequency

Bandwidth

PowerOver

DetectionProbe

MonitorBand Structure

Description An RF sweep band. This structure is used for configuring a sweep band.

[Visual Basic] Public Structure MonitorBand

[C#] public struct MonitorBand

[C++] public value struct MonitorBand

Requirements

Namespace: AeroshieldAPI

Assembly: AeroshieldAPI (in AeroshieldAPI.dll)

Fields

Active, Antenna, RBW, ReferenceLevel, StartFrequency, StopFrequency

StatusMessage Event

Description This event is raised each time something is written to the on-screen event log. This event still fires if the DroneTracker user control is not displayed. This provides a means for an external program using AeroShield to access all event data.

[Visual Basic] Public Event StatusMessage(_

ByVal Msg As String _

)

[C#] public event StatusMessageEventHandler StatusMessage

delegate void StatusMessageEventHandler(

string Msg

[C++] public:

event StatusMessageEventHandler^ StatusMessage {

void add (StatusMessageEventHandler^ value);

void remove (StatusMessageEventHandler^ value);

}

delegate void StatusMessageEventHandler(

String^ Msg

)

Parameters

Msg

TrackingEvent Event

Description This is an event raised by the DroneTracker control each time a position estimate is obtained. It returns a tracking point index and the point coordinates.

[Visual Basic] Public Event TrackingEvent(_

ByVal Index As Integer, _

ByVal Location As PointF _

)

[C#] public event TrackingEventEventHandler TrackingEvent
delegate void TrackingEventEventHandler(
int Index,
PointF Location
)

[C++] public:
event TrackingEventEventHandler^ TrackingEvent {
void add (TrackingEventEventHandler^ value);
void remove (TrackingEventEventHandler^ value);
}
delegate void TrackingEventEventHandler(
int Index,
PointF Location
)

Parameters Index
Location

TrackingStarted Event

Description Event raised when a new tracking event begins. This returns an index into the tracking event list.

[Visual Basic] Public Event TrackingStarted(_
ByVal Index As Integer _
)

[C#] public event TrackingStartedEventHandler TrackingStarted
delegate void TrackingStartedEventHandler(
int Index
)

[C++] public:
event TrackingStartedEventHandler^ TrackingStarted {
void add (TrackingStartedEventHandler^ value);
void remove (TrackingStartedEventHandler^ value);
}
delegate void TrackingStartedEventHandler(
int Index
)

Parameters Index

TrackingStopped Event

Description Event raised when a tracking event ends. It returns an index into the tracking event list, and a parameter that indicates the reason the event ended. Reasons to end a tracking event are: Event ended (meaning DroneTracker could no longer find an IQ correlation, so the drone is assumed to have left the area or landed); Stationary event (meaning the RF source detected did not move, so assumed not to be a drone); User ended (meaning the user has used a UI element in the control to manually stop tracking).

[Visual Basic] Public Event TrackingStopped(_
ByVal Index As Integer, _
ByVal Reason As Integer _

)

[C#] public event TrackingStoppedEventHandler TrackingStopped

delegate void TrackingStoppedEventHandler(

int Index,

int Reason

)

[C++] public:

event TrackingStoppedEventHandler^ TrackingStopped {

void add (TrackingStoppedEventHandler^ value);

void remove (TrackingStoppedEventHandler^ value);

}

delegate void TrackingStoppedEventHandler(

int Index,

int Reason

)

Parameters Index

Reason

DroneListInfo Structure

Description This structure is used to retrieve tracking event information from the tracking list.

[Visual Basic] Public Structure DroneListInfo

[C#] public struct DroneListInfo

[C++] public value struct DroneListInfo

Requirements

Namespace: AeroshieldAPI

Assembly: AeroshieldAPI (in AeroshieldAPI.dll)

Fields

Active, Bandwidth, DetectionCount, Frequency, LibraryMatch, Location, Points, ReportFile, TimeStampIn, TimeStampOut, TrackAveraging, TrackCount, TracksFile, TraveledDistance, TriggerChannel, TriggerProbe

Active Field

Description The tracking event is currently still ongoing if this value is true.

[Visual Basic] Public Active As Boolean

[C#] public bool Active

[C++] public:

bool Active;

Bandwidth Field

Description RF bandwidth of the mask violation that started the tracking event.

[Visual Basic] Public Bandwidth As Long

[C#] public long Bandwidth

[C++] public:

long Bandwidth;

DetectionCount Field

Description The number of times an event has triggered with the same set of RF parameters (frequency range).

[Visual Basic] Public DetectionCount As Integer

[C#] public int DetectionCount

[C++] public:
int DetectionCount;

Frequency Field

Description The center frequency of the tracking signal.

[Visual Basic] Public Frequency As Long

[C#] public long Frequency

[C++] public:
long Frequency;

LibraryMatch Field

Description Not used.

[Visual Basic] Public LibraryMatch As String

[C#] public string LibraryMatch

[C++] public:
String^ LibraryMatch;

Location Field

Description Not used.

[Visual Basic] Public Location As PointF

[C#] public PointF Location

[C++] public:
PointF Location;

Points Field

Description A list of GPS coordinates that make up the track flown by the drone.

[Visual Basic] Public Points As String

[C#] public string Points

[C++] public:
String^ Points;

ReportFile Field

Description The full pathname of the tracking report.

[Visual Basic] Public ReportFile As String

[C#] public string ReportFile

[C++] public:
String^ ReportFile;

TimeStampIn Field

Description When the tracking event started.

[Visual Basic] Public TimeStampIn As Date

[C#] public DateTime TimeStampIn

[C++] public:
DateTime TimeStampIn;

TimeStampOut Field

Description When the tracking event ended.

[Visual Basic] Public TimeStampOut As Date

[C#] public DateTime TimeStampOut

[C++] public:

DateTime TimeStampOut;

TrackAveraging Field

Description Tracks, when displayed on the map, use a running average to smooth the track lines. (There is inherent uncertainty in each TDOA location estimate, and this makes for a much smoother and typically more accurate lilne) The values can be: None; Low; Medium; or High.

[Visual Basic] Public TrackAveraging As Integer

[C#] public int TrackAveraging

[C++] public:

int TrackAveraging;

TrackCount Field

Description The number of track points in the tracking event.

[Visual Basic] Public TrackCount As Integer

[C#] public int TrackCount

[C++] public:

int TrackCount;

TracksFile Field

Description Not used

[Visual Basic] Public TracksFile As String

[C#] public string TracksFile

[C++] public:

String^ TracksFile;

TraveledDistance Field

Description The total distance tracked.

[Visual Basic] Public TraveledDistance As Integer

[C#] public int TraveledDistance

[C++] public:

int TraveledDistance;

TriggerChannel Field

Description The sweep band that triggered the event.

[Visual Basic] Public TriggerChannel As Integer

[C#] public int TriggerChannel

[C++] public:

int TriggerChannel;

TriggerProbe Field

Description The RSM that first triggers the tracking event.

[Visual Basic] Public TriggerProbe As String

[C#] public string TriggerProbe

[C++] public:

String^ TriggerProbe;

Active Field

Description Whether or not this band is active in monitoring.

[Visual Basic] Public Active As Boolean

[C#] public bool Active

[C++] public:
bool Active;

Antenna Field

Description The antenna port used on each RSM for this band.

[Visual Basic] Public Antenna As Integer

[C#] public int Antenna

[C++] public:
int Antenna;

RBW Field

Description Resolution bandwidth

[Visual Basic] Public RBW As Integer

[C#] public int RBW

[C++] public:
int RBW;

ReferenceLevel Field

Description Reference level

[Visual Basic] Public ReferenceLevel As Single

[C#] public float ReferenceLevel

[C++] public:
float ReferenceLevel;

StartFrequency Field

Description Start frequency

[Visual Basic] Public StartFrequency As Long

[C#] public long StartFrequency

[C++] public:
long StartFrequency;

StopFrequency Field

Description Stop frequency

[Visual Basic] Public StopFrequency As Long

[C#] public long StopFrequency

[C++] public:
long StopFrequency;