



MEA

3.1

MeshAPI for Pocket PC 2003

MeshAPI (Version 3.1.1.13)

Document Revision Number 3.1.3



Copyrights

The Motorola products described in this document may include copyrighted Motorola computer programs. Laws in the United States and other countries reserve for Motorola certain exclusive rights for copyrighted computer programs. Accordingly, any copyrighted Motorola computer programs contained in the Motorola products described in this document may not be copied or reproduced in any manner without the express written permission of Motorola. Furthermore, the purchase of Motorola products shall not be deemed to grant either directly or by implication, estoppels or otherwise, any license under the copyrights, patents or patent applications of Motorola, except for the normal nonexclusive, royalty-free license to use that arises by operation of law in the sale of a product.

Disclaimer

Please note that certain features, facilities and capabilities described in this document may not be applicable to or licensed for use on a particular system, or may be dependent upon the characteristics of a particular mobile subscriber unit or configuration of certain parameters. Please refer to your Motorola contact for further information.

Trademarks

Motorola, the Motorola logo, and all other trademarks identified as such herein are trademarks of Motorola, Inc. All other product or service names are the property of their respective owners.

Copyrights

© 2005-2006 Motorola, Inc. All rights reserved. No part of this document may be reproduced, transmitted, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, without the prior written permission of Motorola, Inc.

Table of Contents

1	INTRODUCTION.....	3
2	FEATURES AND BENEFITS	3
3	SOFTWARE REQUIREMENTS.....	3
4	THE OBJECTS AND INTERFACES	3
5	THE IMESHNET INTERFACE.....	4
5.1	Methods Supported by This Interface.....	4
5.1.1	<i>Initialize</i>	4
5.1.2	<i>GetMAC</i>	6
5.1.3	<i>GetRoutingTable.....</i>	7
5.1.4	<i>GetLinkQuality.....</i>	9
5.1.5	<i>GetGenericOID</i>	10
5.1.6	<i>SetGenericOID.....</i>	12
5.1.7	<i>SetEjectMessage</i>	13
5.1.8	<i>GetRawNeighborTable.....</i>	14
5.2	Connection Point Events Supported by the _IMeshNetEvents Interface	16
5.2.1	<i>GeoLocation.....</i>	16
5.2.2	<i>LinkResistance.....</i>	16
5.2.3	<i>RawRoutingTable.....</i>	16
5.2.4	<i>RoutingTable.....</i>	16
5.2.5	<i>NeighborTable.....</i>	17
5.2.6	<i>RawNeighborTable</i>	17
5.2.7	<i>LicenseGEO.....</i>	17
5.2.8	<i>LicenseRT</i>	17
5.2.9	<i>AuthLog.....</i>	18
APPENDIX A	19
	Embedded Visual C++ project setup	19
	OID's List Table.....	20
	IP Schemes Table	21
	Device Types Table	21



This page intentionally left blank.

1 Introduction

The Mesh Application Programming Interface (MeshAPI) functions as an interface to the Motorola WMC6300 line of products. Developers can utilize MeshAPI to develop their own client applications. For PocketPC, MeshAPI is an in-process Component Object Model (COM) server that exposes an interface called IMeshNet. To use this interface, a developer must include the *MeshAPI.tlb* type library file or MeshAPI.dll file when working with it in development projects. In addition, a MeshAPI client has the option of receiving data either on demand or as an event. MeshTray is an example of a Mesh Enabled Architecture (MEA™) application that uses MeshAPI.

2 Features and Benefits

MeshAPI provides developers with the ability to configure a local MEA adapter, as well as retrieve information such as routing tables and neighbor tables.

3 Software Requirements

MeshAPI is compatible with most development languages that support Microsoft's Component Object Model (COM). All interface methods are supported by the C++ language using Microsoft's Embedded Visual C++,

4 The Objects and Interfaces

The COM object that is exposed by MeshAPI is CoMeshNet. This is the object you must instantiate in order to communicate with MeshAPI. The Interfaces that are supported on this object are IMeshNet and _IMeshNetEvents. The IMeshNet interface is the primary interface used to communicate with MeshAPI. It is this interface that will be used to obtain information from the MEA adapter in the system. The _IMeshNetEvents interface is a dispinterface that is used to receive events from MeshAPI.

5 The IMeshNet Interface

The IMeshNet interface controls the MEA card on the system. It is an IDispatch derived interface, and is therefore capable of supporting custom or automation clients. Scripted client support is not available since method parameters are more strongly typed than scripted clients can support. Attempting to access any of its methods while the MEA card is not present on the system will result in an "E_ACCESSDENIED" HRESULT error code.

5.1 Methods Supported by This Interface

5.1.1 Initialize

Initializes MeshAPI. Calls to other methods exported by IMeshNet will fail until this method is called.

HRESULT Initialize ();

Parameters

none

Return Values

One of the values described in the following table is returned.

Value	Description
S_OK	Success.
S_FALSE	Initialize has already been called.
E_ACCESSDENIED	The MEA card is not present on the system.
REGDB_E_CLASSNOTREG	MeshAPI is not installed properly. Please reinstall.
E_UNEXPECTED	MeshAPI could not be initialized.
E_FAIL	The driver failed to open the MEA adapter.

Example

The following example shows how to use the Initialize() method:

```
#import "meshAPI.dll" no_namespace named_guids
.
.
.
```

```
HRESULT hr;
IMeshNet *pMeshNet;
hr = CoCreateInstance( CLSID_CoMeshNet, NULL, CLSCTX_ALL, uuidof(IMeshNet),
                      (void **) & pMeshNet);
if (SUCCEEDED(hr))
{
    hr = pMeshNet->Initialize();
    if (SUCCEEDED(hr))
    {
        ...
    }
    pMeshNet->Release();
}
```

5.1.2 GetMAC

GetMAC gets the MAC address of the MEA adapter. This is the value that is displayed for MAC on the *Status* page in MeshTray.

```
HRESULT GetMAC( BSTR *pbstrMAC );
```

Parameters

pbstrMAC

[out] is a pointer to a BSTR to contain the MAC address.

Return Values

One of the values described in the following table is returned.

Value	Description
S_OK	Success.
E_ACCESSDENIED	The MEA card is not present on the system.
E_FAIL	The driver failed to open the MEA adapter.

Example

The following example retrieves the MAC address. (Also see [Appendix A](#))

```
CComBSTR bstrMAC;  
IMeshNet *pMeshNet = NULL;  
HRESULT hr = pMeshNet->CreateInstance("MeshAPI.MeshNet");  
if( SUCCEEDED(hr) )  
{  
    hr = pMeshNet->Initialize();  
    if (SUCCEEDED(hr))  
    {  
        hr = pMeshNet->GetMAC( &bstrMAC );  
    }  
    pMeshNet->Release();  
}
```

5.1.3 GetRoutingTable

Get the Raw Routing Table from the MEA device. The resulting value contains information about routes to other MEA devices.

GetRoutingTable(VARIANT *pvtRawRT)

Parameters

pvtRawRT

[out] a pointer to a VARIANT to receive the raw routing table. Each entry in the table is a structure containing information about an individual route entry in the route table. The structure is an RT_INFO structure described below.

Return Values

One of the values described in the following table is returned.

Value	Description
S_OK	Success.
E_ACCESSDENIED	The MEA card is not present on the system or the Routing function's license has not been granted.
E_POINTER	The API failed to compose the SAFEARRAY object.
E_FAIL	The driver failed to open the MEA adapter.

Remarks

The client must call `SafeArrayDestroy(vtRawRT.parray)` to free up the SAFEARRAY's memory.

The number of entries in the table can be found using the following expression:

```
vtRawRT.parray->rgsabound->cElements / sizeof(RT_INFO)
```

Example

```
CCoVariant vtRawRT;    // No need to call SafeArrayDestroy, the CCoVariant class
                        // will clean itself up once it goes out of scope.
IMeshNet pMeshNet;

// Each entry in the returned table is formatted as an RT_INFO structure
// The structure is aligned on a single byte boundary
#pragma pack(push)
#pragma pack(1)
struct
```



```
{
    unsigned char dst_addr[6];           // MAC address of the route destination
    unsigned char next_hop_addr[6];     // MAC address of the next hop to the dest
    unsigned short link_resistance;     // Total link resistance along the path
    unsigned short num_hops;           // Number of hops to the dest
    unsigned short dev_type;           // Device Type (see below)
} RT_INFO;
#pragma pack(pop)

// These are the valid values returned for dev_type in the RT_INFO structure
enum DeviceTypeEnum
{
    DT_IAP = 0, // IAP - Firmware 7.0 or later
    DT_WIRELESS_ROUTER = 1, // Wireless Router - Firmware 7.0 or later
    DT_SUBSCRIBER_DEVICE = 2, // Subscriber device - Firmware 7.0 or later
    DT_ENH_WIRELESS_ROUTER = 3, // Enhanced WR - Firmware 7.0 or later
    DT_VEHICLE_MOUNTED_MODEM = 4, // VMM - Firmware 7.0 or later
    DT_SUBSCRIBER_DEVICE_V6 = 100, // Subscriber device - Firmware 6.0 or earlier
    DT_WIRELESS_ROUTER_V6 = 101, // Wireless Router - Firmware 6.0 or earlier
    DT_IAP_V6 = 102, // IAP - Firmware 6.0 or earlier
    DT_NON_ROUTING_DEVICE = 104, // Non-Route Device - Firmware 6.0 or earlier
};

HRESULT hr = pMeshNet->CreateInstance("MeshAPI.MeshNet");
if (SUCCEEDED(hr))
{
    Sleep(1500); // Give API time to start
    hr = pMeshNet->GetRoutingTable( &vtRawRT );
    int iNumElements = vtRawRT.parray->rgsabound->cElements / sizeof(RT_INFO);
    RT_INFO *aRouteTable = new RT_INFO[iNumElements];
    memcpy( aRouteTable, vtRawRT.parray->pvData,
           vtRawRT.parray->rgsabound[0].cElements );
    ::SafeArrayDestroy( vtRawRT.parray );

    for (int iIndex=0; iIndex<iNumElements; iIndex++)
    {
        ...
        // Process route table
        CString strMac;
        strMac.Format(_T("%02X-%02X-%02X-%02X-%02X-%02X"),
                    aRouteTable[iIndex].dst_addr[0],
                    aRouteTable [iIndex].dst_addr[1],
                    aRouteTable [iIndex].dst_addr[2],
                    aRouteTable [iIndex].dst_addr[3],
                    aRouteTable [iIndex].dst_addr[4],
                    aRouteTable [iIndex].dst_addr[5]);

        CString str;
        str.Format(_T("%lu: Type:%u MAC:%s Hops:%u \n"),
                  iIndex+1, aRouteTable[iIndex].dev_type, strMac,
                  (long) aRouteTable[iIndex].num_hops);

        TRACE(str);
        ...
    }
    delete []aRouteTable;
}
}
```

5.1.4 GetLinkQuality

Get the link quality value to the associated IAP. This is the value that is displayed for IAP Link Quality on the *Status* page in MeshTray.

```
HRESULT GetLinkQuality( unsigned short *pnLQ, BSTR *pbstrIapMac, UINT *pnReserved );
```

Parameters

pnLQ

[out] a pointer to an unsigned short to receive the Link Quality

pbstrIapMAC

[out] is a pointer to a BSTR to receive the MAC address of the IAP.

pnReserved

[out] a pointer to an unsigned int. Reserved.

Return Values

One of the values described in the following table is returned.

Value	Description
S_OK	Success.
E_ACCESSDENIED	The MEA card is not present on the system.
E_FAIL	The driver failed to open the MEA adapter.

Example

```
IMeshNet *pMeshNet = NULL;
HRESULT hr = pMeshNet->CreateInstance("MeshAPI.MeshNet");
if( SUCCEEDED(hr) )
{
    hr = pMeshNet->Initialize();
    if (SUCCEEDED(hr))
    {
        USHORT nLQ;
        CComBSTR bstrMAC;
        UINT nReserved;
        hr = pMeshNet->GetLinkQuality( &nLQ, &bstrMAC, &nReserved );
    }
    pMeshNet->Release();
}
```

5.1.5 GetGenericOID

Request a value from the MEA adapter by its specific object identifier (OID). See the *OID's List Table* section in [Appendix A](#) for a non-exhaustive list of OIDs accepted by this method.

```
HRESULT GetGenericOID( unsigned long ulOID, VARIANT *vtValue, unsigned long *lpSize );
```

Parameters

ulOID

[in] A 32-bit value containing the OID being requested. [see Appendix A]

vtValue

[in, out] The address of a VARIANT to receive the value requested. This must be initialized prior to calling GetGenericOID().

lpSize

[in, out] The address of a 32-bit value containing the size of the buffer allocated to vtValue above. On input, this value contains the size of the buffer allocated prior to the call. On output, this value contains the number of bytes placed into vtValue.

Return Values

One of the values described in the following table is returned.

Value	Description
S_OK	Success.
E_ACCESSDENIED	The MEA card is not present on the system.
E_INVALIDARG	Unrecognized VARIANT type.
E_FAIL	The driver failed to open the MEA adapter.

Example

The following example retrieves data specified by an OID [see Appendix A].

```
// This example will retrieve the Build Date of the firmware
// This is returned as a text value

ULONG lSize = 20;
LPSAFEARRAY lpsa = SafeArrayCreateVector(VT_UI1, 0, lSize);

VARIANT vt;
vt.vt = VT_ARRAY|VT_UI1;    // The Variant is an array of bytes
vt.parray = lpsa;          // Assign the allocated array

HRESULT hr = pMeshNet->CreateInstance("MeshAPI.MeshNet");
```

```
if (SUCCEEDED(hr))
{
    hr = pMeshNet->Initialize();
    if (SUCCEEDED(hr))
    {
        hr = pMeshNet->GetGenericOID(G_OID_BUILD_DATE, &vt, &lSize);
        if ( SUCCEEDED(hr) )
        {
            CString strGenericOID = (PBYTE) vtue.parray->pvData;
            TRACE(_T("Build Date is: %s\n"), strGenericOID);
            ::SafeArrayDestroy( vtValue.parray );
        }
    }
    pMeshNet->Release();
}
```

5.1.6 SetGenericOID

Set a value in the MEA adapter by its specific object identifier (OID).

```
HRESULT SetGenericOID( [in] unsigned long ulOID, [in, out] VARIANT *vtValue, [in, out]
unsigned long *lpSize );
```

Parameters

ulOID

[in] the OID being set. [see Appendix A]

vtValue

[in, out] a pointer to the location where the value is to be stored.

lpSize

[in, out] a pointer to the buffer size: On input, this is the size of the buffer you allocated. On output, this receives the size of the data returned in vtValue.

Return Values

One of the values described in the following table is returned.

Value	Description
S_OK	Success.
E_ACCESSDENIED	The MEA card is not present on the system.
E_INVALIDARG	Unrecognized VARIANT type.
E_FAIL	The driver failed to open the MEA adapter.

Example

The following example sets the transmission priority to be used by the MEA adapter.

```
VARIANT vt;
ZeroMemory(&vt, sizeof(vt));
vt.vt = VT_UINT;
vt.uintVal = 3;
USHORT nSize = sizeof(DWORD);

HRESULT hr = pMeshNet->CreateInstance( "MeshAPI.MeshNet" );
if (SUCCEEDED(hr))
{
    hr = pMeshNet->Initialize();
    if (SUCCEEDED(hr))
    {
        hr = pMeshApi->SetGenericOID( AC_OID_HOST_CURRENT_PRIORITY, &vt, &nSize);
    }
    pMeshNet->Release();
}
```

5.1.7 SetEjectMessage

Set the message that the driver will send to the application to let the application know that the adapter has been ejected from the PCMCIA slot.

HRESULT SetEjectMessage(*UINT nMsg*, *VARIANT *vtHwnd*);

Parameters

nMsg

[in] The value of the message to be sent to the specified window.

vtHwnd

[in] The address of a **VARIANT** containing the window handle that should receive the eject message. **Note: Only the last application to call SetEjectMessage will receive the eject message.**

Return Values

One of the values described in the following table is returned.

Value	Description
S_OK	Success.
E_ACCESSDENIED	The MEA card is not present on the system or Initialize() has not been called.
E_INVALIDARG	The variant was not passed by reference.
E_FAIL	The driver failed to open the MEA adapter.

Example

The following example illustrates how to set the eject message.

```

HRESULT hr = pMeshNet->CreateInstance("MeshAPI.MeshNet");
if (SUCCEEDED(hr))
{
    hr = pMeshNet->Initialize();
    if (SUCCEEDED(hr))
    {
        VARIANT vtHwnd;

        ZeroMemory( &vtHwnd, sizeof(vtHwnd) );
        vtHwnd.vt = VT_BYREF;
        vtHwnd.byref = m_hWnd; // HWND of window to receive the eject msg.
        hr = m_pMeshNet ->SetEjectMessage( WM_EXIT_APP, &vtHwnd );
    }
    pMeshNet->Release();
}
    
```

5.1.8 GetRawNeighborTable

Get the Raw Neighbor Table from the MEA device. The resulting value contains information about other nearby MEA devices.

GetRawNeighborTable(VARIANT *pvtRawNT)

Parameters

pvtRawNT

[out] a pointer to a VARIANT to receive the raw neighbor table. Each entry in the table is a structure containing information about an individual neighbor entry in the neighbor table. The structure is an NT_INFO structure described below.

Return Values

One of the values described in the following table is returned.

Value	Description
S_OK	Success.
E_ACCESSDENIED	The MEA card is not present on the system or the Routing function's license has not been granted.
E_POINTER	The API failed to compose the SAFEARRAY object.

Remarks: The client must call SafeArrayDestroy(vtRawNT.parray) to free up the SAFEARRAY's memory. The number of entries in the table can be found using the following expression:

```
vtRawNT.parray->rgsabound->cElements / sizeof(NT_INFO)
```

Example

```
CCoVariant vtRawNT; // No need to call SafeArrayDestroy, the CCoVariant class
// will clean itself up once it goes out of scope.
IMeshNet pMeshNet;

// Each entry in the returned table is formatted as an NT_INFO structure
// The structure is aligned on a single byte boundary
#pragma pack(push)
#pragma pack(1)
struct
{
    unsigned char dst_addr[6]; // MAC address of the neighbor
    unsigned short dev_type; // Device Type (see below)
    unsigned char lastRateIndex; // last data rate, from ATP data
    signed char signalLevel; // Signal Level in dBm from ATP data
} NT_INFO;
#pragma pack(pop)
```

```

// These are the valid values returned for dev_type in the NT_INFO structure
enum DeviceTypeEnum
{
    DT_IAP                = 0,        // IAP                - Firmware 7.0 or later
    DT_WIRELESS_ROUTER    = 1,        // Wireless Router    - Firmware 7.0 or later
    DT_SUBSCRIBER_DEVICE  = 2,        // Subscriber device  - Firmware 7.0 or later
    DT_ENH_WIRELESS_ROUTER = 3,      // Enhanced WR        - Firmware 7.0 or later
    DT_VEHICLE_MOUNTED_MODEM = 4,    // VMM                - Firmware 7.0 or later
    DT_SUBSCRIBER_DEVICE_V6 = 100,    // Subscriber device  - Firmware 6.0 or earlier
    DT_WIRELESS_ROUTER_V6 = 101,     // Wireless Router    - Firmware 6.0 or earlier
    DT_IAP_V6             = 102,     // IAP                - Firmware 6.0 or earlier
    DT_NON_ROUTING_DEVICE = 104,     // Non-Route Device   - Firmware 6.0 or earlier
};

// These are the valid values returned for lastRateIndex in the NT_INFO structure
enum LastRateEnum
{
    RATE_1_5      = 3,        // 1.5 Mbps
    RATE_3_0      = 4,        // 3.0 Mbps
    RATE_4_0      = 5,        // 4.0 Mbps
    RATE_6_0      = 6,        // 6.0 Mbps
};

HRESULT hr = pMeshNet->CreateInstance("MeshAPI.MeshNet");
if (SUCCEEDED(hr))
{
    Sleep(1500);           // Give API time to start
    hr = pMeshNet->GetRawNeighborTable( &vtRawNT );
    int iNumElements = vtRawNT.parray->rgsabound->cElements / sizeof(NT_INFO);
    NT_INFO *aNeighborTable = new NT_INFO[iNumElements];
    memcpy( aNeighborTable, vtRawNT.parray->pvData,
            vtRawNT.parray->rgsabound[0].cElements );
    ::SafeArrayDestroy( vtRawNT.parray );

    for (int iIndex=0; iIndex<iNumElements; iIndex++)
    {
        ...
        // Process neighbor table
        CString strMac;
        strMac.Format(_T("%02X-%02X-%02X-%02X-%02X-%02X"),
                    aNeighborTable[iIndex].dst_addr[0],
                    aNeighborTable[iIndex].dst_addr[1],
                    aNeighborTable[iIndex].dst_addr[2],
                    aNeighborTable[iIndex].dst_addr[3],
                    aNeighborTable[iIndex].dst_addr[4],
                    aNeighborTable[iIndex].dst_addr[5]);

        CString str;
        str.Format(_T("%lu: Type:%lu MAC:%s Signal:%ld LastRateIndex:%ld\n"),
                  iIndex+1, aNeighborTable[iIndex].dev_type, strMac,
                  (long) aNeighborTable[iIndex].signalLevel,
                  (long) aNeighborTable[iIndex].lastRate);

        TRACE(str);
        ...
    }
    delete [] aNeighborTable;
}

```

5.2 Connection Point Events Supported by the *_IMeshNetEvents* Interface

To establish a connection between the client and the server, the client must create an event sink object and pass its interface pointer to the server. This establishes a bi-directional communication between them. Once a connection point is established, the server will start posting event notifications to the client.

Caution

All pointers provided by the connection point should be considered temporary and must be copied immediately to a different location.

5.2.1 GeoLocation

DispID = 1

Event Data: *double* dAlt, *double* dLat, *double* dLong, *BOOL* bEnhanced

Remarks: This event returns double values for the current Altitude, Latitude, and Longitude, and a Boolean indicating if Enhanced GEO is enabled. See `GetGeoLocation()`. This event is raised approximately every 2.5 seconds. If Enhanced GEO is enabled, this event may be fired many times per second.

5.2.2 LinkResistance

DispID = 2

Event Data: *BSTR* bstrIapMAC, *unsigned short* nLR

Remarks: This event returns the MAC address of the currently associated IAP, and the link resistance to that IAP. This event is raised on approximately four second intervals. See the methods for `GetAssociatedIAP()`, and `GetIAPLinkResistance()`.

5.2.3 RawRoutingTable

DispID = 3

Event Data: *VARIANT* vtRT

Remarks: This event returns the current route table as described in call to `GetRoutingTable()`. This event is raised on approximately four second intervals.

5.2.4 RoutingTable

DispID = 4

Event Data: *VARIANT* vtRTex

Remarks: This event returns a variation of the raw routing table. This event is raised on approximately four second intervals. The Routing Table is a processed Raw Routing Table. Each Routing Table Entry is formed by comparing the latest two Raw Routing Tables and each entry is defined as:

```
struct
{
    BYTE status;
    RT_INFO rte;
} RT_INFO_EX;
```

where,

status: is the result of the table comparison and can have the following values: S (Same), U (Updated), N (New), D (Deleted).

rte: the raw table entry (see **GetRoutingTable**)

The number of entries in the table can be found by:
`vtRT.parray->rgsabound->cElements / sizeof(RT_INFO_EX)`

5.2.5 NeighborTable

DispID = 5

Event Data: *VARIANT* vtNTEx

Remarks: This event returns a variation of the neighbor table. The Neighbor Table is a processed Raw Neighbor Table. This event is raised on approximately four second intervals. Each Neighbor Table is formed by comparing the latest two Raw Neighbor Tables and each entry is defined as:

```
Struct
{
    BYTE status;
    NT_INFO nte;
} NT_INFO_EX;
```

where,

status: is the result of the table comparison and can have the following values: S (Same), U (Updated), N (New), D (Deleted).

nte: the raw table entry (see **GetRawNeighborTable**)

The number of entries in the table can be found by:
`vtNT.parray->rgsabound->cElements / sizeof(NT_INFO_EX)`

5.2.6 RawNeighborTable

DispID = 6

Event Data: *VARIANT* vtNTEx

Remarks: This event returns the neighbor table. It is raised on approximately four second intervals. See `GetRawNeighborTable()`.

5.2.7 LicenseGEO

DispID = 7

Event Data: *BOOL* bNormalGEO, *BOOL* bEnhancedGEO

Remarks: This event returns the state of the GEO license whenever it changes. It is only raised when a change in the state of the GEO license is changed remotely. It is also sent initially when MeshAPI is first started.

5.2.8 LicenseRT

DispID = 8

Event Data: *BOOL* bRT

Remarks: This event returns the state of the Routing license whenever it changes. It is only raised when a change in the state of the Routing license is changed remotely. It is also sent initially when MeshAPI is first started.



5.2.9 AuthLog

DispID = 9

Event Data: *long* GroupFailures, *long* GroupSuccesses, *long* IapFailures, *long* IapSuccesses

Remarks: This event reports the number of successful and failed authentication attempts that have occurred since the last time this event was posted. It is raised only when a change occurs in the number of authentication successes or failures.

Appendix A

Embedded Visual C++ project setup

meshAPI.dll / meshAPI.tlb

The Type Library or DLL file needs to be either copied to the project's directory or the project's include path needs to be modified to include the path to that file. For example, if the .tlb or .dll file is in C:\Program Files\meshAPI then add that path to the project by clicking: Project / Settings / C/C++ / Preprocessor / Additional Include Directories.

StdAfx.h

Add the following two lines:

```
#include <atlbase.h>
```

```
#import "meshAPI.dll" no_namespace named_guids
```

Make sure that you initialize COM in your client by calling:

AfxOleInit()

Or

CoInitialize()

OID's List Table

OID	OID Value	Read/Write	Value Type
G_OID_BUILD_DATE	0x00000004	Read	safearray (char array)
G_OID_BUILD_TIME	0x00000005	Read	safearray (char array)
G_OID_FIRMWARE_VERSION_MAJOR	0x00000016	Read	32-bit unsigned
G_OID_FIRMWARE_VERSION_MINOR	0x00000017	Read	32-bit unsigned
G_OID_FIRMWARE_VERSION_ITERATION	0x00000018	Read	32-bit unsigned
G_OID_NODE_TYPE	0x0000002b	Read	32-bit unsigned
AC_OID_HOST_USER_SPECIFIED_IP	0x00040003	Read/Write	32-bit unsigned
AC_OID_HOST_USER_SPECIFIED_SUBNET_MASK	0x00040004	Read/Write	32-bit unsigned
LS_OID_REPORT_GEO_POSITION_ON	0x000c000c	Read/Write	32-bit unsigned
LS_OID_ENHANCED_GEO_MODE_ON	0x000c0011	Read/Write	32-bit unsigned
LS_OID_CALC_GEO_LOCATION_INTERVAL	0x000c0008	Read/Write	16-bit unsigned
LS_OID_GEO_INTERVAL_LOW_LIMIT	0x000c0010	Read	BYTE
IP_OID_MAP_SERVER_IP_ADDRESS	0x00030014	Read/Write	32-bit unsigned
AC_OID_HOST_CURRENT_ADDRESSING_SCHEME	0x00040001	Read/Write	32-bit unsigned
AC_OID_HOST_CURRENT_PRIORITY	0x00040010	Read/Write	32-bit unsigned
AC_OID_HOST_ALLOWED_ADDRESSING_SCHEMES	0x00040002	Read	32-bit unsigned
AC_OID_HOST_MAXIMUM_PRIORITY	0x00040013	Read	32-bit unsigned

IP Schemes Table

Scheme	Value
AC_ADDRESSING_SCHEME_REMOTE_DHCP	0x04
AC_ADDRESSING_SCHEME_LOCALLY_PROVISIONED	0x02
AC_ADDRESSING_SCHEME_USER_SPECIFIED	0x01

Device Types Table

Device Type	Value
IAP	0x00
WR	0x01
SD	0x02
EWR	0x03
VMM	0x04