Windows Platform Design Notes

Design Information for the Microsoft® Windows® Family of Operating Systems

IEEE 802.11 Network Adapter Design Guidelines for Windows XP

Abstract

This paper describes best practices and recommendations for IEEE 802.11 network adapters designed to work with Microsoft® Windows® XP.

To fully support the wireless local area network (LAN) enhancements included in Windows XP, an IEEE 802.11 network adapter should implement the requirements listed in this white paper.

May 22, 2003

Contents

Introduction....4
Network Association....4
MediaSense....4
WPA....5
Encryption/Integrity....8
Authentication Event....8
WEP Authentication....10
Link Speed....10
Power Modes....10
Client Name....10
Sub-Media Type....11
NDIS Version....11
IEEE 802.11 OIDs....11
IEEE 802.1X Support....13
Device Requirements....13
IEEE 802.11a/b Radios....13
IEEE 802.11 Data Types....14
IEEE 802.11 OID Descriptions....19
- OID_802_11_BSSID....19
- OID_802_11_SSID....19
- OID_802_11_NETWORK_TYPES_SUPPORTED....20
- OID_802_11_NETWORK_TYPE_IN_USE....21
- OID_802_11_TX_POWER_LEVEL....21
- OID_802_11_RSSI....21
- OID_802_11_RSSI_TRIGGER....22
- OID_802_11_INFRASTRUCTURE_MODE....22
- OID_802_11_FRAGMENTATION_THRESHOLD....23
- OID_802_11_RTS_THRESHOLD....23
- OID_802_11_NUMBER_OF_ANTENNAS....24
- OID_802_11_RX_ANTENNA_SELECTED....24
- OID_802_11_TX_ANTENNA_SELECTED....24
- OID_802_11_SUPPORTED_RATES....25
- OID_802_11_DESIRED_RATES....25
- OID_802_11_CONFIGURATION....26
- OID_802_11_STATISTICS....27
- OID_802_11_ADD_WEP....28
- OID_802_11_ADD_KEY....30
- OID_802_11_REMOVE_KEY....34
- OID_802_11_REMOVE_WEP....35

OID_802_11_DISASSOCIATE 35
OID_802_11_POWER_MODE 35
OID_802_11_BSSID_LIST 36
OID_802_11_ AUTHENTICATION_MODE 39
OID_802_11_PRIVACY_FILTER 40
OID_802_11_BSSID_LIST_SCAN 40
OID_802_11_ENCRYPTION_STATUS 41
OID_802_11_RELOAD_DEFAULTS 44
OID_802_11_ASSOCIATION_INFORMATION 45
OID_802_11_TEST 47
Resources and Call to Action 48

© 2003 Microsoft Corporation. All rights reserved.

Microsoft does not make any representation or warranty regarding specifications in this document or any product or item developed based on these specifications. This document is provided to you on an AS IS basis. Microsoft disclaims all express and implied warranties, including but not limited to the implied warranties or merchantability, fitness for a particular purpose and freedom from infringement. Without limiting the generality of the foregoing, Microsoft does not make any warranty of any kind that any item developed based on these specifications, or any portion of a specification, will not infringe any copyright, patent, trade secret or other intellectual property right of any person or entity in any country. It is your responsibility to seek licenses for such intellectual property rights where appropriate. Microsoft shall not be liable for any damages of any kind arising out of or in connection with the use of these specifications, including without limitation, any direct, indirect, incidental, consequential (including any lost profits), punitive or special damages, whether or not Microsoft has been advised of such damages. Some states do not allow the exclusion or limitation of liability or consequential or incidental damages; the above limitation may not apply to you.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

Microsoft, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners.

© 2002-2003 Microsoft Corporation. All rights reserved.

© 2003 Microsoft Corporation. All rights reserved.

Introduction

IEEE 802.11, IEEE 802.11a, and IEEE 802.11b are wireless LAN standards. They supply bandwidth of 1 Mbit/sec or more. However, they have a number of issues that will slow the uptake of these networks unless solved.

The issues include:

- The configuration of the client necessary for many scenarios.
- The reconfiguration required when moving between scenarios.
- The configuration required for securing wireless LANs.

To reduce the number of reconfigurations required, the Network Driver Interface Specification (NDIS) network interface card (NIC) drivers for these networks should behave in specific ways by default.

Network Association

For NDIS version 5.1 drivers, it is preferable for the NIC not to attempt to connect to a network, but rather to wait until OID_802_11_SSID is called. When the driver is loaded initially, it should generate a disconnect event.

MediaSense

Current IEEE 802.11 NDIS miniport drivers in Windows 2000 and Windows XP support MediaSense when in infrastructure mode. If the NIC is associated with an access point then a media connect event is generated . When the NIC is not associated with an access point (AP), a media disconnect event is generated. A disconnect event should not be generated when roaming from one AP to another; there should merely be a series of media connect events.

A media connect event is generated by calling **NdisMIndicateStatus** with an argument of NDIS_STATUS_MEDIA_CONNECT. Similarly, a media disconnect event is generated by calling NdisMIndicateStatus with an argument of NDIS_STATUS_MEDIA_DISCONNECT.

Connect/Disconnect Events:

- For NDIS 5.1, a media connect event must be generated in the following cases:
  - The NIC associates with a basic service set identifier (BSSID), either an AP or an ad hoc NIC.
  - When the NIC reassociates with an AP (roam to a new access point).
- A connect event must be generated when OID_802_11_BSSID is set.
- A connect event must be generated when OID_802_11_SSID is set. The connect event is expected to occur, at most, 2 seconds after setting this object identifier (OID).
- The NIC should wait 10 seconds before generating any disconnect event. No disconnect should be generated if the same or another AP for the SSID is found within this period. In this case, a connect event would be generated.
- A disconnect event must be generated when the driver is first loaded.
- A disconnect event must be generated when OID_802_11_DISASSOCIATE is set.

© 2003 Microsoft Corporation. All rights reserved.

- A connect event must be generated if a NIC is not associated with any AP and associates with an AP.
- A connect event must be generated if a NIC associates or reassociates with the same AP but not if the NIC stops sending to the AP for a brief period and starts sending again without associating or reassociating with the AP.
- If a NIC is associated with AP(a) belonging to SSID(1) and attempts to associate with AP(b) belonging to SSID(2), it generates a disconnect event with AP(a) and a connect event with AP(b).
- If a NIC is associated with AP(a) belonging to SSID(1) and attempts to associate with AP(b) belonging to SSID(1), it generates only a connect event with AP(b).
- If a NIC is in ad hoc mode it should generate a media connect event only when it sees the first ad hoc station for the SSID. When no other stations are visible, the driver should generate a media disconnect event.

Stations (STAs) Forming the IBSS Cell
The following example describes the expected behavior in IEEE 802.11 IBSS mode.

1. The first STA forming the IBSS cell must not generate a connect event and is in a disconnected media state. The rationale is that a network forms only when two or more STAs are in an IBSS cell.
2. When a second STA joins the existing IBSS cell formed in step 1, an IBSS network is formed and a connect event is generated at the two STAs. The two STAs is in a connected media state.
3. When a third STA joins the existing IBSS cell, a connect event is generated at the third STA and it is in a connected media state. Note that the media state for the first and second STAs does not change; no MediaSense events are required, and they remain in a connected media state.

STAs Leaving the IBSS Cell
1. When a STA that is a member of an IBSS cell leaves the cell, a disconnect event is generated at the STA disassociating from the particular IBSS cell, and the STA is in a disconnected media state. Note that hysteresis is applied to the disconnect event.
2. When two STAs are in an IBSS cell and one STA disassociates, a disconnect event is generated at the remaining STA and it is in a disconnected media state. Note that hysteresis is applied to the disconnect event.

Note: A media disconnect event should not be generated when the STA is responding to an encrypted packet that cannot be decrypted with a disassociate message followed by an associate request, unless the associate request fails.

WPA

Wi-Fi Protected Access (WPA) is an improved security system for 802.11. For vendors who have already implemented a driver and want to update it for WPA, the following OIDs must be changed:

- OID_802_11_AUTHENTICATION_MODE
- OID_802_11_ENCRYPTION_STATUS
- OID_802_11_BSSID_LIST

And the following new OIDs must be added:

© 2003 Microsoft Corporation. All rights reserved.

- OID_802_11_ADD_KEY
- OID_802_11_REMOVE_KEY
- OID_802_11_ASSOCIATION_INFORMATION
- OID_802_11_TEST

Support for WPA must be checked by using the OID_802_11_AUTHENTICATION_MODE OID to set the authentication mode to Ndis802_11AuthModeWPA and then querying the current authentication mode. If either the set fails or the returned value is not Ndis802_11AuthModeWPA, then it must be assumed that the driver does not support the WPA extensions.

In addition, the authentication event must be supported.

A WPA NIC must support Temporal Key Integrity Protocol (TKIP) encryption and Michael integrity, and may support Advanced Encryption Standard (AES) encryption and integrity.

The following table describes the various configuration options and the expected system behavior. If the manual key or 802.1X is above the NDIS OIDs, the OID_802_11_ADD_KEY OID is used in both cases to configure the correct key into the NIC. The “Key configured before joining network” column describes whether a key is required to be configured or not before joining the network.

Note: An illegal combination of infrastructure, authentication, and encryption modes must not generate a MediaSense connect and should generate a MediaSense disconnect if the media is already associated. For example, the combination of IBSS, WPANone, and WEP is illegal.

Infrastructure mode	Authentication mode	Encryption status	Manual Key required?	IEEE 802.1X enabled?	Key configured before joining network?
ESS	Open	None	No	No	No
ESS	Open	WEP	Optional	Optional	Yes
ESS	Shared	None	Yes	No	Yes
ESS	Shared	WEP	Optional	Optional	Yes
ESS	WPA	WEP	No	Yes	No
ESS	WPA	TKIP	No	Yes	No
ESS	WPA	AES	No	Yes	No
ESS	WPA-PSK	WEP	Yes	Yes	No
ESS	WPA-PSK	TKIP	Yes	Yes	No
ESS	WPA-PSK	AES	Yes	Yes	No
IBSS	Open	None	No	No	No
IBSS	Open	WEP	Yes	No	Yes
IBSS	Shared	None	Yes	No	Yes
IBSS	Shared	WEP	Yes	No	Yes
IBSS	WPA-None	WEP	Yes	No	Yes
IBSS	WPA-None	TKIP	Yes	No	Yes
IBSS	WPA-None	AES	Yes	No	Yes

The following table describes what OID support is required for WPA.

OID	Required
OID_802_11_AUTHENTICATION_MODE supports WPA, WPA-PSK and WPA-None for set and query	Required

© 2003 Microsoft Corporation. All rights reserved.

OID_802_11_ENCRYPTION_STATUS supports encryption1 and encryption 2 for set and query	Required
OID_802_11_BSSID_LIST returns NDIS_802_11_BSSID_LIST_EX structure	Required
OID_802_11_ASSOCIATION_INFORMATION supports query	Required
OID_802_11_ADD_KEY supports single group key set and one pairwise key (may use Group Key 0)	Required
OID_802_11_ADD_KEY supports four group keys, index 0 to 3.	Required
OID_802_11_ADD_KEY supports bit 28 set to 0	Required
OID_802_11_TEST supports generation of MEDIA_SPECIFIC_EVENT	Required
Detect Michael integrity failures for group keys and for pairwise keys and generate a MEDIA_SPECIFIC_EVENT event. Non-802.1X packets are dropped with the current keys	Required
Non-802.1X data packets are not sent until a group key is installed	Required
WPA information element in associate/reassociate request message	Required
No key is needed to associate to an AP with the privacy bit set in the AP capability	Required
TKIP 48 bit and Michael	Required

Note: The operating system checks for a WPA driver using the following checklist:

1. Set OID_802_11_AUTHENTICATION_MODE with Ndis802_11AuthModeWPA. The call must succeed.
2. Query OID_802_11_AUTHENTICATION_MODE. The call must succeed and must return Ndis802_11AuthModeWPA as the value of the OID.
3. Determine the highest supported cipher by using the following process:
    a. Set OID_802_11_ENCRYPTION_STATUS subsequently with Ndis802_11Encryption3Enabled, Ndis802_11Encryption2Enabled and Ndis802_11Encryption1Enabled in that order.
    b. Query OID_802_11_ENCRYPTION_STATUS. The query should succeed and the value returned should match the value that has been set. If this does not happen, the loop is broken and the last cipher verified successfully is assumed as the highest supported cipher.
    c. The highest supported cipher should be either TKIP or AES.
4. Set OID_802_11_ADD_KEY with a random key of length 32 and with an index of 0xc0000001. The result must be ERROR_INVALID_PARAMETER (which corresponds to NDIS_STATUS_INVALID_DATA from the driver).
5. Query OID_802_11_ASSOCIATION_INFORMATION. The call must succeed - do not check the returned value in any way.

    If all these tests pass, the NIC is marked as "WPA capable.“ Preferred networks can then be set with the extended authentication modes and encryption types. This check happens once as the card is inserted.

© 2003 Microsoft Corporation. All rights reserved.

Encryption/Integrity

If a NIC supports TKIP or AES, it should associate or reassociate using the highest setting that is enabled and compatible with the AP. If the NIC supports TKIP, it must support WEP. If the NIC supports AES, it must support TKIP and WEP.

When associating to an AP, the cipher used is a combination of the cipher allowed by the AP and the cipher enabled by the NIC. The NIC must always pick the most secure cipher possible:

- AES should be chosen if the NIC and AP support AES.
- TKIP should be chosen if both the NIC and AP support TKIP, but one of them does not support AES.
- WEP should be chosen if both the NIC and AP support WEP and TKIP or AES is not supported by both of the NIC and AP.

This decision is made after an SSID is selected and the NIC is trying to associate to the AP.

The decisions are made independently for the key mapping and default ciphers, so different ciphers are possible. For example, the AP could be sending broadcast packets on WEP while a station and AP is communicating using AES for unicast packets.

When roaming to another BSSID, the cipher and authentication suites must not be changed.

Authentication Event

A NIC should generate an NdisMIndicateStatus of NDIS_STATUS_MEDIA_SPECIFIC_INDICATION for the following events:

- A NIC must use this event to inform IEEE 802.1X about data integrity errors; in this case, the flags must be set to 0x06 or 0x0e (see below).
- A NIC may use this event to request preauthentication of BSSIDs for roaming purposes.

The StatusBuffer must contain the following structure:

typedef struct _NDIS_802_11_STATUS_INDICATION

{

```
    NDIS_802_11_STATUS_TYPE StatusType;
```

} NDIS_802_11_STATUS_INDICATION, *PNDIS_802_11_STATUS_INDICATION;

NDIS_802_11_STATUS_INDICATION must not be negative.

```
typedef enum _NDIS_802_11_STATUS_TYPE
{
    Ndis802_11StatusType_Authentication,
    Ndis802_11StatusTypeMax    // not a real type, defined as
an upper bound
} NDIS_802_11_STATUS_TYPE, *PNDIS_802_11_STATUS_TYPE;
```

StatusType contains Ndis802_11StatusType_Authentication for an Authentication Event.

© 2003 Microsoft Corporation. All rights reserved.

If the StatusType is authentication event, then an array of the following structures should appear after StatusType;

```
typedef struct _NDIS_802_11_AUTHENTICATION_REQUEST
{
    ULONG Length;            // Length of structure
    NDIS_802_11_MAC_ADDRESS Bssid;
    ULONG Flags;
} NDIS_802_11_AUTHENTICATION_REQUEST,
*PNDIS_802_11_AUTHENTICATION_REQUEST;
```

Length contains the length of the structure in bytes, from and including length to and including flags.

Bssid is a BSSID of an AP.

Flags

#define NDIS_802_11_AUTH_REQUEST_REAUTH 0x01

This bit is set if this is a request for an 802.1X re-authentication. If 802.1X has already authenticated this BSSID and this bit is not set, then 802.1X will not re-authenticate.

Note: If this bit is set, an 802.1X re-authentication is initiated. If 802_11_AUTH_REQUEST_REAUTH and 802_11_AUTH_REQUEST_GROUP_ERROR are not set, then an 802.1X authentication is only initiated if the station is not already authenticated.

#define NDIS_802_11_AUTH_REQUEST_KEYUPDATE 0x02

This bit set if this is a request for a key update. Key update happens after 802.1X authentication occurs. If this bit is set, then key update occurs even if 802.1X authentication is not triggered. This may not be set at the same time as 802_11_AUTH_REQUEST_REAUTH.

Note: If this bit is set and 802_11_AUTH_REQUEST_REAUTH is not set, then an EAPOL-Key message is sent with the request bit set, the error bit clear, and the key type set to pairwise.

#define NDIS_802_11_AUTH_REQUEST_PAIRWISE_ERROR 0x06

These bits are set if the NIC detected a data integrity error with the pairwise key for the BSSID. Any data packets that will be decrypted using this pairwise key, except IEEE 802.1X packets, must be discarded.

Note: If these bits are set and 802_11_AUTH_REQUEST_REAUTH is not set, then an EAPOL-Key message is sent with the request bit set, the error bit set, and the key type set to pairwise.

#define NDIS_802_11_AUTH_REQUEST_GROUP_ERROR 0x0E

These bits are set if the NIC detected data integrity error with any of the group keys for the BSSID. Any group keys for this BSSID must be deleted from the NIC before sending this event.

Notes:

- If these bits are set and 802_11_AUTH_REQUEST_REAUTH is not set, then an EAPOL-Key message is sent with the request bit set, the error bit set, and the key type set to group.

  Other bits in the Flags field must be set to 0.
- When the NIC changes the MIC keys, care should be taken that integrity errors are not generated due to overlap of packets being received while the key is changing.

© 2003 Microsoft Corporation. All rights reserved.

If the NIC reports two data integrity errors within 60 seconds of each other, the NIC should stop sending non-802.1X packets and should disassociate after the next 802.1X message is transmitted.

If the NIC is using this event for pre-authentication then:

- The BSSIDs should be ordered with the NICs most preferred BSSID for association purposes first in the list. The maximum number of structures in the list is five and the StatusBufferSize should be set to the size of the StatusBuffer. This should reflect the number of BSSIDs in StatusBuffer. The current associated BSSID should always be in the StatusBuffer, but may not be the first in the list if multiple BSSIDs are in the list.
- The NIC should incorporate some damping on list of BSSIDs in the authenticate event. That is, allowances should be made for BSSIDs that move in and out of range. The NIC should only request preauthentication for BSSIDs when the NIC is evaluating whether to roam to another BSSID.
- The NIC should generate the event if the NIC would like to roam, even if there are no other BSSIDs available. In this case, only the currently associated BSSID will be in the list.

WEP Authentication

A NIC should support open authentication (no key required) as the default authentication setting.

Link Speed

OID_802_11_GEN_LINK_SPEED should return the current speed that data packets can be sent. It should return the maximum speed of the NIC or return 0 rather than failure .

Power Modes

A NIC should support two power settings: one for AC-powered machines and another for battery-powered machines.

By default, the AC setting should be maximum speed, and the battery setting should be maximum battery savings. NDIS 5.1 calls the miniport driver's PnPEventNotifyHandler, on initializing or setting the miniport to power state D0, and when the power profiles change. The PnP Event is NdisDevicePnPEventPowerProfileChanged, and the InformationBuffer points to an ULONG containing:

```
//
// Power profiles
//
typedef enum _NDIS_POWER_PROFILE
{
      NdisPowerProfileBattery,
      NdisPowerProfileAcOnLine
} NDIS_POWER_PROFILE, *PNDIS_POWER_PROFILE;
```

Client Name

If the client name can be configured, it should be set to the machine name by default. NDIS 5.1 calls an optional OID after the initialization of the miniport driver

© 2003 Microsoft Corporation. All rights reserved.

and whenever the machine name changes. The OID is OID_GEN_MACHINE_NAME. The InformationBuffer contains an array of Unicode characters, and the InformationBufferLength contains the length in bytes of the array.

Sub-Media Type

The miniport driver must support the OID_GEN_PHYSICAL_MEDIUM and return a value of NdisPhysicalMediumWirelessLan.

NDIS Version

An NDIS miniport driver must declare itself as an NDIS 5.1 miniport driver in the NdisMRegisterMiniport MiniportCharacteristics structure. NDIS then supports the additional information described in this document. On platforms other than NDIS 5.1 the NdisMRegisterMiniport fails with NDIS_STATUS_BAD_VERSION. The miniport driver can then declare itself as an NDIS 5.0 miniport driver and work around the reduced functionality. Note that although it is preferable to implement on NDIS 5.1, the IEEE 802_11 OIDs can be implemented on NDIS versions earlier than NDIS 5.1.

IEEE 802.11 OIDs

A number of functions are required from the IEEE 802.11 NDIS driver to enable this new functionality. All these OIDs are made available through WMI in NDIS 5.1. The mandatory OIDs are required to be supported, but they may fail the call if the hardware does not support the functionality. The following table provides a summary of the WLAN-dependent wireless objects:

OID (Hex)	OID Name	Indication	Query	Set	Mandatory for Windows XP	Mandatory for WPA	Mandatory for Windows 2000, NT 4.0, 98/Me
0D010101	OID_802_11_BSSID		X	X	X	X	X
0D010102	OID_802_11_SSID		X	X	X	X	X
0D010203	OID_802_11_NETWORK_TYPES_SUPPORTED		X		Recommended	X	Recommended
0D010204	OID_802_11_NETWORK_TYPE_IN_USE		X	X	X	X	
0D010205	OID_802_11_TX_POWER_LEVEL		X	X			
0D010206	OID_802_11_RSSI	X	X		X	X	
0D010207	OID_802_11_RSSI_TRIGGER		X	X			
0D010108	OID_802_11_INFRASTRUCTURE_MODE		X	X	X	X	Recommended
0D010209	OID_802_11_FRAGMENTATION_THRESHOLD		X	X			
0D01020A	OID_802_11_RTS_THRESHOLD		X	X			
0D01020B	OID_802_11_NUMBER_OF_ANTENNAS		X				
0D01020C	OID_802_11_RX_ANTENNA_SELECTED		X	X			
0D01020D	OID_802_11_TX_ANTENNA_SELECTED		X	X			
0D01020E	OID_802_11_SUPPORTED_RATES		X		X	X	
0D010210	OID_802_11_DESIRED_RATES		X	X			
0D010211	OID_802_11_CONFIGURATION		X	X	X	X	
0D020212	OID_802_11_STATISTICS		X		Recommended	Recommended	Recommended
0D010113	OID_802_11_ADD_WEP			X	X	X	X
0D010114	OID_802_11_REMOVE_WEP			X	X	X	Recommended

© 2003 Microsoft Corporation. All rights reserved.

OID (Hex)	OID Name	Indication	Query	Set	Mandatory for Windows XP	Mandatory for WPA	Mandatory for Windows 2000, NT 4.0, 98/Me
0D010115	OID_802_11_DISASSOCIATE			X	X	X	
0D010216	OID_802_11_POWER_MODE		X	X	Recommended	Recommended	Recommended
0D010217	OID_802_11_BSSID_LIST		X		X	X	Recommended
0D010118	OID_802_11_AUTHENTICATION_MODE		X	X	X	X	Recommended
0D010119	OID_802_11_PRIVACY_FILTER		X	X			
0D01011A	OID_802_11_BSSID_LIST_SCAN			X	X	X	Recommended
0D01011B	OID_802_11_ENCRYPTION_STATUS		X	X	X	X	Recommended
0D01011C	OID_802_11_RELOAD_DEFAULTS			X	X	X	Recommended
0D01011D	OID_802_11_ADD_KEY					X	
0D01011E	OID_802_11_REMOVE_KEY					X	
0D01011F	OID_802_11_ASSOCIATION_INFORMATION					X	
0D010120	OID_802_11_TEST					X	

WPA supports and requires support for all the mandatory OIDS and the following WPA OIDS:

- OID_802_11_ADD_KEY
- OID_802_11_REMOVE_KEY
- OID_802_11_ASSOCIATION_INFORMATION
- OID_802_11_TEST
- OID_802_11_ENCRYPTION_STATUS (note same as mandatory OID OID_802_11_WEP_STATUS)
- OID_802_11_NETWORK_TYPES_SUPPORTED

Windows XP supports and requires support for all the following mandatory OIDs:

- OID_802_11_BSSID
- OID_802_11_SSID
- OID_802_11_ADD_WEP
- OID_802_11_REMOVE_WEP
- OID_802_11_NETWORK_TYPES_IN_USE
- OID_802_11_RSSI
- OID_802_11_INFRASTRUCTURE_MODE
- OID_802_11_SUPPORTED_RATES
- OID_802_11_CONFIGURATION
- OID_802_11_DISASSOCIATE
- OID_802_11_BSSID_LIST
- OID_802_11_BSSID_LIST_SCAN
- OID_802_11_AUTHENICATION_MODE

© 2003 Microsoft Corporation. All rights reserved.

- OID_802_11_WEP_STATUS
- OID_802_11_RELOAD_DEFAULTS

Windows 2000, Windows NT® 4.0, and Windows 98/Windows Millennium Edition require support for the following OIDs:

- OID_802_11_BSSID
- OID_802_11_SSID
- OID_802_11_ADD_KEY

It is recommended that the OID_802_11_STATISTICS, OID_802_11_POWER_MODE and OID_802_11_NETWORK_TYPES_SUPPORTED are implemented on all platforms.

IEEE 802.1X Support

To support IEEE 802.1X authentication, a driver must support the mandatory OIDs and enumerate itself as a wireless device. (See the topic, Sub-Media Type.). For non-WPA networks, 802.1X messages must not be encrypted, regardless of whether encryption is enabled or disabled on the NIC. For WPA networks, 802.1X messages must be encrypted with a pairwise key, but must not be encrypted with a group key.

Device Requirements

The device must conform to Plug and Play standards.

All driver and support software should install at device enumeration time. The device must not require a setup.exe or .dll to install.

The driver should update the firmware if the firmware does not support the driver version. If firmware is required to be flashed, this must be done during installation. Firmware can only be updated by updating the driver.

Cards should be 32-bit card-bus capable in preference to 16-bit PCMCIA capable. Programmed I/O should not be implemented.

IEEE 802.11a-capable cards that support USB should implement USB v 2.0.

IEEE 802.11a/b Radios

A dual-mode NIC may choose to implement the NIC as two independent “sub NICs” on a NIC. In this case, two drivers should be loaded, and the two sub NICs must work independently of each other. That is, each should be able to associate and transfer data independently of the other.

A dual-mode NIC can be implemented as a single NIC that associates either on IEEE 802.11a or IEEE 802.11b, but is not able to associate on IEEE 802.11a and IEEE 802.11b radios at the same time (referred to as an IEEE 802.11a- and IEEE 802.11b-capable NIC). In this case, one driver is loaded and the OID_802_11_BSSID_LIST must return the list of APs from both radios. The NIC can do a background scan across all channels and cache the results to have a full scan list available and should scan the most frequently used channels first. If an AP uses the same BSSID for multiple radios (for example, 2.4 GHz and 5 GHz), then an entry should be made in the OID_802_11_BSSID_LIST for each radio.

© 2003 Microsoft Corporation. All rights reserved.

OID OID_802_11_NETWORK_TYPES_SUPPORTED should return all the radios that the NIC supports.

The following OIDs must query or set the values for the current PHY that would be returned by OID_802_11_NETWORK_TYPE_IN_USE:

- OID_802_11_RSSI
- OID_802_11_SUPPORTED_RATES
- OID_802_11_CONFIGURATION

The following OIDs are not mandatory, but should also implement query/set per PHY:

- OID_802_11_TX_POWER_LEVEL
- OID_802_11_NUMBER_OF_ANTENNAS
- OID_802_11_RX_ANTENNA_SELECTED
- OID_802_11_TX_ANTENNA_SELECTED
- OID_802_11_DESIRED_RATES

IEEE 802.11 Data Types

This section lists the types that are used with the 802.11 OIDs:

```
//
// IEEE 802.11 Structures and definitions
//

typedef enum _NDIS_802_11_NETWORK_TYPE
{
    Ndis802_11FH,
    Ndis802_11DS,
    Ndis802_11OFDM5,
    Ndis802_11OFDM24,
    Ndis802_11NetworkTypeMax    // not a real type, defined
as an upper bound
} NDIS_802_11_NETWORK_TYPE, *PNDIS_802_11_NETWORK_TYPE;

typedef struct _NDIS_802_11_NETWORK_TYPE_LIST
{
    ULONG                       NumberOfItems;  // in list
below, at least 1
    NDIS_802_11_NETWORK_TYPE    NetworkType [1];
} NDIS_802_11_NETWORK_TYPE_LIST,
*PNDIS_802_11_NETWORK_TYPE_LIST;

typedef enum _NDIS_802_11_POWER_MODE
{
    Ndis802_11PowerModeCAM,
    Ndis802_11PowerModeMAX_PSP,
    Ndis802_11PowerModeFast_PSP,
    Ndis802_11PowerModeMax      // not a real mode, defined
as an upper bound
} NDIS_802_11_POWER_MODE, *PNDIS_802_11_POWER_MODE;

typedef ULONG   NDIS_802_11_TX_POWER_LEVEL; // in milliwatts

//
```

© 2003 Microsoft Corporation. All rights reserved.

```
// Received Signal Strength Indication
//
typedef LONG   NDIS_802_11_RSSI;           // in dBm

typedef struct _NDIS_802_11_CONFIGURATION_FH
{
    ULONG           Length;            // Length of structure
    ULONG           HopPattern;        // As defined by
                                       //802.11, MSB set
    ULONG           HopSet;            // to one if
                                       //non-802.11
    ULONG           DwellTime;         // units are Kusec
} NDIS_802_11_CONFIGURATION_FH,
*PNDIS_802_11_CONFIGURATION_FH;

typedef struct _NDIS_802_11_CONFIGURATION
{
    ULONG           Length;             // Length of
                                        // structure
    ULONG           BeaconPeriod;       // units are Kusec
    ULONG           ATIMWindow;         // units are Kusec
    ULONG           DSConfig;           // Frequency, units
                                        // are kHz
    NDIS_802_11_CONFIGURATION_FH    FHConfig;
} NDIS_802_11_CONFIGURATION, *PNDIS_802_11_CONFIGURATION;

typedef struct _NDIS_802_11_STATISTICS
{
    ULONG           Length;             // Length of
                                        // structure
    LARGE_INTEGER   TransmittedFragmentCount;
    LARGE_INTEGER   MulticastTransmittedFrameCount;
    LARGE_INTEGER   FailedCount;
    LARGE_INTEGER   RetryCount;
    LARGE_INTEGER   MultipleRetryCount;
    LARGE_INTEGER   RTSSuccessCount;
    LARGE_INTEGER   RTSFailureCount;
    LARGE_INTEGER   ACKFailureCount;
    LARGE_INTEGER   FrameDuplicateCount;
    LARGE_INTEGER   ReceivedFragmentCount;
    LARGE_INTEGER   MulticastReceivedFrameCount;
    LARGE_INTEGER   FCSErrorCount;
} NDIS_802_11_STATISTICS, *PNDIS_802_11_STATISTICS;

typedef  ULONG  NDIS_802_11_KEY_INDEX;

typedef struct _NDIS_802_11_WEP
{
    ULONG Length;
    ULONG KeyIndex;
    ULONG KeyLength;
    UCHAR KeyMaterial [1];
} NDIS_802_11_WEP, *PNDIS_802_11_WEP;

typedef ULONGULONG NDIS_802_11_KEY_RSC;

typedef struct _NDIS_802_11_KEY
```

© 2003 Microsoft Corporation. All rights reserved.

```
{
    ULONG     Length;        // Length of this structure
    ULONG     KeyIndex;
    ULONG     KeyLength;     // length of key in bytes
    NDIS_802_11_MAC_ADDRESS BSSID;
    NDIS_802_11_KEY_RSC KeyRSC;
    UCHAR     KeyMaterial[1];// variable length depending on
                             // above field
} NDIS_802_11_KEY, *PNDIS_802_11_KEY;

typedef struct _NDIS_802_11_REMOVE_KEY
{
    ULONG     Length;        // Length of this structure
    ULONG     KeyIndex;
    NDIS_802_11_MAC_ADDRESS BSSID;
} NDIS_802_11_REMOVE_KEY, *PNDIS_802_11_REMOVE_KEY;

typedef enum _NDIS_802_11_NETWORK_INFRASTRUCTURE
{
    Ndis802_11IBSS,
    Ndis802_11Infrastructure,
    Ndis802_11AutoUnknown,
    Ndis802_11InfrastructureMax     // Not a real value,
                                    // defined as upper bound
} NDIS_802_11_NETWORK_INFRASTRUCTURE,
*PNDIS_802_11_NETWORK_INFRASTRUCTURE;

typedef enum _NDIS_802_11_AUTHENTICATION_MODE
{
    Ndis802_11AuthModeOpen,
    Ndis802_11AuthModeShared,
    Ndis802_11AuthModeAutoSwitch,
    Ndis802_11AuthModeWPA,
    Ndis802_11AuthModeWPAPSK,
    Ndis802_11AuthModeWPANone,
    Ndis802_11AuthModeMax           // Not a real mode,
                                    // defined as upper bound
} NDIS_802_11_AUTHENTICATION_MODE,
*PNDIS_802_11_AUTHENTICATION_MODE;

typedef  UCHAR   NDIS_802_11_RATES[8];  // Set of 8 data
rates

typedef  UCHAR   NDIS_802_11_RATES_EX[16];  // Set of 16
data rates

typedef UCHAR   NDIS_802_11_MAC_ADDRESS[6];

typedef struct _NDIS_802_11_SSID
{
    ULONG   SsidLength;         // length of SSID field
                                // below, in bytes;
                                // this can be zero.
    UCHAR   Ssid[32];           // SSID information field
} NDIS_802_11_SSID, *PNDIS_802_11_SSID;

typedef struct _NDIS_WLAN_BSSID_EX
```

© 2003 Microsoft Corporation. All rights reserved.

```
{
    ULONG                                   Length;
                                // Length of this structure
    NDIS_802_11_MAC_ADDRESS                 MacAddress;
                                // BSSID
    UCHAR                                   Reserved[2];
    NDIS_802_11_SSID                        Ssid;
                                // SSID
    ULONG                                   Privacy;
                                // WEP encryption requirement
    NDIS_802_11_RSSI                        Rssi;
                                // receive signal
                                // strength in dBm
    NDIS_802_11_NETWORK_TYPE                NetworkTypeInUse;
    NDIS_802_11_CONFIGURATION               Configuration;
    NDIS_802_11_NETWORK_INFRASTRUCTURE      InfrastructureMode;
    NDIS_802_11_RATES_EX                    SupportedRates;
    ULONG                                   IELength;
    UCHAR                                   IEs[1];
} NDIS_WLAN_BSSID_EX, *PNDIS_WLAN_BSSID_EX;

typedef struct _NDIS_802_11_FIXED_IEs
{
      UCHAR Timestamp[8];
      USHORT BeaconInterval;
      USHORT Capabilities;
} NDIS_802_11_FIXED_IEs, *PNDIS_802_11_FIXED_IEs;

typedef struct _NDIS_802_11_VARIABLE_IEs
{
      UCHAR ElementID;
      UCHAR Length;     // Number of bytes in data field
      UCHAR data[1];
} NDIS_802_11_VARIABLE_IEs, *PNDIS_802_11_VARIABLE_IEs;

typedef struct _NDIS_802_11_BSSID_LIST_EX
{
    ULONG           NumberOfItems;      // in list below, at
                                        // least 1
    NDIS_WLAN_BSSID_EX Bssid[1];
} NDIS_802_11_BSSID_LIST_EX, *PNDIS_802_11_BSSID_LIST_EX;

typedef  ULONG   NDIS_802_11_FRAGMENTATION_THRESHOLD;

typedef  ULONG   NDIS_802_11_RTS_THRESHOLD;

typedef  ULONG   NDIS_802_11_ANTENNA;

typedef enum _NDIS_802_11_PRIVACY_FILTER
{
    Ndis802_11PrivFilterAcceptAll,
    Ndis802_11PrivFilter8021xWEP
} NDIS_802_11_PRIVACY_FILTER, *PNDIS_802_11_PRIVACY_FILTER;

typedef enum _NDIS_802_11_WEP_STATUS
{
    Ndis802_11WEPEnabled,
```

© 2003 Microsoft Corporation. All rights reserved.

```
    Ndis802_11Encryption1Enabled = Ndis802_11WEPEnabled,
    Ndis802_11WEPDisabled,
    Ndis802_11EncryptionDisabled = Ndis802_11WEPDisabled,
    Ndis802_11WEPKeyAbsent,
    Ndis802_11Encryption1KeyAbsent = Ndis802_11WEPKeyAbsent,
    Ndis802_11WEPNotSupported,
    Ndis802_11EncryptionNotSupported = Ndis802_11WEPNotSupported,
    Ndis802_11Encryption2Enabled,
    Ndis802_11Encryption2KeyAbsent,
    Ndis802_11Encryption3Enabled,
    Ndis802_11Encryption3KeyAbsent
} NDIS_802_11_WEP_STATUS, *PNDIS_802_11_WEP_STATUS,
NDIS_802_11_ENCRYPTION_STATUS,
*PNDIS_802_11_ENCRYPTION_STATUS;

typedef enum _NDIS_802_11_RELOAD_DEFAULTS
{
    Ndis802_11ReloadWEPKeys
} NDIS_802_11_RELOAD_DEFAULTS, *PNDIS_802_11_RELOAD_DEFAULTS;

#define NDIS_802_11_AI_REQFI_CAPABILITIES      1
#define NDIS_802_11_AI_REQFI_LISTENINTERVAL    2
#define NDIS_802_11_AI_REQFI_CURRENTAPADDRESS  4

#define NDIS_802_11_AI_RESFI_CAPABILITIES      1
#define NDIS_802_11_AI_RESFI_STATUSCODE        2
#define NDIS_802_11_AI_RESFI_ASSOCIATIONID     4

typedef struct _NDIS_802_11_ASSOCIATION_INFORMATION
{
    ULONG Length;
    USHORT AvailableRequestFixedIEs;
    struct _NDIS_802_11_AI_REQFI {
            USHORT Capabilities;
            USHORT ListenInterval;
                NDIS_802_11_MAC_ADDRESS  CurrentAPAddress;
    } RequestFixedIEs;
    ULONG RequestIELength;
    ULONG OffsetRequestIEs;
    USHORT AvailableResponseFixedIEs;
    struct _NDIS_802_11_AI_RESFI {
            USHORT Capabilities;
            USHORT StatusCode;
            USHORT AssociationId;
    } ResponseFixedIEs;
    ULONG ResponseIELength;
    ULONG OffsetResponseIEs;
} NDIS_802_11_ASSOCIATION_INFORMATION,
*PNDIS_802_11_ASSOCIATION_INFORMATION;

typedef struct _NDIS_802_11_TEST
{
 ULONG Length;
 ULONG Type;
 union {
      struct _AuthenticationEvent {
            NDIS_802_11_STATUS_TYPE StatusType;
```

© 2003 Microsoft Corporation. All rights reserved.

```
            NDIS_802_11_AUTHENTICATION_REQUEST Request[1];
      } AuthenticationEvent;
      NDIS_802_11_RSSI RssiTrigger;
 };
} NDIS_802_11_TEST, *PNDIS_802_11_TEST;
```

IEEE 802.11 OID Descriptions

This section lists IEEE 802.11 OID descriptions.

OID_802_11_BSSID

This object is the MAC address of the associated AP. Background scans must not affect the MAC address returned. This setting is useful when doing a site survey.

Data type: NDIS_802_11_MAC_ADDRESS.
Query: Returns the current AP MAC address.
Set: Sets the MAC address of the desired AP.
Indication: Not supported.
Errors: Returns an error code of NDIS_STATUS_ADAPTER_NOT_READY if the NIC is not associated with an access point. Return IBSS MAC address if in ad hoc mode.

OID_802_11_SSID

The OID selects the SSID that the driver should associate with. If the SSID is available on multiple radios (for example, IEEE 802.11a, IEEE 802.11b, and IEEE 802.11g), then the driver should decide which radio to use unless OID_802_11_NETWORK_TYPE_IN_USE has not been used to set the PHY. Two criteria that the driver may use to decide which AP to associate with are signal strength and PC battery life.

This object defines the SSID. The SSID is a string of up to 32 characters long. It identifies a set of interconnected basic service sets. Passing in an empty string means it can associate with any SSID. Setting an SSID results in the following conditions:

- Disassociating if already associated with a particular SSID.
- Turning on the radio if the radio is currently in the off state.
- Setting the SSID with the value specified, or setting it to any SSID if SSID is not specified.
- Attempting to associate with the set SSID.
- If the NIC associates with a different AP with the same SSID as the one already in use by the NIC, then the driver should generate a media connect event.

Data type: NDIS_802_11_SSID.
Query: Returns the SSID with which the NIC is associated. The driver returns 0 SSIDLength if the NIC is not associated with any SSID.
Set: Sets the SSID to a specified value.
Indication: Not supported.

This OID uses the NDIS_802_11_SSID structure that is defined as follows:

```
typedef UCHAR    NDIS_802_11_SSID;
```

© 2003 Microsoft Corporation. All rights reserved.

```
{
  ULONG      SsidLength;
  UCHAR      Ssid[32];
} NSID_802_11_SSID, *PNDIS_802_11_SSID;
```

The members of this structure contain the following information:

SsidLength
Specifies the length of the Ssid member in octets.

Ssid
Specifies the SSID. An empty string (that is, a string with the first byte set to zero) requests the 802.11 NIC to associate with any available SSID.

OID_802_11_NETWORK_TYPES_SUPPORTED

The OID_802_11_NETWORK_TYPES_SUPPORTED OID requests the miniport driver to return an array of all physical layer network subtypes that the IEEE 802.11 NIC and the driver support.

Data type: NDIS_802_11_NETWORK_TYPE_LIST.
Query: Returns an array of all NDIS_802_11_NETWORK_TYPE(s) supported by the driver and the device.
Set: Not supported.
Indication: Not supported.
Errors: Returns an error code of NDIS_STATUS_NOT_SUPPORTED if the NIC does not support this OID.

This OID uses the NDIS_802_11_NETWORK_TYPE_LIST structure that is defined as follows:

```
typedef struct _NDIS_802_11_NETWORK_TYPE_LIST {
  ULONG  NumberOfItems;
  NDIS_802_11_NETWORK_TYPE NetworkType [1];
} NDIS_802_11_NETWORK_TYPE_LIST,
*PNDIS_802_11_NETWORK_TYPE_LIST;
```

The members of this structure contain the following information:

NumberOfItems
Specifies the number of items in the **NetworkType** array. This array must contain at least one item.

NetworkType
Specifies an array of physical layer network subtypes. The NDIS_802_11_NETWORK_TYPE enumeration defines values to assign to the physical layer network subtypes.

The following network subtypes are defined:

Ndis802_11FH
Indicates the physical layer of the frequency-hopping, spread-spectrum radio.

Ndis802_11DS
Indicates the physical layer for the direct-sequencing, spread-spectrum radio.

Ndis802_11OFDM5
Indicates the physical layer for 5 GHz orthogonal frequency-division multiplexing (OFDM) radios.

Ndis802_11OFDM24
Indicates the physical layer for 2.4 GHz OFDM radios.

© 2003 Microsoft Corporation. All rights reserved.

OID_802_11_NETWORK_TYPE_IN_USE

The NDIS_802_11_NETWORK_TYPE enumeration specifies network subtype values for the physical layer.

Data type: NDIS_802_11_NETWORK_TYPE.
Query: Returns the current NDIS_802_11_NETWORK_TYPE used by the device.
Set: Sets the network type that should be used for the driver.
Indication: Not supported.
Errors: Returns an error code of NDIS_STATUS_INVALID_DATA if an invalid value is used.

The network subtypes are defined as follows:

Ndis802_11FH
Indicates the physical layer of the frequency-hopping, spread-spectrum radio.

Ndis802_11DS
Indicates the physical layer of the direct-sequencing, spread-spectrum radio.

Ndis802_11OFDM5
Indicates the physical layer for 5 GHz OFDM radios.

Ndis802_11OFDM24
Indicates the physical layer for 2.4 GHz OFDM radios.

OID_802_11_TX_POWER_LEVEL

Transmit power level in milliwatts (mW).

Data type: NDIS_802_11_TX_POWER_LEVEL.
Query: Returns the current NDIS_802_11_TX_POWER_LEVEL value in mW.
Set: Sets the current NDIS_802_11_TX_POWER_LEVEL value in mW.
Indication: Not supported.
Errors: Returns an error code of NDIS_STATUS_NOT_SUPPORTED if the NIC does not support this OID.
Returns an error code of NDIS_STATUS_INVALID_DATA if a non-regulatory value is used.

The NDIS_802_11_TX_POWER_LEVEL value is defined as follows:

```
typedef ULONG   NDIS_802_11_TX_POWER_LEVEL;
```

Note: The IEEE 802.11 NIC will not exceed regulatory power limits.

OID_802_11_RSSI

Requests the miniport driver to return the Received Signal Strength Indication (RSSI) in response to a query or as a status indication event. The RSSI is measured in decibel milliwatts (dBm).

Data type: NDIS_802_11_RSSI.
Query: Returns the current RSSI value.
Set: Not supported.
Indication: If an indication request is enabled, then an event is triggered according to the value as given in the set.

The NDIS_802_11_RSSI is defined as follows:

© 2003 Microsoft Corporation. All rights reserved.

```
typedef LONG  NDIS_802_11_RSSI;
```

If status indications are enabled with OID_802_11_RSSI_TRIGGER, the miniport driver must call **NdisMIndicateStatus** to provide notification values according to the RSSI that was specified in the OID_802_11_RSSI_TRIGGER set.

Typical values for the RSSI value are between -10 and -200.

OID_802_11_RSSI_TRIGGER

Queries or sets a trigger value for the RSSI event. If the trigger value is less than the current RSSI value, then the indication occurs when the current value is greater than or equal to the trigger value. If the trigger value is greater than the current value, then the indication occurs when the current value is less than or equal to the trigger value. If the trigger value is equal to the current value, then the indication occurs immediately. **NdisMIndicateStatus** is called with NDIS_STATUS_MEDIA_SPECIFIC_INDICATION as the *GeneralStatus* and the *StatusBuffer* points to a NDIS_802_1_RSSI buffer.

Data type:	NDIS_802_11_RSSI.
Query:	Returns the current RSSI trigger value.
Set:	Sets the RSSI trigger value for an event.
Indication:	Not supported.
Errors:	Returns an error code of NDIS_STATUS_NOT_SUPPORTED if the NIC does not support this OID. Returns an error code of NDIS_STATUS_INVALID_DATA if an invalid RSSI value is used.

The trigger value contains the RSSI measurement in units of dBm and is defined as follows:

```
typedef LONG  NDIS_802_11_RSSI;
```

When a status indication triggers, the miniport driver calls **NdisMIndicateStatus** with the *GeneralStatus* parameter set to NDIS_STATUS_MEDIA_SPECIFIC_INDICATION. In addition, the *StatusBuffer* passed to **NdisMIndicateStatus** points to an NDIS_802_11_RSSI buffer. Typical values for the RSSI are between -10 and -200 and must be negative.

If the trigger value is less than the current RSSI value, the status indication is triggered when the current value is greater than or equal to the trigger value. If the trigger value is greater than the current value, the indication is triggered when the current value is less than or equal to the trigger value. If the trigger value is equal to the current value, the indication is triggered immediately.

OID_802_11_INFRASTRUCTURE_MODE

Queries or sets how an IEEE 802.11 NIC connects to the network. This also resets the network association algorithm.

Data type:	NDIS_802_11_NETWORK_INFRASTRUCTURE.
Query:	Returns either Infrastructure or IBSS, unknown.
Set:	Sets mode to Infrastructure or IBSS, or automatic switch between the two.
Indication:	Not supported.
Errors:	Returns an error code of NDIS_STATUS_INVALID_DATA if an invalid mode value is used.

© 2003 Microsoft Corporation. All rights reserved.

This OID uses the NDIS_802_11_NETWORK_INFRASTRUCTURE enumeration that is defined as follows:

Ndis802_11IBSS
Specifies the IBSS mode. This mode is also known as ad hoc mode.

Ndis802_11Infrastructure
Specifies the infrastructure mode.

Ndis802_11AutoUnknown
Specifies an automatic mode. In this mode, the IEEE 802.11 NIC can switch between ad hoc and infrastructure modes as required. Using this setting is not recommended.

Note: When this OID is called to set the mode, all keys set through OID_802_11_ADD_WEP and OID_802_11_ADD_KEY should be deleted.

OID_802_11_FRAGMENTATION_THRESHOLD

Packets that are larger than the fragmentation threshold are fragmented before they are transmitted. Packets that are smaller than the specified fragmentation threshold value are not fragmented. A fragmentation threshold of zero indicates that the NIC should not fragment packets.

Data type:	NDIS_802_11_FRAGMENTATION_THRESHOLD.
Query:	Returns the current fragmentation threshold in bytes.
Set:	Sets the fragmentation threshold in bytes.
Indication:	Not supported.
Errors:	Returns an error code of NDIS_STATUS_NOT_SUPPORTED if the NIC does not support this OID. Returns an error code of NDIS_STATUS_INVALID_DATA if an invalid fragmentation value is used.

The fragmentation threshold is defined as follows:

```
typedef  ULONG   NDIS_802_11_FRAGMENTATION_THRESHOLD;
The minimum threshold value is 256 and the maximum is 2346.
```

OID_802_11_RTS_THRESHOLD

This value, which is specified in bytes, specifies the packet size beyond which the IEEE 802.11 wireless LAN invokes its Request to Send/Clear to Send (RTS/CTS) mechanism. Packets that exceed the specified RTS threshold trigger the RTS/CTS mechanism. The NIC transmits smaller packets without using RTS/CTS. An RTS threshold value of zero indicates that the NIC should transmit all packets using RTS/CTS.

Data type:	NDIS_802_11_RTS_THRESHOLD.
Query:	Returns the current RTS threshold.
Set:	Sets the RTS threshold.
Indication:	Not supported.
Errors:	Returns an error code of NDIS_STATUS_NOT_SUPPORTED if the NIC does not support this OID. Returns an error code of NDIS_STATUS_INVALID_DATA if an invalid RTS value is used.

The RTS threshold is defined as follows:

© 2003 Microsoft Corporation. All rights reserved.

```
typedef  ULONG   NDIS_802_11_RTS_THRESHOLD;
The minimum RTS threshold value is 0 and the maximum is 2347.
```

OID_802_11_NUMBER_OF_ANTENNAS

Returns the number of antennas on the radio.

Data type: ULONG.
Query: Returns the number of antennas on the radio.
Set: Not supported.
Indication: Not supported.
Errors: Returns an error code of NDIS_STATUS_NOT_SUPPORTED if the NIC does not support this OID.

OID_802_11_RX_ANTENNA_SELECTED

Returns the antenna selected for receiving on the radio.

Data type: NDIS_802_11_ANTENNA.
Query: Returns the antenna currently selected for receiving.
Set: Sets the antenna used for receiving.
Indication: Not supported.
Errors: Returns an error code of NDIS_STATUS_NOT_SUPPORTED if the NIC does not support this OID.
Returns an error code of NDIS_STATUS_INVALID_DATA if an invalid value is used.

Note: -1 (0xffffffff) indicates that all antennas are selected. That is, the LAN is in a state of “diversity.”

The antenna number is defined as follows:

```
typedef  ULONG   NDIS_802_11_ANTENNA;
The antenna number starts at 0.
```

OID_802_11_TX_ANTENNA_SELECTED

Returns the antenna selected for transmitting on the radio.

Data type: NDIS_802_11_ANTENNA.
Query: Returns the antenna currently selected for transmitting.
Set: Sets the antenna used for transmitting.
Indication: Not supported.
Errors: Returns an error code of NDIS_STATUS_NOT_SUPPORTED if the NIC does not support this OID.
Returns an error code of NDIS_STATUS_INVALID_DATA if an invalid value is used.

Note: -1 (0xffffffff) indicates that all antennas are selected. That is, the LAN is in a state of “diversity.”

The antenna value is defined as follows:

```
typedef  ULONG   NDIS_802_11_ANTENNA;
The antenna number starts at 0.
```

© 2003 Microsoft Corporation. All rights reserved.

OID_802_11_SUPPORTED_RATES

The OID_802_11_SUPPORTED_RATES OID requests that the miniport driver return its NICs data rate set, which includes the data rates that the NIC's radio supports. Data rates are encoded as eight octets, where each octet describes a single supported rate in units of 0.5 Mbps. Supported rates belonging to the BSSBasicRateSet are used for frames such as control and broadcast frames. Each supported rate belonging to the BSSBasicRateSet is encoded as an octet with the most significant bit (bit 7) set to 1 (for example, a 1-Mbps rate belonging to the BSSBasicRateSet is encoded as 0x82). Rates not belonging to the BSSBasicRateSet are encoded with the most significant bit set to 0 (for example, a 2-Mbps rate not belonging to the BSSBasicRateSet is encoded as 0x04).

Data type: NDIS_802_11_RATES.

Query: Returns the set of supported data rates that the radio is capable of running.

Set: Not supported.

Indication: Not supported.

This OID uses the NDIS_802_11_RATES array that is defined as follows:

```
typedef  UCHAR   NDIS_802_11_RATES[8];
```

The rates array contains a set of eight octets. Each octet contains a preferred data rate in units of 0.5 Mbps. Any unused entries left at the end of the array should be set to zero.

OID_802_11_DESIRED_RATES

A set of preferred data rates for the radio to operate. Data rates are encoded as eight octets, where each octet describes a single rate in units of 0.5 Mbps. Frames that are directed to the radio can run at a different value from the supported rates belonging to the BSSBasicRateSet.

Data type: NDIS_802_11_RATES.

Query: Returns the set of data rates.

Set: Sets the set of data rates.

Indication: Not supported.

Errors: Returns an error code of NDIS_STATUS_NOT_SUPPORTED if the NIC does not support this OID.

Returns an error code of NDIS_STATUS_INVALID_DATA if any rates set in the array are invalid. If any rates are invalid, no rates should be set.

This OID uses the NDIS_802_11_RATES array. This array is defined as follows:

```
typedef  UCHAR   NDIS_802_11_RATES[8];
```

This array contains a set of eight octets. Each octet specifies a preferred data rate in units of 0.5 Mbps. Any unused entries left at the end of the array should be set to 0. The IEEE 802.11 network could transmit packets at rates other than the basic rates. The NIC can receive data at rates other than the rates included in the BSSBasicRateSet.

© 2003 Microsoft Corporation. All rights reserved.

OID_802_11_CONFIGURATION

Configures the radio parameters.

Data type: NDIS_802_11_CONFIGURATION.
Query: Returns the current radio configuration.
Set: Sets the radio configuration.
Indication: Not supported.
Errors: Returns an error code of NDIS_STATUS_INVALID_DATA if any of the data elements are invalid.

This OID should only be set when the device is not associated with an AP. This OID uses an NDIS_802_11_CONFIGURATION structure, which describes the configuration of a radio. This structure is defined as follows:

```
typedef struct _NDIS_802_11_CONFIGURATION {
 ULONG Length;
 ULONG BeaconPeriod;
 ULONG ATIMWindow;
 ULONG DSConfig;
 NDIS_802_11_CONFIGURATION_FH FHConfig;
} NDIS_802_11_CONFIGURATION, *PNDIS_802_11_CONFIGURATION;
```

The members of this structure contain the following information:

Length
Specifies the length of the NDIS_802_11_CONFIGURATION structure in bytes.

BeaconPeriod
Specifies the interval between beacon message transmissions. This value is specified in kilomicroseconds (kµsec or 1024 µsec). On query, this should return the current beacon period of the associated AP or if in IBSS mode the IBSS beacon period or zero, if not associated and not in IBSS mode. On set, this specifies the beacon interval for IBSS mode; it is ignored in Extended Service Set (ESS) mode.

ATIMWindow
Specifies the Announcement Traffic Information Message (ATIM) window in kµsec. The ATIM window is a short time period immediately after the transmission of each beacon in an IBSS configuration. During the ATIM window, any station can indicate the need to transfer data to another station during the following data-transmission window.

DSConfig
Specifies the frequency of the selected channel in kHz. On set, it specifies the frequency for IBSS and is ignored in ESS mode. On query, it should return the current radio frequency.

Note: For 2.4 GHz DSSS, 2.4 GHz OFDM, and 5 GHz OFDM, the current frequency should be in DSConfig as in the IEEE 802.11a specification.

Valid ranges for DSConfig are

2,412,000 to 2,484,000 for 2.4 GHz

5,000,000 to 6,000,000 for 5 GHz

Note: In the United States, the valid subset for 5 GHz is 5,180,000 to 5,240,000, 5,260,000 to 5,320,000 and 5,745,000 to 5,805,000.

FHConfig
Specifies the frequency-hopping configuration in an NDIS_802_11_CONFIGURATION_FH structure.

© 2003 Microsoft Corporation. All rights reserved.

The NDIS_802_11_CONFIGURATION_FH structure is defined as follows:

```
typedef struct _NDIS_802_11_CONFIGURATION_FH {
 ULONG Length;
 ULONG HopPattern;
 ULONG HopSet;
 ULONG DwellTime;
} NDIS_802_11_CONFIGURATION_FH,
*PNDIS_802_11_CONFIGURATION_FH;
```

The members of this structure contain the following information:

Length
Specifies the length of the NDIS_802_11_CONFIGURATION_FH structure in bytes.

HopPattern
Specifies the hop pattern used to determine the hop sequence. As defined by the IEEE 802.11 standard, the layer management entity (LME) of the physical layer uses a hop pattern to determine the hop sequence.

HopSet
Specifies a set of patterns. The LME of the physical layer uses these patterns to determine the hop sequence.

DwellTime
Specifies the maximum period of time during which the transmitter should remain fixed on a channel. This interval is described in kµsec.

OID_802_11_STATISTICS

Requests the miniport driver to return the current statistics for the IEEE 802.11 interface.

Data type:	NDIS_802_11_STATISTICS.
Query:	Returns the current statistics.
Set:	Not supported.
Indication:	Not supported.
Errors:	Returns an error code of NDIS_STATUS_NOT_SUPPORTED if the NIC does not support this OID.

This OID is not mandatory, but its implementation is recommended. These statistics are described by an NDIS_802_11_STATISTICS structure, defined as follows:

```
typedef struct _NDIS_802_11_STATISTICS
{
 ULONG Length;
 LARGE_INTEGER TransmittedFragmentCount;
 LARGE_INTEGER MulticastTransmittedFrameCount;
 LARGE_INTEGER FailedCount;
 LARGE_INTEGER RetryCount;
 LARGE_INTEGER MultipleRetryCount;
 LARGE_INTEGER RTSSuccessCount;
 LARGE_INTEGER RTSFailureCount;
 LARGE_INTEGER ACKFailureCount;
 LARGE_INTEGER FrameDuplicateCount;
 LARGE_INTEGER ReceivedFragmentCount;
 LARGE_INTEGER MulticastReceivedFrameCount;
 LARGE_INTEGER FCSErrorCount;
} NDIS_802_11_STATISTICS, *PNDIS_802_11_STATISTICS;
```

© 2003 Microsoft Corporation. All rights reserved.

The members of this structure contain the following information:

Length
Specifies the length of the NDIS_802_11_STATISTICS structure in bytes.

TransmittedFragmentCount
Indicates the number of data and management fragments that the NIC has successfully transmitted.

MulticastTransmittedFrameCount
Indicates the number of frames that the NIC has transmitted by multicast or broadcast. This count is incremented each time that the multicast/broadcast bit is set in the destination MAC address of a transmitted frame.

FailedCount
Indicates the number of NIC frame transmissions that failed after exceeding either the short frame or the long frame retry limits.

RetryCount
Indicates the number of frames that the NIC successfully retransmitted after one or more retransmission attempts.

MultipleRetryCount
Indicates the number of frames that the NIC successfully retransmitted after more than one retransmission attempt.

RTSSuccessCount
Indicates the number of times that the NIC received a CTS in response to an RTS.

RTSFailureCount
Indicates the number of times that the NIC did not receive a CTS in response to an RTS.

ACKFailureCount
Indicates the number of times the NIC expected an ACK that was not received.

FrameDuplicateCount
Indicates the number of duplicate frames that were received. The sequence control field in the frame identifies duplicate frames.

ReceivedFragmentCount
Indicates the number of data and management fragments that the NIC has successfully received. This count is incremented each time that either a data fragment or a management fragment is received.

MulticastReceivedFrameCount
Indicates the number of received frames that were set to multicast or broadcast. This count is incremented each time the NIC receives a frame with the multicast/broadcast bit set in the destination MAC address.

FCSErrorCount
Indicates the number of frames the NIC received that contained FCS errors.

OID_802_11_ADD_WEP

The OID_802_11_ADD_WEP OID requests the miniport driver to set an IEEE 802.11 wired equivalent privacy (WEP) key to a specified value. A WEP key can be a preshared key (a key that is provided to the NICs before use) for authentication, encryption, or both.

Data type: NDIS_802_11_WEP.
Query: Not supported.
Set: Sets the desired WEP key.

© 2003 Microsoft Corporation. All rights reserved.

Indication: Not supported.

Errors: Returns an error code of NDIS_STATUS_INVALID_DATAFAILURE if the key cannot be set for any reason.

There are two types of WEP keys:

- Per-client key

  A per-client key is used by the client to send information to the AP.

- Global key

  A global key is used to send broadcast and multicast packets to all clients that are communicating with an AP. A global key can also be used to send unicast packets to and from the client. There must be a minimum of four global keys supported by the NIC.

A WEP key is described by the NDIS_802_11_WEP structure that is defined as follows:

```
typedef struct _NDIS_802_11_WEP
{
 ULONG Length;
 ULONG KeyIndex;
 ULONG KeyLength;
 UCHAR KeyMaterial [1];
} NDIS_802_11_WEP, *PNDIS_802_11_WEP;
```

The members of this structure contain the following information:

Length
Specifies the length of the NDIS_802_11_WEP structure in bytes.

KeyIndex
Specifies which key to add or remove. The global keys are represented by values of zero to *n*. When bit 31 is set to 1, it indicates the key used to transmit to the access point. When bit 30 is set to 1, it indicates that the key is a per-client key.

KeyLength
Specifies the length of the **KeyMaterial** character array in bytes.

KeyMaterial
Specifies an array that identifies the WEP key. The length of this array is variable and depends upon the value of the **KeyLength** member.

Notes:

When this OID sets the same index twice, the miniport driver should overwrite the previous WEP key at that index.

The underlying miniport driver and its NIC should not maintain WEP keys in permanent storage (disk, registry, flash or other). WEP keys should be discarded immediately after any of the following events:

- After the NIC disassociates itself from all BSSIDs.
- If shared key authentication using the WEP key fails.
- When the driver is unloaded.
- When the NIC is disabled.

© 2003 Microsoft Corporation. All rights reserved.

If the card does not support the key length, the OID should be failed. For example, if the card only supports 40-bit keys, an ADD_WEP for a 104-bit key should be failed.

Note: 802.1X messages should not be encrypted regardless of whether encryption is enabled or disabled on the NIC.

Within the set of available unicast and broadcast keys, at most only one key may be designated as the transmit key for a BSSID. Similarly, within the set of broadcast keys, at most only one key may be designated as the transmit key. This implies that the driver cannot maintain two transmit keys simultaneously. For example, having one unicast and one broadcast key set to transmit is not a valid configuration. Therefore, the last instantiated transmit WEP key must be effective, thus ensuring that there is at most only one transmit WEP key within the available unicast and broadcast keys.

Notes:

- When setting a per-client key, the BSSID for the key is not specified and should be assumed to be the access points BSSID. This means that a per-client key can only be set for infrastructure mode.
- A driver receiving an NDISReset should save the current state of the NIC, reset the device, and reconfigure the original state, including keys.

OID_802_11_ADD_KEY

The OID_802_11_ADD_KEY OID requests that the miniport driver set a key to a specified value. A key can be a preshared key (a key that is provided to the NICs before use) for authentication, encryption, or both.

Data type: NDIS_802_11_KEY
Query: Not supported.
Set: Sets the desired key.
Indication: Not supported.

There are two types of keys:

- Pairwise key

  A pairwise key is used by the client to send and receive unicast information to and from the AP or station.

- Group key

  A group key is used to send broadcast and multicast packets to all clients that are communicating with an AP. A group key can also be used to send unicast packets to and from the client if no pairwise keys have been set. There must be a minimum of four group keys supported by the NIC.

Note: Pairwise and group keys are the new terminology used by the WPA document. The driver and NIC needs to map these keys into their hardware/software implementation. One way to map them is to map pairwise keys to key mapping keys, and map group keys to default keys.

A key is described by the NDIS_802_11_KEY structure that is defined as follows:

```
typedef struct _NDIS_802_11_KEY
{
 ULONG Length;
```

© 2003 Microsoft Corporation. All rights reserved.

```
 ULONG KeyIndex;
 ULONG KeyLength;
 NDIS_802_11_MAC_ADDRESS BSSID;
 NDIS_802_11_KEY_RSC KeyRSC;
 UCHAR KeyMaterial [1];
} NDIS_802_11_KEY, *PNDIS_802_11_KEY;
```

The members of this structure contain the following information:

Length

Specifies the length of the NDIS_802_11_KEY structure in bytes.

KeyIndex

When bit 31 is set to 1, it indicates that the key used to transmit. Only one transmit key per BSSID is allowed for group keys. The last call of this OID to configure a Group key with the transmit bit set will be used as the transmit key. If a pairwise key has been set for a BSSID, the transmit flag for the Group keys will be ignored for unicast messages.

When bit 30 is set to 1, it indicates that the key is a pairwise key. Otherwise, it indicates that the key is a Group key.

Thus, if the first bit is the transmit/receive bit and the second bit is the pairwise/group bit, then:

00	Group key that is receive only.
10	Group key that can be used to transmit.
01	Illegal combination return NDIS_STATUS_INVALID_DATA.
11	Pairwise key that can be used to transmit.

In practice, transmit group keys will only be used when pairwise keys for transmission are not available.

When Bit 29 is set to 1, the KeyRSC should be used to set the initial receive SC for the key. When it is 0, the receive SC should be set by the NIC.
When Bit 28 is set to 1, the key is being set by an Authenticator. If the bit is set to zero, the key is being set by a Supplicant. It must be set to 0 in IBSS WPA mode.

Note: This bit is used to decide which part of the key material should be used for the transmit and receive Michael integrity keys.

Note: In the WPA IBSS case, the system above the OID will make sure that the Authenticator Tx key and Authenticator Rx key are the same.

The key index is specified in the lower 8 bits by values of zero to 255. The key index should be set to zero for a pairwise key: the driver should fail with NDIS_STATUS_INVALID_DATA if the lower 8 bits is not zero.

Remaining bits must be set to zero.

KeyLength

Specifies the length of the **KeyMaterial** character array in bytes. The maximum supported size of **KeyLength** is 32 bytes.

BSSID

This is the MAC address of the BSSID to which the key applies. The field should be set to 0xffffffffffff if the access point BSSID is unknown.

© 2003 Microsoft Corporation. All rights reserved.

Key Type	BSSID	IBSS/ ESS	Key Mapping Keys Supported	Behavior
Pairwise	0xffffffffffff	N/A	Yes	Fail with NDIS_STATUS_INVALID_DATA..
Pairwise	Not 0xffffffffffff	N/A	Yes	Configure immediately. If all pairwise keys are configured, delete a pairwise key that is not being used. That is, do not delete the pairwise key for the currently associated AP, but changing the pairwise key for the current associated AP is allowed. If all pairwise keys are in use, map as Group Key 0.
Pairwise	N/A	N/A	No	Map as group Key 0.
Group	0xffffffffffff	N/A	N/A	Configure immediately.
Group	Not 0xffffffffffff	IBSS	N/A	Fail with NDIS_STATUS_INVALID_DATA.
Group	Equal to current associated BSSID	ESS	N/A	Configure immediately.
Group	Not 0xffffffffffff and not equal to the current associated BSSID or not associated	ESS	N/A	Save key and configure just before association/reassociation to BSSID. When configured into the hardware, the saved copy should be deleted.

Note: If the NIC implementation cannot support the additional pairwise key, then it should delete a previous key that is not in current use. For example, if the NIC supports 10 pairwise keys and an eleventh pairwise key is added, one of the earlier 10 keys should be deleted. However, keys that are in current use (for example the pairwise key to the currently associated access point) should not be deleted. If no more pairwise keys can be supported but all pairwise keys are currently in use, then the pairwise key should be mapped as Group Key 0.

The number of supported BSSID sets of group keys should be the same as the number of pairwise keys supported, except if no pairwise keys are supported when at least five group key sets should be supported. A minimum of five sets of group keys should be supported with each set supporting at least four keys.

After a group key is configured into the encryption/integrity engine, the saved copy must be deleted and must not be reconfigured into the encryption/integrity engine later.

© 2003 Microsoft Corporation. All rights reserved.

KeyRSC

This field specifies the initial value of the Key Receive Sequence Counter, if bit 29 of the KeyIndex is 1. If the encryption cipher needs less bytes then supplied by KeyRSC, the least significant bytes should be used. For example, if 6 bytes are needed KeyRSC[0-5] should be used.

KeyMaterial

Specifies an array that identifies the key. The length of this array is variable and depends upon the value of the **KeyLength** member.

Notes:

When this OID sets the same index twice for the same key type and the same BSSID, the miniport driver should overwrite the previous key at that index.

The underlying miniport driver and its NIC should not maintain any keys set through OID_802_11_ADD_KEY in permanent storage (disk, registry, flash, or other). All keys set through this OID for any BSSID should be discarded immediately after any of the following events:

- After the NIC disassociates itself from all BSSIDs (changing from ad hoc or infrastructure and vice versa disassociates from all BSSIDs before attempting to join a BSSID).
- If shared key authentication using the key fails.
- When the driver is unloaded.
- When the NIC is disabled or reset.

A NIC should discard a pairwise key set via OID_802_11_ADD_KEY for a station or AP on receiving a Disassociate or Deauthenticate from the station or AP or on an associate/re-associate..

If the card does not support the key length for the encryption modes currently enabled or in use, the OID should be failed with NDIS_STATUS_INVALID_DATA. For example, if the card only supports 40-bit keys for the current enabled encryption modes, an ADD_KEY for a 104-bit or other key lengths should be failed. Also, if the NIC is associated and TKIP has been negotiated, then key lengths that are not valid should be failed.

If the card does not support pairwise keys the OID should map the pairwise key to Group Key 0. The key is still used to encrypt 802.1X packets as if it is a pairwise key. Group key 0 should be saved if the BSSID is not the current BSSID.

If TKIP is supported, the key length is 256 bits. The first (starting at the least significant bit) 128 bits must be used for the encryption key and the second 128 bits must be used for two MIC keys.

For TKIP keys, the first (starting at the least significant bit in the 128 bits) 64 bits of the MIC key is used in the direction from the Authenticator to the Supplicant if bit 28 is 1, and in the other direction if bit 28 is 0. The second 64 bits of the MIC key are used in the direction from the Supplicant to Authenticator if bit 28 is 1, and in the other direction if bit 28 is 0.

WPA auth	Bit 28	Bits 0 to 127 of TKIP key	Bits 128 to 191 of TKIP key	Bits 192 to 255 of TKIP key
WPA/ WPAPSK	0 – Supplicant	Encryption key	Rx MIC key	Tx MIC key
WPA/	1 - Authenticator	Encryption key	Tx MIC key	Rx MIC key

© 2003 Microsoft Corporation. All rights reserved.

WPAPSK				
WPANone	N/A	Encryption key	Tx/Rx MIC key	Not used

If AES is supported the key, length is 128 bits.

Note: A driver receiving an NDISReset should save the current state of the NIC, reset the device, and reconfigure the original state, including keys.

OID_802_11_REMOVE_KEY

The OID requests that the miniport driver remove the key at the specified key index for the current association. The key should not be held in permanent storage, but should be removed as soon as the card disassociates with all BSSIDs. The request should complete successfully if there was no key in the index as long as the key index is valid.

Data type: NDIS_802_11_REMOVE_KEY
Query: Not supported.
Set: Removes the desired key.
Indication: Not supported.
Errors: Returns an error code of NDIS_STATUS_INVALID_DATA if the index value is invalid.

The NDIS_802_11_REMOVE_KEY structure is defined as follows:

```
typedef struct _NDIS_802_11_REMOVE_KEY
{
    ULONG     Length;         // Length of this structure
    ULONG     KeyIndex;
    NDIS_802_11_MAC_ADDRESS BSSID;
} NDIS_802_11_REMOVE_KEY, *PNDIS_802_11_REMOVE_KEY;
```

Length
Specifies the length of the structure.

KeyIndex
Specifies which key to remove. The keys are represented by values of zero to 255. Bit 31 is set to 0 and NDIS_STATUS_INVALID_DATA should be returned if the bits are not 0. When bit 30 is set to 1, it indicates that the key is a pairwise key. Otherwise, it indicates that the key is a group key. If the transmit key is removed for a BSSID, there is no transmit key for the key type for a BSSID, until a transmit key is specified. Remaining bits must be set to 0 and NDIS_STATUS_INVALID_DATA should be returned if the bits are not 0.

BSSID
This is the MAC address of the BSSID to which the key applies. The field should be set to 0xffffffffffff if the access point BSSID is unknown.

© 2003 Microsoft Corporation. All rights reserved.

Key Type	BSSID	Pairwise Supported	Key Index	Behavior
Pairwise	0xffffffffffff	Yes	Ignored	Delete all pairwise keys.
Pairwise	Not 0xffffffffffff	Yes	Ignored	Delete pairwise key for this BSSID.
Pairwise	N/A	No	Ignored	Delete Group Key 0.
Group	0xffffffffffff	N/A	Required	Delete all group keys for key index.
Group	Not 0xffffffffffff	N/A	Required	Delete group key for this BSSID for key index.

OID_802_11_REMOVE_WEP

The OID requests that the miniport driver remove the key at the specified key index for the current association. The key should not be held in permanent storage, but should be removed as soon as the card disassociates with all BSSIDs.

Data type: NDIS_802_11_KEY_INDEX.
Query: Not supported.
Set: Removes the desired key.
Indication: Not supported.

The key index is defined as follows:

```
typedef  ULONG    NDIS_802_11_KEY_INDEX;
```

Note: Specifies which key to remove. The keys are represented by values of zero to 255. Bit 31 will be set to zero and NDIS_STATUS_INVALID_DATA should be returned if the bits are not zero.

OID_802_11_DISASSOCIATE

Requests that the miniport driver command its NIC to disassociate from the current service set and turn off the radio.

Data type: No data is associated with this Set.
Query: Not supported.
Set: Disassociates with the current SSID and turns off the radio.
Indication: Not supported.

Note: The driver should generate a media disconnected event on successful disassociation.

Note: The driver should return success NDIS_STATUS_SUCCESS when this OID is called but the NIC is disassociated.

In IBSS mode the driver should generate a media disconnected event and turn off the radio.

OID_802_11_POWER_MODE

Data type: NDIS_802_11_POWER_MODE.
Query: Returns the current NDIS_802_11_POWER_MODE.
Set: Sets the current NDIS_802_11_POWER_MODE.
Indication: Not supported.

© 2003 Microsoft Corporation. All rights reserved.

Errors: Returns an error code of NDIS_STATUS_NOT_SUPPORTED if the NIC does not support this OID.
Returns an error code of NDIS_STATUS_INVALID_DATA if an invalid value is used.

This OID is not mandatory, but its implementation is recommended. The power modes that can be assigned to the IEEE 802.11 NIC are defined in the NDIS_802_11_POWER_MODE enumeration as follows:

Ndis802_11PowerModeCAM
Specifies continuous access mode (CAM). When the power mode is set to CAM, the device is always on. If the NIC supports power saving, the NIC should associate to the AP with a listen interval specified so that the power mode state can be changed without requiring a reassociation.

Ndis802_11PowerModeMAX_PSP
Specifies the maximum (MAX) power saving. A power mode of MAX results in the greatest power savings for the IEEE 802.11 NIC radio.

Ndis802_11PowerModeFast_PSP
Specifies fast power-saving mode. This power mode should provide the best combination of network performance and power usage.

OID_802_11_BSSID_LIST

Returns the list of all BSSIDs detected including their attributes specified in the data structure. The list of BSSIDs returned is the cached list stored in the IEEE 802.11 NIC's database. The list of BSSIDs in the IEEE 802.11 NIC's database is the set of BSSs detected by the IEEE 802.11 NIC during the last survey of potential BSSs. A call to this OID should result in an immediate return of the list of BSSIDs in the IEEE 802.11 NIC's database.

Note that if this OID is called without a preceding OID_802_11_BSSID_LIST_SCAN and the IEEE 802.11 NIC is active, it may return a list of BSSIDs limited to those BSSIDs that the IEEE 802.11 NIC considers valid to join based on its current configuration. However, if this OID is immediately preceded by OID_802_11_BSSID_LIST_SCAN, then the list of BSSIDs should additionally contain any BSSIDs found during the scan.

For IEEE 802.11a- and IEEE 802.11b-capable NICs, this OID should return a complete list of all the IEEE 802.11a and IEEE 802.11b APs.

Data type: NDIS_802_11_BSSID_LIST_EX.
Query: Returns the NDIS_802_11_BSSID_LIST_EX structure.
Set: Not supported.
Indication: Not supported.

Errors: Returns an error code of NDIS_STATUS_INVALID_DATA if an invalid value is used.

```
typedef_struct _NDIS_802_11_BSSID_LIST_EX
{
 ULONG  NumberOfItems;
 NDIS_WLAN_BSSID_EX  Bssid[1];
} NDIS_802_11_BSSID_LIST_EX, *PNDIS_802_11_BSSID_LIST_EX;
```

The members of this structure contain the following information:

© 2003 Microsoft Corporation. All rights reserved.

NumberOfItems

Specifies the number of items contained in the **Bssid** array. The **Bssid** array must be at least 1 in size. If there are no BSSIDs detected, NumberOfItems is 0.

Bssid

Specifies an array of NDIS_WLAN_BSSID_EX structures.

An NDIS_WLAN_BSSID_EX structure is defined as follows:

```
struct _NDIS_WLAN_BSSID_EX
{
 ULONG  Length;
 NDIS_802_11_MAC_ADDRESS  MacAddress;
 Uchar  Reserved[2];
 NDIS_802_11_SSID  Ssid;
 ULONG  Privacy;
 NDIS_802_11_RSSI  Rssi;
 NDIS_802_11_NETWORK_TYPE  NetworkTypeInUse;
 NDIS_802_11_CONFIGURATION  Configuration;
 NDIS_802_11_NETWORK_INFRASTRUCTURE  InfrastructureMode;
 NDIS_802_11_RATES_EX  SupportedRates;
 ULONG IELength;
 UCHAR IEs[1];
} NDIS_WLAN_BSSID_EX, *PNDIS_WLAN_BSSID_EX;
```

The members of this structure contain the following information:

Length

Specifies the length of the NDIS_WLAN_BSSID_EX structure in bytes. Each structure needs to be a multiple of 4 bytes in length. That is, this field must be on a 4-byte address boundary. Thus, the Length field must contain a value that is a multiple of 4 bytes. As a result, there may be zero to 3 bytes of padding after the end of the IEs field.

Note: The padding for the structure is to allow drivers that return NDIS_WLAN_BSSID_EX to work with operating systems expecting the old NDIS_WLAN_BSSID structure.

MacAddress.

Specifies a media access control (MAC) address. Each access point has a unique MAC address that is the same as the BSSID.

Reserved

Reserved. Do not use. This member maintains the DWORD alignment of the structure.

Ssid

Specifies an SSID as defined in the NDIS_802_11_SSID structure.

Privacy

Specifies a WEP encryption requirement.

Rssi

Specifies the RSSI in dBm. The typical values are -10 to -200.

NetworkTypeInUse

Specifies the network type as defined in the NDIS_802_11_NETWORK_TYPE enumeration.

Configuration

Specifies the radio parameter configuration as defined in the NDIS_802_11_CONFIGURATION structure.

Notes:

© 2003 Microsoft Corporation. All rights reserved.

- For 2.4 GHz DSSS, 2.4 GHz, and 5 GHz OFDM, the current frequency should be returned in DSConfig.
- If an AP uses the same BSSID for multiple radios (for example, 2.4 GHz and 5 GHz), then an entry should be made in the OID_802_11_BSSID_LIST for each radio.
- For 2.4 GHz DSSS, 2.4 GHz OFDM, and 5 GHz OFDM, the current frequency should be in DSConfig as in the 802.11a specification.

Valid ranges for DSConfig are:

2,412,000 to 2,484,000 for 2.4 GHz

5,000,000 to 6,000,000 for 5 GHz

Note: In the United States, the valid subset for 5 GHz is 5,180,000 to 5,240,000, 5,260,000 to 5,320,000, and 5,745,000 to 5,805,000.

InfrastructureMode
Specifies the infrastructure mode as defined in the NDIS_802_11_NETWORK_INFRASTRUCTURE enumeration.

SupportedRates
Specifies a set of supported rates defined in an NDIS_802_11_RATES array for BSSID_LIST and NDIS_802_11_RATES_EX for BSSID_LIST_EX. The additional entries in NDIS_802_11_RATE_EX are to support the additional rates that 802.11g supports. Any unused entries left at the end of the array should be set to zero.

IELength

This field is only defined in NDIS_802_11_BSSID_EX. This field contains the number of octets in the IEs array. This should not contain any padding as described in the **Length** description.

IEs

This field is only defined in NDIS_802_11_BSSID_EX. This field contains any information elements from the beacon or probe response messages. The field must include the time stamp, beacon interval, and capability fixed-sized elements as the first three elements in the field. The information elements should be the latest information. Where the information element is available only in one of the messages, the information elements should be merged together. Some information elements to note are the TIM and SSID information elements.

The order of the information elements should be:

- First, the three fixed information elements (time stamp, beacon interval and capability) from the last beacon or probe response message.

  Note: The time stamp is eight octets in size, the beacon interval is two octets in size, and capability is two octets in size. This is defined in structure NDIS_802_11_FIXED_IEs. Refer to section 7.3.1 in the IEEE 802.11 1999 specification for the definition of these fields.

- Second, the information elements in the order received in the last beacon or probe response.

  Note: The structure NDIS_802_11_VARIABLE_IEs describes the format of the variable length information elements, See section 7.3.2 in the IEEE 802.11 1999 specification for the definition of the Information elements.

© 2003 Microsoft Corporation. All rights reserved.

- Third, any information elements not in the last beacon or probe response. For example, if the probe response message was the last message received, the TIM information element is added from the last beacon message received, along with any other information elements that were in the beacon, but not in the probe response. If the beacon message was the last message received and the SSID is blank, the SSID from the last probe message received is added along with any other information elements that were in the probe response, but not in the beacon message.

OID_802_11_ AUTHENTICATION_MODE

Sets the IEEE 802.11 authentication mode.

Data type: NDIS_802_11_AUTHENTICATION_MODE.

Query: Current authentication mode.

Set: Sets the authentication mode to open, shared, auto-switch, WPA, WPA-PSK or WPA-None.

Indication: Not supported.

Errors: Returns an error code of NDIS_STATUS_INVALID_DATA if an invalid value is used.

Returns an error code of NDIS_STATUS_NOT_SUPPORTED if the NIC does not support this OID or if it does not support an authentication mode such as Ndis802_11AuthModeWPANone.

The NDIS_802_11_AUTHENTICATION_MODE enumeration defines the authentication modes as follows:

Ndis802_11AuthModeOpen
Specifies IEEE 802.11 open authentication mode. There are no checks when accepting clients in this mode.

Ndis802_11AuthModeShared
Specifies IEEE 802.11 shared authentication that uses a preshared WEP key.

Ndis802_11AuthModeAutoSwitch
Specifies auto-switch mode. When using auto-switch mode, the NIC tries IEEE 802.11 shared authentication mode first. If shared mode fails, the NIC attempts to use IEEE 802.11 open authentication mode. The use of this setting is not recommended.

Ndis802_11AuthModeWPA
Specifies WPA version 1 security. In Infrastructure, specifies the use of IEEE 802.1X for security. When this mode is set, data messages transmitted before keys are installed will be unencrypted; data messages transmitted after keys are installed will be encrypted. IEEE 802.1X messages are encrypted using key mapping keys only. Only IEEE 802.1X messages are allowed to be sent unencrypted. The authentication suite in the WPA information element is set to 802.1X WPA.

© 2003 Microsoft Corporation. All rights reserved.

Ndis802_11AuthModeWPAPSK

Specifies WPA version 1 security. Specifies the use of a preshared key with IEEE 802.1X infrastructure mode. This mode will be set when a preshared key has been configured for use with IEEE 802.1X. When this mode is set, data messages transmitted before keys are installed will be unencrypted; data messages transmitted after keys are installed will be encrypted. IEEE 802.1X messages are encrypted using pairwise keys only. Only IEEE 802.1X messages are allowed to be sent unencrypted. The authentication suite in the WPA information element is set to preshared key over 802.1X.

Note: The only difference between Ndis802_11AuthModeWPAPSK and Ndis802_11AuthModeWPA is the authentication suite in the WPA information element and that the client user interface (UI) requires the user to enter a pre-shared key for the Ndis802_11AuthModeWPAPSK case.

Ndis802_11AuthModeWPANone

Specifies WPA version 1 security. Specifies the use of a preshared key without IEEE 802.1X in IBSS mode.

If the authentication mode is set to WPA or WPA-PSK mode, the NIC should not associate with a non-WPA access point.

OID_802_11_PRIVACY_FILTER

Requests that the miniport driver set its NIC's IEEE 802.1X privacy filter mode to a specified value.

Data type:	NDIS_802_11_PRIVACY_FILTER.
Query:	Current mode.
Set:	Sets to open or 802.1X filtering, zero is open, 1 is 802.1X filtering.
Indication:	Not supported.
Errors:	Returns an error code of NDIS_STATUS_NOT_SUPPORTED if the NIC does not support this OID. Returns an error code of NDIS_STATUS_INVALID_DATA if an invalid value is used.

The NDIS_802_11_PRIVACY_FILTER enumeration defines the authentication modes as follows:

Ndis802_11PrivFilterAcceptAll

Specifies an open mode. In this mode, the NIC accepts any packet if the packet is not encrypted or if the NIC successfully decrypts it.

Ndis802_11PrivFilter8021xWEP

Specifies a filtering mode. In the IEEE 802.1X filtering mode, IEEE 802.1X packets are accepted even if they are not encrypted. However, the NIC accepts nothing else unless it is encrypted.

OID_802_11_BSSID_LIST_SCAN

Requests that the miniport driver direct the IEEE 802.11 NIC to request a survey of BSSs. No data is associated with this OID. The NIC uses the following parameters, as defined in the IEEE 802.11 specifications, in its request:

- BSSType equals infrastructure BSS and independent BSS.
- BSSID equals broadcast BSSID.
- SSID equals broadcast SSID.

© 2003 Microsoft Corporation. All rights reserved.

- ScanType equals active, passive, or a combination of both passive and active scanning approaches.
- ChannelList equals all permitted frequency channels.

On receiving a call to this OID, the IEEE 802.11 NIC may perform a scan using active, passive, or a combination of both passive and active scanning approaches and refresh the list of BSSIDs in the IEEE 802.11 NIC's database.

The IEEE 802.11 NIC should minimize the duration to complete a call to this OID. Therefore, active scanning is preferred whenever it is considered appropriate.

For IEEE 802.11a- and IEEE 802.11b-capable NICs, this OID should initiate a scan on both IEEE 802.11a and IEEE 802.11b. The NIC can perform background scanning across all channels between scan periods and return the full scan list when queried to ensure that this list is returned within two seconds of this OID being set. It can also choose the order of channels to scan to ensure that the most likely channels are scanned first.

If the radio is turned off when this OID is called, it should return NDIS_STATUS_SUCCESS and return a BSSID list with **NumberOfItems** equal to zero, not a cached BSSID list.

Notes:

- If the IEEE 802.11 NIC is currently associated with a particular BSSID and SSID that is not contained in the list of BSSIDs generated by this scan, the BSSDescription of the currently associated BSSID and SSID should be appended to the list of BSSIDs in the IEEE 802.11 NIC's database. Responses from all BSSs on frequency channels that are permitted in the particular region where the IEEE 802.11 NIC is currently operating, should be included in the list of BSSIDs in the IEEE 802.11 NIC's database.
- The OID may be called very frequently (say, every five seconds). Miniport drivers should minimize the performance impact of calling this OID. In particular, this OID should not cause the NIC to associate with a different AP.

Data type: No data is associated with this set.

Query: Not supported.

Set: Performs a survey of potential BSSs.

Indication: Not supported.

Errors: Returns an error code of NDIS_STATUS_INVALID_DATA if an invalid value is used.

Returns an error code of NDIS_STATUS_NOT_SUPPORTED if the NIC does not support this OID.

OID_802_11_ENCRYPTION_STATUS

This OID was the OID_802_11_WEP_STATUS OID. Query should respond with the encryption status indicating whether encryption is disabled, absent, or not supported, or which encryption mode or modes are enabled.

A query response indicating that the key is absent implies that no group keys are available for the NIC to encrypt data. Note that this implies that while keys may be available, the NIC does not have a key for MPDU transmission with encryption. A query response indicating that this OID is not supported implies that the NIC does not support any encryption modes. That is, the NIC is not capable of encrypting data and, therefore, encryption cannot be enabled or disabled.

© 2003 Microsoft Corporation. All rights reserved.

A single encryption value can be set on the NIC, though this may enable one or more encryption modes or disable all encryption modes only. Encryption may be enabled or disabled without a transmit encryption key being available. If the set encryption status operation cannot be done, the driver returns an NDIS_STATUS value of NDIS_STATUS_NOT_ACCEPTED.

If the NIC does not support AES, attempting to set Ndis802_11Encryption3Enabled should fail with NDIS_STATUS_NOT_SUPPORTED.

If the NIC does not support TKIP, attempting to set Ndis802_11Encryption3Enabled or Ndis802_11Encryption2Enabled should fail with NDIS_STATUS_NOT_SUPPORTED.

If WEP, TKIP or AES is enabled and no key is available for MPDU transmission, only 802.1X packets should be sent.

Data type:	NDIS_802_11_ENCRYPTION_STATUS.
Query:	Returns the current encryption status.
Set:	Permitted to enable or disable encryption and the encryption modes.
Indication:	Not supported.
Errors:	Returns an error code of NDIS_STATUS_INVALID_DATA if an invalid value is used. Returns an error code of NDIS_STATUS_NOT_SUPPORTED if the NIC does not support this OID.

The NDIS_802_NDIS_802_11_ENCRYPTION_STATUS enumeration defines the encryption status values as follows:

Ndis802_11EncryptionNotSupported
Indicates that encryption (WEP, TKIP, and AES) is not supported.

Ndis802_11EncryptionDisabled
Indicates that AES, TKIP, and WEP are disabled, and a transmit key is available.

Ndis802_11Encryption1Enabled
Indicates that WEP is enabled; TKIP and AES are not enabled, and a transmit key may or may not be available.

Ndis802_11Encryption1KeyAbsent
Indicates that AES, TKIP, and WEP are disabled, and a transmit key is not available.

Ndis802_11TEncryption2Enabled
Indicates that TKIP and WEP are enabled; AES is not enabled, and a transmit key is available.

Ndis802_11Encryption2KeyAbsent
Indicates that there are no transmit keys available for use by TKIP or WEP, and TKIP and WEP are enabled; AES is not enabled.

Ndis802_11Encryption3Enabled
Indicates that AES, TKIP, and WEP are enabled, and a transmit key is available.

Ndis802_11Encryption3KeyAbsent
Indicates that there are no transmit keys available for use by AES, TKIP, or WEP, and AES, TKIP, and WEP are enabled.

Note: When the NIC is not associated, the decision whether a key is available or absent is made only on default keys.

© 2003 Microsoft Corporation. All rights reserved.

Note: The following table shows the return values for all possible states. The values have been chosen so that drivers with the new functionality still work with operating systems expecting the previous OID functionality.

Status Return	AES	TKIP	WEP	Transmit Key
EncryptionNotSupported	Not Supported	Not Supported	Not Supported	No
EncryptionNotSupported	Not Supported	Not Supported	Not Supported	Yes
Encryption1KeyAbsent	Disabled/Not Supported	Disabled/Not Supported	Disabled	No
EncryptionDisabled	Disabled/Not Supported	Disabled/Not Supported	Disabled	Yes
Encryption1Enabled	Disabled/Not Supported	Disabled/Not Supported	Enabled	No
Encryption1Enabled	Disabled/Not Supported	Disabled/Not Supported	Enabled	Yes
Not Applicable	Disabled/Not Supported	Enabled	Disabled/Not Supported	No
Not Applicable	Disabled/Not Supported	Enabled	Disabled/Not Supported	Yes
Encryption2KeyAbsent	Disabled/Not Supported	Enabled	Enabled	No
Encryption2Enabled	Disabled/Not Supported	Enabled	Enabled	Yes
Not Applicable	Enabled	Disabled/Not Supported	Disabled/Not Supported	No
Not Applicable	Enabled	Disabled/Not Supported	Disabled/Not Supported	Yes
Not Applicable	Enabled	Disabled/Not Supported	Enabled	No
Not Applicable	Enabled	Disabled/Not Supported	Enabled	Yes
Not Applicable	Enabled	Enabled	Disabled/Not Supported	No
Not Applicable	Enabled	Enabled	Disabled/Not Supported	Yes
Encryption3KeyAbsent	Enabled	Enabled	Enabled	No
Encryption3Enabled	Enabled	Enabled	Enabled	Yes

Notes:

- These OIDs enable and disable the encryption/integrity hardware in groups. This does not reflect any policy decisions made elsewhere as to which ciphers an AP must support in order for a system to be allowed to associate with the AP.
- If an invalid value is used, the OID should fail with NDIS_STATUS_INVALID_DATA.

This OID also affects how the 802.11 MAC generates the WPA information element in the Associate request and Reassociate request, and whether it associates to the AP or not, or whether it authenticates in IBSS mode. The following table defines the possible options:

© 2003 Microsoft Corporation. All rights reserved.

AP Unicast Cipher	AP Multicast Cipher	Encryption Status OID Set Value	ESS Associate or IBSS Authenticate	Associate Unicast Cipher	Associate Multicast Cipher
None	WEP	Encryption1	Yes	None	WEP
None	WEP	Encryption2	No	N/A	N/A
None	WEP	Encryption3	No	N/A	N/A
None	TKIP	Encryption1	No	N/A	N/A
None	TKIP	Encryption2	Yes	None	TKIP
None	TKIP	Encryption3	No	N/A	N/A
None	AES	Encryption1	No	N/A	N/A
None	AES	Encryption2	No	N/A	N/A
None	AES	Encryption3	Yes	None	AES
TKIP	WEP	Encryption1	No	N/A	N/A
TKIP	WEP	Encryption2	Yes	TKIP	WEP
TKIP	WEP	Encryption3	No	N/A	N/A
TKIP	TKIP	Encryption1	No	N/A	N/A
TKIP	TKIP	Encryption2	Yes	TKIP	TKIP
TKIP	TKIP	Encryption3	No	N/A	N/A
TKIP	AES	Encryption1	No	N/A	N/A
TKIP	AES	Encryption2	No	N/A	N/A
TKIP	AES	Encryption3	No	N/A	N/A
AES	WEP	Encryption1	No	N/A	N/A
AES	WEP	Encryption2	No	N/A	N/A
AES	WEP	Encryption3	Yes	AES	WEP
AES	TKIP	Encryption1	No	N/A	N/A
AES	TKIP	Encryption2	No	N/A	N/A
AES	TKIP	Encryption3	Yes	AES	TKIP
AES	AES	Encryption1	No	N/A	N/A
AES	AES	Encryption2	No	N/A	N/A
AES	AES	Encryption3	Yes	AES	AES

A NIC reports supporting WEP if it supports WEP-40 or WEP-104 and reports supporting AES if it supports AES-WRAP or AES-CCMP. The NIC should not associate to an AP if the AES or WEP cipher suite advertised by the AP is not supported.

OID_802_11_RELOAD_DEFAULTS

A call to this OID will cause the IEEE 802.11 NIC Driver/Firmware to reload the available default settings for the specified type field.

For example, when reload defaults are set to WEP keys, the IEEE 802.11 NIC Driver/Firmware should reload the available default WEP key settings from the permanent storage (that is, registry key or flash settings, into the running configuration of the IEEE 802.11 NIC that is not held in the permanent storage). A call to this OID with the reload defaults set to WEP keys ensures that the WEP key settings in the IEEE 802.11 NIC running configuration are returned to the configuration following the enabling of the interface from a disabled state (that is, when the IEEE 802.11 driver is loaded upon enabling the IEEE 802.11 NIC). The keys should only be loaded if it is valid for the NIC (that is, if the encryption state is TKIP, then WEP keys should be silently skipped).

© 2003 Microsoft Corporation. All rights reserved.

All current encryption and integrity keys should be discarded before loading default keys.

Note: Keys set through OID_802_11_ADD_KEY should not be loaded with OID_802_11_RELOAD_DEFAULTS.

Data type: NDIS_802_11_RELOAD_DEFAULTS.
Query: Not supported.
Set: Sets results in the reloading of the available defaults for the specified type field.
Indication: Not supported.

The NDIS_802_11_RELOAD_DEFAULTS enumeration defines the available reload options as follows:

Ndis802_11ReloadWEPKeys
Specifies that the driver should reload available default WEP keys from permanent storage.

OID_802_11_ASSOCIATION_INFORMATION

This OIDS is used to query and set the information elements used in the last association/reassociation request to an AP and in the last association/reassociation response from the AP.

Data type: NDIS_802_11_ASSOCIATION_INFORMATION
Query: Queries the information.
Set: Sets the desired information.
Indication: Not supported.
Errors: Returns an error code of NDIS_STATUS_INVALID_DATA if an invalid value is used.

Returns an error code of NDIS_STATUS_NOT_SUPPORTED if the NIC does not support this OID.

```
#define NDIS_802_11_AI_REQFI_CAPABILITIES      1
#define NDIS_802_11_AI_REQFI_LISTENINTERVAL    2
#define NDIS_802_11_AI_REQFI_CURRENTAPADDRESS  4

#define NDIS_802_11_AI_RESFI_CAPABILITIES      1
#define NDIS_802_11_AI_RESFI_STATUSCODE        2
#define NDIS_802_11_AI_RESFI_ASSOCIATIONID     4

typedef struct _NDIS_802_11_ASSOCIATION_INFORMATION
{
    ULONG Length;
    USHORT AvailableRequestFixedIEs;
    struct _NDIS_802_11_AI_REQFI {
            USHORT Capabilities;
            USHORT ListenInterval;
            NDIS_802_11_MAC_ADDRESS  CurrentAPAddress;
    } RequestFixedIEs;
    ULONG RequestIELength;
    ULONG OffsetRequestIEs;
    USHORT AvailableResponseFixedIEs;
    struct _NDIS_802_11_AI_RESFI {
```

© 2003 Microsoft Corporation. All rights reserved.

```
            USHORT Capabilities;
            USHORT StatusCode;
            USHORT AssociationId;
    } ResponseFixedIEs;
    ULONG ResponseIELength;
    ULONG OffsetResponseIEs;
} NDIS_802_11_ASSOCIATION_INFORMATION,
*PNDIS_802_11_ASSOCIATION_INFORMATION;
```

The members of this structure contain the following information:

Length
This field is the length of the structure not including any variable length information elements also included in the OID buffer.

AvailableRequestFixedIEs

This field contains a bit mask of the available elements in the **RequestFixedIEs**.

RequestFixedIEs

This field contains any fixed information elements from the association or reassociation request message. The MacAddress only exists in the reassociation request so **RequestFixedIELength** does not include the MacAddress if the request is an associate request.

During a query, this contains the fixed information elements that were in the last association or reassociate request.

During a set, the specified fixed information elements should be merged into the next association or reassociation request. The NIC should add any fixed information elements not specified. The fixed information elements may not be specified on a set. In this case, the NIC should supply the fixed information elements.

RequestIELength

This field contains the number of octets in the **OffetRequestIEs** buffer. This should be set to 0 if a request has not been made.

OffsetRequestIEs

This field contains the offset in the buffer containing any variable length information elements from the association or reassociation request message.

During a query, this contains the information elements that were in the last association or reassociate request attempt. The request may succeed or fail.

During a set, the specified information elements should be merged into the next association or reassociation request. The NIC may add any information elements into the association or reassociation request, but should not modify any supplied information element.

Note: The information elements must be in the buffer after the NDIS_802_11_ASSOCIATION_INFORMATION structure and **OffsetRequestIEs** is the offset from the start of the NDIS_802_11_ASSOCIATION_INFORMATION structure to the start of the information elements. That is, the address of the start of the information elements must be able to be calculated by adding **OffsetRequestIEs** to the address of the start of the association information structure.

AvailableResponseFixedIEs

This field contains a bit mask of the available fixed information elements in the **ResponseFixedIEs** structure.

© 2003 Microsoft Corporation. All rights reserved.

ResponseFixedIEs

This field contains any fixed information elements from the association or reassociation response message. There are three fixed information elements in an associate or reassociate response.

During a query, this contains the fixed information elements that were in the last association or reassociate response.

ResponseIELength

This field contains the number of octets in the **OffsetResponseIEs** buffer. This must be 0 in length for a set. It must be set to 0 in a query if a response for the last request is not received,

OffsetResponseIEs

This field contains the offset in the buffer containing any variable length information elements from the association or reassociation response message.

During a query, this contains the information elements that were in the last association or reassociate response.

Note: The information elements must be in the buffer after the NDIS_802_11_ASSOCIATION_INFORMATION structure and the **OffsetResponseIEs** is the offset from the start of the NDIS_802_11_ASSOCIATION_INFORMATION structure to the start of the information elements. That is, the address of the start of the information elements must be able to be calculated by adding **OffsetResponseIEs** to the address of the start of the association information structure.

OID_802_11_TEST

This OID is used to test the IEEE 802.11 driver and should never be used in normal operation.

Data type: NDIS_802_11_TEST
Query: Not supported.
Set: Sets the desired test.
Indication: Not supported.

```
typedef struct _NDIS_802_11_TEST
{
 ULONG Length;
 ULONG Type;
 union {
      struct _AuthenticationEvent {
       NDIS_802_11_STATUS_TYPE StatusType;
       NDIS_802_11_AUTHENTICATION_REQUEST Request[1];
      } AuthenticationEvent;
      NDIS_802_11_RSSI RssiTrigger;
 };
} NDIS_802_11_TEST, *PNDIS_802_11_TEST;
```

The members of this structure contain the following information:

Length
Specifies the length of the NDIS_802_11_TEST structure in bytes.

Type
Specifies the Test action.

© 2003 Microsoft Corporation. All rights reserved.

Type Value 1

When Type is 1, the OID should generate an NDIS_STATUS_MEDIA_SPECIFIC_INDICATION with the StatusBuffer containing StatusType and Request equal to the values supplied with this OID. The number of Request structures is worked out by the **Length** field.

Type Value 2

When Type is 2 the OID should generate an NDIS_STATUS_MEDIA_SPECIFIC_INDICATION with NDIS_802_11_RSSI as the contents of the StatusBuffer.

Any other Type Value

Fail with NDIS_STATUS_INVALID_DATA

Resources and Call to Action

- To optimize network adapter capabilities for Windows XP, follow the guidelines described in this white paper. For implementation details, see the Windows Driver Development Kit (DDK).
- For more information about networking and wireless technologies, see: http://www.microsoft.com/hwdev/tech/network/
- For information about the current Hardware Compatibility Test (HCT) and Windows Logo Program for Hardware requirements, see: http://www.microsoft.com/hwdq/hwtest/ http://www.microsoft.com/winlogo/hardware/default.mspx

© 2003 Microsoft Corporation. All rights reserved.