[MS-RNDIS]: Remote Network Driver Interface Specification (RNDIS) Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
09/25/2009	0.1	Major	First Release.
11/06/2009	0.1.1	Editorial	Revised and edited the technical content.
12/18/2009	0.1.2	Editorial	Revised and edited the technical content.
01/29/2010	0.2	Minor	Updated the technical content.
03/12/2010	0.3	Minor	Updated the technical content.
04/23/2010	0.3.1	Editorial	Revised and edited the technical content.
06/04/2010	0.3.2	Editorial	Revised and edited the technical content.
07/16/2010	0.3.2	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	0.3.2	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	0.3.2	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	0.3.2	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	0.3.2	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	0.3.2	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

This document specifies the Remote Network Driver Interface Specification (RNDIS) Protocol. The Remote Network Driver Interface Specification Protocol, referred to also as RNDIS in this document defines the communication between a host and **network device** connected over an external **bus transport** such as Universal Serial Bus (**USB**), so that the host can obtain network connectivity through the RNDIS-compliant device. The protocol enables the host to provide a vendor-independent **class driver** for an RNDIS compliant network device.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

bus
device
little-endian
padding
resource
universal serial bus (USB)

The following terms are specific to this document:

bus transport: A **bus** mechanism that can transport bits between the host and the **device**, such as **USB**.

channel: An independent communication mechanism on the **bus transport** between the host and the connected **device**.

class driver: A vendor independent driver running on the host operating system that supports any device that conforms to device interface specified by the host operating system.

control channel: The communication **channel** on the **bus transport** between the host and the device on which **control messages** are sent.

control message: A message used to control the **device**.

data channel: The communication **channel** on the **bus transport** between the host and the **device** on which networking data is sent and received.

data message: A message containing the network packet data.

flow control: The mechanism to rate-limit the messages from the sender so as to not overflow the buffers on the receiver.

host: A general purpose computer that is networking capable.

NDIS: Network Driver Interface Specification, which allows different networking components of an operating system to interoperate with one another, including the drivers for the **networking devices** connected to the system.

network device: A **device** that can provide network access for a host.

OID: Object **Id**entifier: A 4-byte integer used to identify a specific property of the device, as specified by the host operating system.

out-of-band data: Any special data that can be sent independent of the network data such as IP checksum offload information.

per-packet-info data: Any metadata associated with the packet such as the TCP Checksum and so on.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

1.2.2 Informative References

[USB-SPC] USB Consortium, "USB 3.0 Specification", April 2000, http://www.usb.org/developers/docs/

[IEEE802.3-2008] Institute of Electrical and Electronics Engineers, "Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications - Description", IEEE Std 802.3, 2008, http://standards.ieee.org/getieee802/802.3.html

[IEEE802.11-2007] Institute of Electrical and Electronics Engineers, "Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", ANSI/IEEE Std 802.11-2007, http://standards.ieee.org/getieee802/download/802.11-2007.pdf

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MSDN-NDIS] Microsoft Corporation, "NDIS Library Function References", http://msdn.microsoft.com/en-us/library/ff557045.aspx

[MSDN-OIDs] Microsoft Corporation, "Remote NDIS OIDs", http://msdn.microsoft.com/en-us/library/ff570633.aspx

[MSDN-RNDISUSB] Microsoft Corporation, "Remote NDIS To USB Mapping", http://msdn.microsoft.com/en-us/library/ff570657.aspx

1.3 Overview

Network device drivers are complex to implement and maintain, and are not always readily available in the **host** operating system. RNDIS attempts to alleviate these problems by eliminating the need for a vendor-supplied network device driver. Instead, the host operating system leverages a built-in class driver which supports any vendor's **device** that is conformant to the RNDIS Protocol.

RNDIS is a message-based protocol that can be implemented over any external bus transport such as USB. The **bus** should be capable of supporting separate control and **data channels** that are reliable and should offer sequential delivery of messages between the host and the device.

The RNDIS Protocol consists of a message set and a sequence of message exchanges between the host and the device to facilitate device configuration, querying device parameters, indicating device/network status change, and sending/receiving network data. The following diagram shows the general architecture of the RNDIS Protocol.

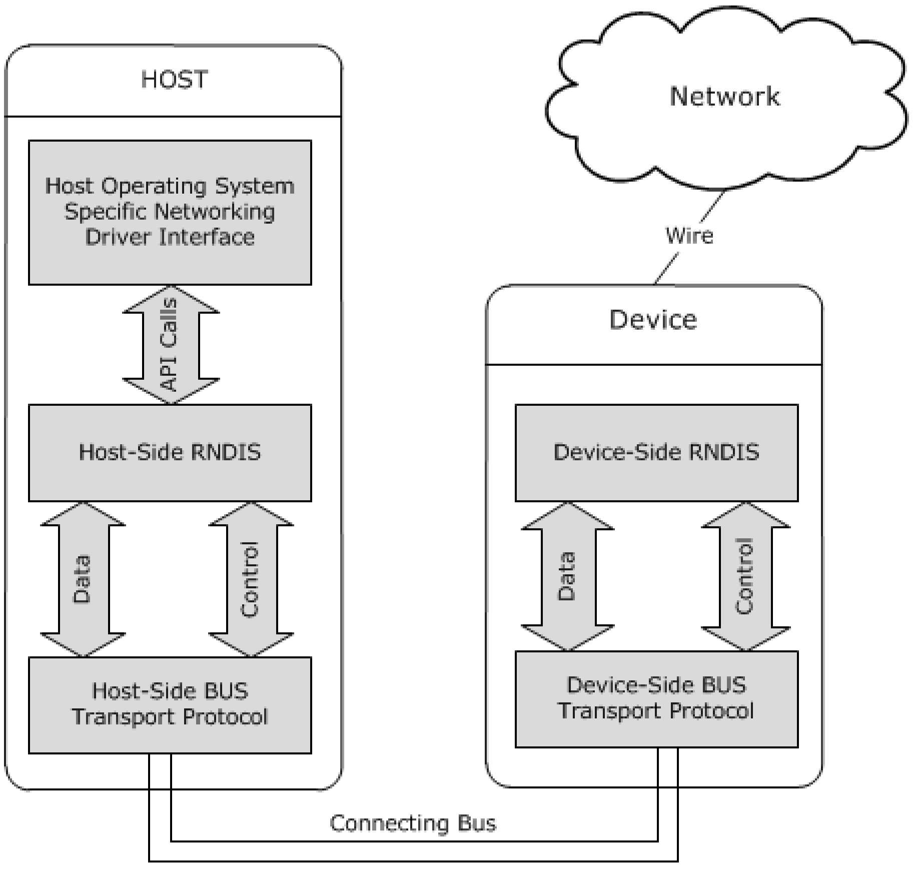

HOST
Host Operating System Specific Networking Driver Interface
API Calls
Host-Side RNDIS
Data
Control
Host-Side BUS Transport Protocol
Network
Wire
Device
Device-Side RNDIS
Data
Control
Device-Side BUS Transport Protocol
Connecting Bus

Figure 1: General architecture of the RNDIS Protocol

The host side of the protocol is responsible for the following:

- Initializing/deinitializing the device, as specified in sections 3.1.1 and 3.1.3.
- Establishing the data and **control channels** between the host and the device over the bus transport, as specified in sections 3.1.1 and 3.1.3.
- Exchanging of **data messages** and **control messages** between the host and the device as mandated by the host operating system's specific network driver interface, as specified in section 3.1.4.
- Encapsulating and transferring of the data packets over the bus transport, as mandated by the supported bus specification, as described in section 1.4.

The device side of the protocol is responsible for the following:

- Interpreting the control messages received on the bus as defined by the bus protocol and appropriately responding to them, as specified in sections 3.2.3 and 3.2.5. This includes handling requests for device initialization/deinitialization and establishing control/data channels.
- Indicating any network or device-related status to the host, as specified in section 3.2.7.
- Exchanging of data messages between the host and the device, as specified in section 3.2.5.7.
- Exercising **flow control** as mandated by the underlying bus to prevent the host from overflowing the data buffers of the device, as described in section 1.4.

1.4 Relationship to Other Protocols

RNDIS requires that the underlying bus protocol support at least two concurrent bidirectional communication **channels** per device and guarantee in-order and reliable message delivery per channel. The bus protocol is also required to provide the flow control mechanism. The module corresponding to these tasks is the bus interface on both the device and the host. The diagram titled "General architecture of the RNDIS protocol" in section 1.3 describes the host and the device side RNDIS stack.

On the host side of the RNDIS Protocol, the network data packets and the control messages are governed by the network driver interface specification of the host operating system.

On the device side, the protocol interfaces with the module responsible for indicating packets received from the network and consuming this network data sent from the host. It also interfaces with the module that controls the networking related state of the device.

The format of the actual networking data encapsulated in each packet depends on the underlying wire or wireless protocols such as [IEEE802.3-2008] or [IEEE802.11-2007].

1.5 Prerequisites/Preconditions

The device must comply with the device identification mechanism specified by the host operating system that enables mapping the device to the RNDIS class driver for device installation.

1.6 Applicability Statement

The protocol can be used with any external bus-based devices that have the ability to provide network access over a medium understood by the host operating system.

1.7 Versioning and Capability Negotiation

Supported Transports: USB 1.0, USB 1.1, and USB 2.0. See [USB-SPC] for details.

Protocol Versions: 1.0

Security and Authentication Methods: None.

Localization: Localization is implementation-dependent and usually applies to any strings that describe the device chosen for display on the host operating system.

Capability Negotiation: The highest-numbered protocol version, determined by the major version and minor version supported by both the host and the device. See section 3.2.5.1 for details.

1.8 Vendors-Extensible Fields

The Object Identifiers (**OID**s) and their semantics MUST be defined by the host operating system. Status values defined in the common status values table specified in section 2.2.1.2 can be extended by the operating system vendor as appropriate. No RNDIS fields can be directly extended by the device vendors. However, the device vendor can supply additional configurable parameters for the device using an OID as described in the documentation of OID_GEN_RNDIS_CONFIG_PARAMETER in [MSDN-OIDs].

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

RNDIS messages are transported between the host and the device by the protocol; the host and the device are connected through by encapsulating them in the bus-native protocol message formats. Messages that are sent on data and control channels MUST comply with the specification of the bus protocol they are mapped to respectively, including any **padding** requirements. The only RNDIS message that has explicit padding requirements is the REMOTE_NDIS_PACKET_MSG message, as defined in section 2.2.14.

2.2 Message Syntax

The REMOTE_NDIS_PACKET_MSG message is the only message exchanged over the data channel, and the rest of the messages defined in the following table are control messages. All multibyte values are represented in **little-endian byte order**.

2.2.1 Common Fields and Values

This section defines the header fields and their values that are common to multiple RNDIS messages.

2.2.1.1 RNDIS Message Types

The following table lists the message names, message type values, and descriptions.

RNDIS message types

Message Name	Message type value	Description
REMOTE_NDIS_PACKET_MSG	0x00000001	The host and device use this to send network data to one another.
REMOTE_NDIS_INITIALIZE_MSG	0x00000002	Sent by the host to initialize the device.
REMOTE_NDIS_INITIALIZE_CMPLT	0x80000002	Device response to an initialize message.
REMOTE_NDIS_HALT_MSG	0x00000003	Sent by the host to halt the device. This does not have a response. It is optional for the device to send this message to the host.
REMOTE_NDIS_QUERY_MSG	0x00000004	Sent by the host to send a query OID.
REMOTE_NDIS_QUERY_CMPLT	0x80000004	Device response to a query OID.
REMOTE_NDIS_SET_MSG	0x00000005	Sent by the host to send a set OID.
REMOTE_NDIS_SET_CMPLT	0x80000005	Device response to a set OID.
REMOTE_NDIS_RESET_MSG	0x00000006	Sent by the host to perform a soft reset on the device.
REMOTE_NDIS_RESET_CMPLT	0x80000006	Device response to reset message.
REMOTE_NDIS_INDICATE_STATUS_MSG	0x00000007	Sent by the device to indicate its status or an error when an unrecognized message is

Message Name	Message type value	Description
		received.
REMOTE_NDIS_KEEPALIVE_MSG	0x00000008	During idle periods, sent every few seconds by the host to check that the device is still responsive. It is optional for the device to send this message to check if the host is active.
REMOTE_NDIS_KEEPALIVE_CMPLT	0x80000008	The device response to a **keepalive** message. The host can respond with this message to a **keepalive** message from the device when the device implements the optional **KeepAliveTimer**.

2.2.1.2 Common Status Values

The following table shows the common status values along with descriptions.

Status values in RNDIS messages

Status identifier	Value	Description
RNDIS_STATUS_SUCCESS	0x00000000	Success
RNDIS_STATUS_FAILURE	0xC0000001	Unspecified error
RNDIS_STATUS_INVALID_DATA	0xC0010015	Invalid data error
RNDIS_STATUS_NOT_SUPPORTED	0xC00000BB	Unsupported request error
RNDIS_STATUS_MEDIA_CONNECT	0x4001000B	Device is connected to a network medium.
RNDIS_STATUS_MEDIA_DISCONNECT	0x4001000C	Device is disconnected from the medium.

2.2.2 REMOTE_NDIS_INITIALIZE_MSG

This message MUST be sent by the host to initialize the device.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageType																															
MessageLength																															
RequestID																															
MajorVersion																															
MinorVersion																															

MaxTransferSize

MessageType (4 bytes): Identifies the type of the RNDIS message and MUST be set to 0x00000002.

MessageLength (4 bytes): The length of this message, in bytes. MUST be set to 0x00000018.

RequestID (4 bytes): A 32-bit integer value, generated by the host, used to match the host's sent request to the response from the device. When responding to this message from the host, the device MUST use this value in the **RequestID** field of the response message. It is the responsibility of the host to ensure the uniqueness of this value among all outstanding request messages sent to the device.

MajorVersion (4 bytes): The major version of the RNDIS Protocol implemented by the host.<1>

MinorVersion (4 bytes): The minor version of the RNDIS Protocol implemented by the host.<2>

MaxTransferSize (4 bytes): The maximum size, in bytes, of any single bus data transfer that the host expects to receive from the device.<3>

2.2.3 REMOTE_NDIS_HALT_MSG

This message MUST be sent by the host to halt the device. The device MAY optionally implement this message to notify the host when it decides to halt.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageType																															
MessageLength																															
RequestID																															

MessageType (4 bytes): Identifies the type of the RNDIS message and MUST be set to 0x00000003.

MessageLength (4 bytes): The length of this message, in bytes. MUST be set to 0x0000000C.

RequestID (4 bytes): A unique 32-bit integer value generated by the host to track this message.

2.2.4 REMOTE_NDIS_QUERY_MSG

This message MUST be sent by the host to query an OID.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageType																															

MessageLength
RequestID
Oid
InformationBufferLength
InformationBufferOffset
Reserved
OIDInputBuffer (variable)
...

MessageType (4 bytes): Identifies the type of the RNDIS message and MUST be set to 0x00000004.

MessageLength (4 bytes): The total length of this message, including the header and the **OIDInputBuffer**.

RequestID (4 bytes): A 32-bit integer value, generated by the host, used to match the host's sent request to the response from the device. When responding to this message from the host, the device MUST use this value in the **RequestID** field of the response message. It is the responsibility of the host to ensure the uniqueness of this value among all outstanding request messages sent to the device.

Oid (4 bytes): The integer value of the host operating system-defined identifier, for the parameter of the device being queried for.<4>

InformationBufferLength (4 bytes): The length, in bytes, of the input data required for the OID query. This MUST be set to 0 when there is no input data associated with the OID.<5>

InformationBufferOffset (4 bytes): The offset, in bytes, from the beginning of **RequestID** field where the input data for the query is located in the message. This value MUST be set to 0 when there is no input data associated with the OID.

Reserved (4 bytes): Reserved for future use and MUST be set to 0. The device MUST treat it as an error otherwise.

OIDInputBuffer (variable): The input data supplied by the host, required for the OID query request processing by the device, as per the host **NDIS** specification.<6>

2.2.5 REMOTE_NDIS_SET_MSG

This message MUST be sent by host to set an OID.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageType																															
MessageLength																															
RequestID																															
Oid																															
InformationBufferLength																															
InformationBufferOffset																															
Reserved																															
OIDInputBuffer (variable)																															
...																															

MessageType (4 bytes): Identifies the type of the RNDIS message and MUST be set to 0x00000005.

MessageLength (4 bytes): The total length of this message, including the header and the **OIDInputBuffer**.

RequestID (4 bytes): A 32-bit integer value generated by the host and used to match the host's sent request to the response from the device. When responding to this message from the host, the device MUST use this value in the **RequestID** field of the response message. It is the responsibility of the host to ensure the uniqueness of this value among all outstanding request messages sent to the device.

Oid (4 bytes): The integer value of the host operating system-defined identifier, uniquely identifying a parameter of the device that is being set.<7>

InformationBufferLength (4 bytes): The length, in bytes, of the input data required for the OID query. This MUST be set to 0 when there is no input data associated with the OID.<8>

InformationBufferOffset (4 bytes): The offset, in bytes, from the beginning of the **RequestID** field, where the input data for the query is located in the message. This MUST be set to 0 when there is no input data associated with the OID.

Reserved (4 bytes): Reserved for future use and MUST be set to 0. The device MUST treat it as an error otherwise.

OIDInputBuffer (variable): The input data required for the OID set request to be processed by the device, as specified by the host operating system.<9>

2.2.6 REMOTE_NDIS_RESET_MSG

This message MUST be sent by the host to perform a soft reset on the device.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageType																															
MessageLength																															
Reserved																															

MessageType (4 bytes): Identifies the type of the RNDIS message and MUST be set to 0x00000006.

MessageLength (4 bytes): The total length of this message in bytes. MUST be set to 0x0000000C.

Reserved (4 bytes): Reserved for future use. MUST be set to zero. The device MUST treat it as an error otherwise.

2.2.7 REMOTE_NDIS_INDICATE_STATUS_MSG

This message MUST be sent by the device to indicate its status or an error when an unrecognized message is received.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageType																															
MessageLength																															
Status																															
StatusBufferLength																															
StatusBufferOffset																															
DiagnosticInfoBuffer (optional)																															
...																															
StatusBuffer (variable)																															
...																															

MessageType (4 bytes): Identifies the type of the RNDIS message and MUST be set to 0x00000007.

MessageLength (4 bytes): The total length of this message, in bytes, including the **StatusBuffer**, if any.

Status (4 bytes): A value indicating a status change on the device, including link state change such as media connect/disconnect, or an error value when the device receives a malformed message. See the common status values table in section 2.2.1.2 for possible values. The exact value of the **Status** field depends on the host operating system specification.

StatusBufferLength (4 bytes): The length of the status data, in bytes. This MUST be set to 0 when there is no **StatusBuffer**.<10>

StatusBufferOffset (4 bytes): When the **Status** field contains an error code, **StatusBufferOffset** is the offset, in bytes, from the beginning of the **Status** field at which the **DiagnosticInfoBuffer** field is located in the message. This buffer contains RNDIS_DIAGNOSTIC_INFO, as specified in section 2.2.7.1. In this case, the **StatusBuffer** contains the offending message that caused the device to send the REMOTE_NDIS_INDICATE_STATUS_MSG message to the host.

It is located immediately following the **DiagnosticInfoBuffer**. When the **Status** field represents a device state change such as a link state change, **StatusBufferOffset** is the offset, in bytes, from the beginning of the **Status** field at which the **StatusBuffer** is located. MUST be set to 0 if there is no associated status buffer.

DiagnosticInfoBuffer (8 bytes): This field is optional. **DiagnosticInfoBuffer** (RNDIS_DIAGNOSTIC_INFO) MUST be included in a response to a malformed message (that is, an unsupported message type) or when a REMOTE_NDIS_PACKET_MSG is received with inappropriate content, and the device cannot respond with any other RNDIS message. See section 2.2.7.1.

StatusBuffer (variable): This field is optional. When the status indication is for device state change, StatusBuffer MAY contain corresponding information as required by the host operating system specification. If the status indication does not have any additional information required by the host operating system specification, **StatusBuffer** is not present. In the case of a malformed message, **StatusBuffer** contains the message that was malformed.<11>

2.2.7.1 RNDIS_DIAGNOSTIC_INFO

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
DiagStatus																															
ErrorOffset																															

DiagStatus (4 bytes): Contains additional status information about the error itself. The common status values table in section 2.2.1.2 lists the possible values.

ErrorOffset (4 bytes): Specifies the zero-based offset from the beginning of the message, in bytes, at which the error was detected.

2.2.8 REMOTE_NDIS_KEEPALIVE_MSG

This message MUST be sent by the host to check that device is still responsive. It is optional for the device to send this message to check if the host is active.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageType																															
MessageLength																															
RequestID																															

MessageType (4 bytes): Identifies the type of the RNDIS message and MUST be set to 0x00000008.

MessageLength (4 bytes): The total length of this message, in bytes. MUST be set to 0x0000000C.

RequestID (4 bytes): A 32-bit integer value, generated by the host, used to match the host sent request to the response from the device. When responding to this message from the host, the device MUST use this value in the **RequestID** field of the response message. It is the responsibility of the host to ensure the uniqueness of this value among all outstanding request messages sent to the device.

2.2.9 REMOTE_NDIS_INITIALIZE_CMPLT

This message MUST be sent by the device in response to an initialize message.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageType																															
MessageLength																															
RequestID																															
Status																															
MajorVersion																															
MinorVersion																															
DeviceFlags																															
Medium																															
MaxPacketsPerTransfer																															
MaxTransferSize																															
PacketAlignmentFactor																															

Reserved
...

MessageType (4 bytes): Identifies the type of the RNDIS message type and MUST be set to 0x80000002.

MessageLength (4 bytes): The total length of this message, in bytes. MUST be set to 0x00000030.

RequestID (4 bytes): A 32-bit integer value that represents the **RequestID** field of the REMOTE_NDIS_INITIALIZE_MSG message to which this message is a response.

Status (4 bytes): The initialization status of the device. MUST be set to RNDIS_STATUS_SUCCESS (0x00000000) upon successful initialization or upon failure, and set to an appropriate error status value defined in the common status values table in section 2.2.1.2.

MajorVersion (4 bytes): Together with MinorVersion, the highest-numbered RNDIS Protocol version supported by the device.

MinorVersion (4 bytes): Together with **MajorVersion**, the highest-numbered RNDIS Protocol version supported by the device.

DeviceFlags (4 bytes): MUST be set to 0x000000010. Other values are reserved for future use.

MaxPacketsPerTransfer (4 bytes): The maximum number of concatenated REMOTE_NDIS_PACKET_MSG messages that the device can handle in a single bus transfer to it. This value MUST be at least 1.

MaxTransferSize (4 bytes): The maximum size, in bytes, of any single bus data transfer that the device expects to receive from the host.

PacketAlignmentFactor (4 bytes): The byte alignment the device expects for each RNDIS message that is part of a multimessage transfer to it. The value is specified as an exponent of 2; for example, the host uses $2^{\{\textbf{PacketAlignmentFactor}\}}$ as the alignment value.

Reserved (8 bytes): Reserved for future use and MUST be set to zero. It SHOULD be treated as an error otherwise.

2.2.10 REMOTE_NDIS_QUERY_CMPLT

This message MUST be sent by the device in response to a query OID message.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
MessageType																															
MessageLength																															
RequestID																															

Status
InformationBufferLength
InformationBufferOffset
OIDInputBuffer (variable)
...

MessageType (4 bytes): Identifies the type of the RNDIS message and MUST be set to 0x80000004.

MessageLength (4 bytes): The total length of this message, in bytes, including the OIDInputBuffer, if any.

RequestID (4 bytes): The 32-bit integer value in the **RequestID** field of the REMOTE_NDIS_QUERY_MSG message, to which this is a response.

Status (4 bytes): The status of processing for the query request. The common status values table in section 2.2.1.2 lists the allowed values and their meaning.

InformationBufferLength (4 bytes): The length, in bytes, of the data in the response to the query. This MUST be set to 0 when there is no OIDInputBuffer.<12>

InformationBufferOffset (4 bytes): The offset, in bytes, from the beginning of **RequestID** field where the response data for the query is located in the message. This MUST be set to 0 when there is no **OIDInputBuffer**.

OIDInputBuffer (variable): The response data to the OID query request as specified by the host. **OIDInputBuffer** is not required when the host specification does not require any information associated with the **Oid** in REMOTE_NDIS_QUERY_MSG to which this is the response.<13>

2.2.11 REMOTE_NDIS_SET_CMPLT

This message MUST be sent by the device in response to a set OID message.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageType																															
MessageLength																															
RequestID																															
Status																															

MessageType (4 bytes): Identifies the type of the RNDIS message and MUST be set to 0x80000005.

MessageLength (4 bytes): The total length of this message, in bytes. MUST be set to 0x00000010.

RequestID (4 bytes): The 32-bit integer value in the **RequestID** field of the REMOTE_NDIS_SET_MSG message, to which this is a response.

Status (4 bytes): The status of processing for the REMOTE_NDIS_SET_MSG message request. The common status values table in section 2.2.1.2 lists the allowed values and their meaning.

2.2.12 REMOTE_NDIS_RESET_CMPLT

This message MUST be sent by the device in response to a reset message.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageType																															
MessageLength																															
Status																															
AddressingReset																															

MessageType (4 bytes): Identifies the type of the RNDIS message and MUST be set to 0x80000006.

MessageLength (4 bytes): The total length of this message, in bytes. MUST be set to 0x00000010.

Status (4 bytes): The status of processing for the REMOTE_NDIS_RESET_MSG message request by the device to which this message is the response. The common status values table in section 2.2.1.2 lists the allowed values and their meaning.

AddressingReset (4 bytes): This field indicates whether the addressing information, which is the multicast address list or packet filter, has been lost during the reset operation. This MUST be set to 0x00000001 if the device requires that the host to resend addressing information or MUST be set to zero otherwise.

2.2.13 REMOTE_NDIS_KEEPALIVE_CMPLT

This message MUST be sent in response to a **keepalive** message.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageType																															
MessageLength																															
RequestID																															

Status

MessageType (4 bytes): Identifies the type of the RNDIS message and MUST be set to 0x80000008.

MessageLength (4 bytes): The total length of this message, in bytes. MUST be set to 0x00000010.

RequestID (4 bytes): The 32-bit integer value in the **RequestID** field of the REMOTE_NDIS_KEEPALIVE_MSG message, to which this message is a response.

Status (4 bytes): The status of processing for the REMOTE_NDIS_KEEPALIVE_MSG message request. The common status values table in section 2.2.1.2 lists the allowed values and their meaning.

2.2.14 REMOTE_NDIS_PACKET_MSG

This message MUST be used by the host and the device to send network data to one another.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageType																															
MessageLength																															
DataOffset																															
DataLength																															
OutOfBandDataOffset																															
OutOfBandDataLength																															
NumOutOfBandDataElements																															
PerPacketInfoOffset																															
PerPacketInfoLength																															
Reserved																															
...																															
PayLoad (variable)																															
...																															

Padding (variable)
...
REMOTE_NDIS_PACKET_MSG (variable)
...

MessageType (4 bytes): Identifies the type of the RNDIS message and MUST be set to 0x00000001.

MessageLength (4 bytes): The total length of this RNDIS message including the header, payload, and padding.

DataOffset (4 bytes): Specifies the offset, in bytes, from the start of the **DataOffset** field of this message to the start of the data. This MUST be an integer multiple of 4.

DataLength (4 bytes): Specifies the number of bytes in the payload of this message.

OutOfBandDataOffset (4 bytes): Specifies the offset, in bytes, of the first **out-of-band data** record from the start of the **DataOffset** field in this message. MUST be an integer multiple of 4 when out-of-band data is present or set to 0 otherwise. When there are multiple out-of-band data records, each subsequent record MUST immediately follow the previous out-of-band data record.

OutOfBandDataLength (4 bytes): Specifies, in bytes, the total length of the out-of-band data.

NumOutOfBandDataElements (4 bytes): Specifies the number of out-of-band records in this message.

PerPacketInfoOffset (4 bytes): Specifies the offset, in bytes, of the start of per-packet-info data record (section 2.2.14.1) from the start of the **DataOffset** field in this message. MUST be an integer multiple of 4 when **per-packet-info data** record is present or MUST be set to 0 otherwise. When there are multiple per-packet-info data records, each subsequent record MUST immediately follow the previous record.

PerPacketInfoLength (4 bytes): Specifies, in bytes, the total length of per-packet-information contained in this message.

Reserved (8 bytes): Reserved for future and MUST be set to zero. The host and the device MUST treat it as an error otherwise.

PayLoad (variable): Network data contained in this message.

Padding (variable): Additional bytes of zeros added at the end of the message to comply with the internal and external padding requirements. Internal padding SHOULD be as per the specification of the out-of-band data record and per-packet-info data record. The external padding size SHOULD be determined based on the **PacketAlignmentFactor** field specification in REMOTE_NDIS_INITIALIZE_CMPLT message by the device, when multiple REMOTE_NDIS_PACKET_MSG messages are bundled together in a single bus-native message. In this case, all but the very last REMOTE_NDIS_PACKET_MSG MUST respect the **PacketAlignmentFactor** field.

REMOTE_NDIS_PACKET_MSG (variable): This field is optional. When the device can support multiple packets per single bus transaction (see section 2.2.9), the host or device MAY append multiple REMOTE_NDIS_PACKET_MSG messages at the end of another REMOTE_NDIS_PACKET_MSG message in the same bus message for improved throughput when multiple packets are available to send to the other side. (This is solely for the purpose of improved performance.) Both the **MaxPacketsPerTransfer** and the **MaxTransferSize** values specified through the REMOTE_NDIS_INITIALIZE_CMPLT message MUST be respected.

2.2.14.1 Out-of-Band Data Record

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Size																															
Type																															
ClassInformationOffset																															
OutOfBandData (variable)																															
...																															

Size (4 bytes): Length, in bytes, of this header and appended out-of-band data and padding. This value MUST be an integer multiple of 4.

Type (4 bytes): MUST be as per host operating system specification.<14>

ClassInformationOffset (4 bytes): The byte offset from the beginning of this out-of-band data record to the beginning of out-of-band data.

OutOfBandData (variable): The out-of-band data.

2.2.14.2 Per-Packet-Info Data Record

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Size																															
Type																															
PerPacketInformationOffset																															
PerPacketData (variable)																															
...																															

Size (4 bytes): Length, in bytes, of this header plus **PerPacketData** plus any padding necessary. MUST be an integer multiple of 4.

Type (4 bytes): Must be as per the host operating system specification.<15>

PerPacketInformationOffset (4 bytes): The byte offset from the beginning of this per-packet-info record to the beginning of **PerPacketData**.

PerPacketData (variable): The per-packet- data.

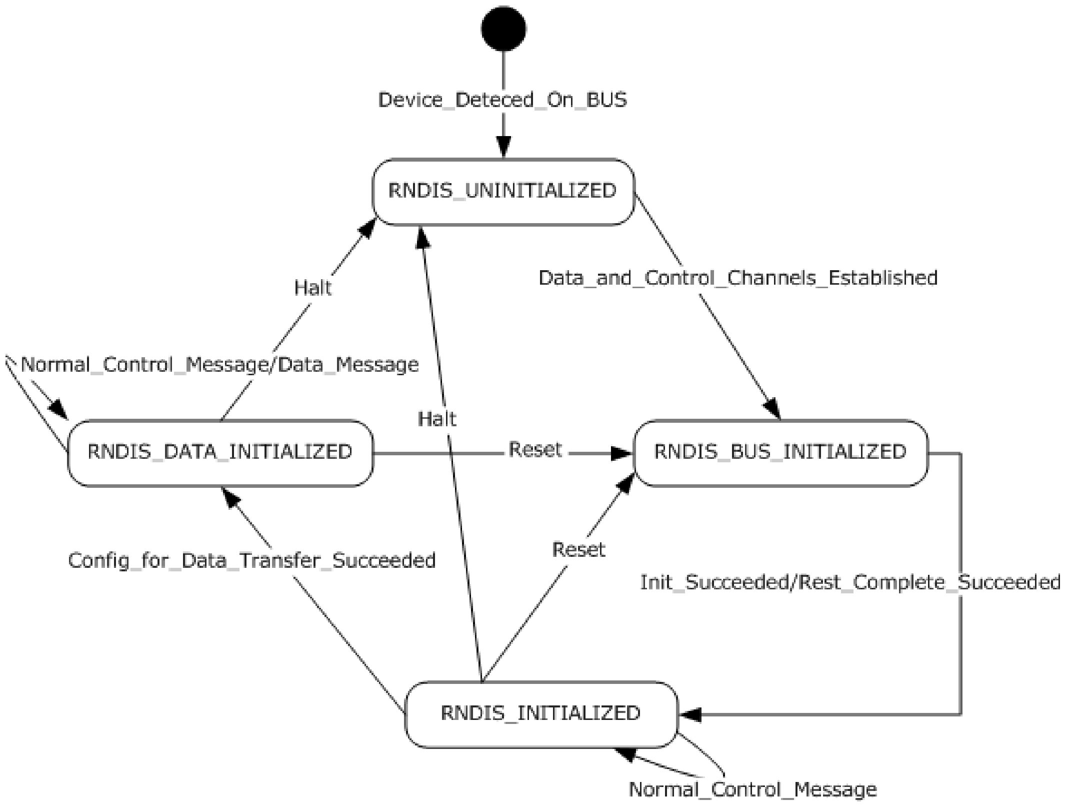
3 Protocol Details

3.1 Host Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. The document does not mandate that the implementations adhere to this model as long as their external behavior is consistent with that described in this document.

Following is the state diagram depicting various states the system could be in, what operations are initiated in each of the states, and what operations trigger a state transition.



Device_Deteced_On_BUS
RNDIS_UNINITIALIZED
Data_and_Control_Channels_Established
Halt
Normal_Control_Message/Data_Message
Halt
RNDIS_DATA_INITIALIZED
Reset
RNDIS_BUS_INITIALIZED
Config_for_Data_Transfer_Succeeded
Reset
Init_Succeeded/Rest_Complete_Succeeded
RNDIS_INITIALIZED
Normal_Control_Message

Figure 2: RNDIS Abstract Data Model sState diagram

3.1.1.1 State Description

RNDIS_UNINITIALIZED: The host operating system is running, the device is connected to the host via the chosen bus transport, and both the host and the device are not yet configured to exchange any RNDIS messages.

RNDIS_BUS _INITIALIZED: The RNDIS Protocol data and control channels are appropriately mapped and initialized on the bus transport, on both the host and the device side.

RNDIS_INITIALIZED: The device and the host are configured to receive any of the RNDIS control messages for suitably configuring or querying the device, indicate device status to the host, or reset the protocol or tear down the communication.

RNDIS_DATA_INITIALIZED: This is a substate of **RNDIS_INITIALIZED**, which is entered after the device has successfully processed all the REMOTE_NDIS_SET_MSG messages sent by the host with all the OIDs required to configure the device for data transfer, and has responded to the host with the appropriate REMOTE_NDIS_SET_CMPLT message. When the host and the device are in this state, apart from the control messages, they can exchange REMOTE_NDIS_PACKET_MSG messages for network data transfer between them.

3.1.1.2 Transition Event Description

Device_Detected_On_BUS: This transition event occurs when the RNDIS device is discovered on the bus by the host operating system.

Data_and_Control_Channels_Established: This transition event occurs when the RNDIS abstract data and control channels are successfully established between the host and the device through a suitable mapping.

Init_Succeeded: This transition occurs when the host sends a REMOTE_NDIS_INITIALIZE_MSG message to the device and device responds with a REMOTE_NDIS_INITIALIZE_CMPLT message indicating success.

Normal_Control_Message: This transition event occurs when the host receives a REMOTE_NDIS_QUERY_CMPLT, REMOTE_NDIS_SET_CMPLT, REMOTE_NDIS_INDICATE_STATUS_MSG, REMOTE_NDIS_KEEPALIVE_CMPLT or REMOTE_NDIS_KEEPALIVE_MSG message, or the device receives a REMOTE_NDIS_SET_MSG, REMOTE_NDIS_QUERY_MSG, REMOTE_NDIS_KEEPALIVE_MSG or REMOTE_NDIS_KEEPALIVE_CMPLT message.

Halt: This transition event occurs when the host or device receives a REMOTE_NDIS_HALT_MSG message or on bus-level disconnect or on hard-reset.

This transition event occurs when the device receives a REMOTE_NDIS_RESET_MSG message.

Reset_Complete_Succeeded: This transition event occurs when the host receives a REMOTE_NDIS_RESET_CMPLT message from the device, indicating success.

Config_For_Data_Transfer_Succeeded: This transition event occurs when any configuration required on the host side and the device side is successfully completed to initiate network data exchange between them.

Data_Message: This transition event occurs when host or device received REMOTE_NDIS_PACKET_MSG message.

3.1.2 Timers

KeepAliveTimer: Used by the host to determine the health of the device when the host and the device are in RNDIS_DATA_INITIALIZED state (section 3.1.1.1) and when there has been no other control or data traffic from the device to the host for a period that SHOULD suitably be chosen, based on the underlying bus characteristics.<16>

ControlTimeoutTimer: Used by the host to impose a timeout on the response from the device in return for any request sent on the control channel except for the REMOTE_NDIS_KEEPALIVE_MSG message.<17>

3.1.3 Initialization

The following diagram illustrates the steps for initialization.

1. The initialization occurs when the device is detected by the host operating system on a supported bus, represented by the **UNINITIALIZED** state.

2. Upon successful establishment of the data and the control channels over the supported bus, from the **BUS_INITIALIZED** state, the host sends a REMOTE_NDIS_INITIALIZE_MSG message to the device. The device responds with a REMOTE_NDIS_INITIALIZE_CMPLT message specifying its basic capabilities.

3. The host and the device are in **RNDIS_INITIALIZED** state and a sequence of REMOTE_NDIS_QUERY_MSG and REMOTE_NDIS_SET_MSG control messages are sent by the host to configure the device further, as specified in section 3.2.5. The device MUST respond to these messages appropriately as specified in section 3.2.5.

4. After the device is configured, the host MUST now send a REMOTE_NDIS_SET_MSG message with appropriate OIDs to initialize the data transfer. On a successful response from the device, the host and the device enter the **RNDIS_DATA_INITIALIZED** state and are ready for network data transfer. These OIDs MAY include parameters for configuring the device to indicate multicast, broadcast or directed packets to the host.

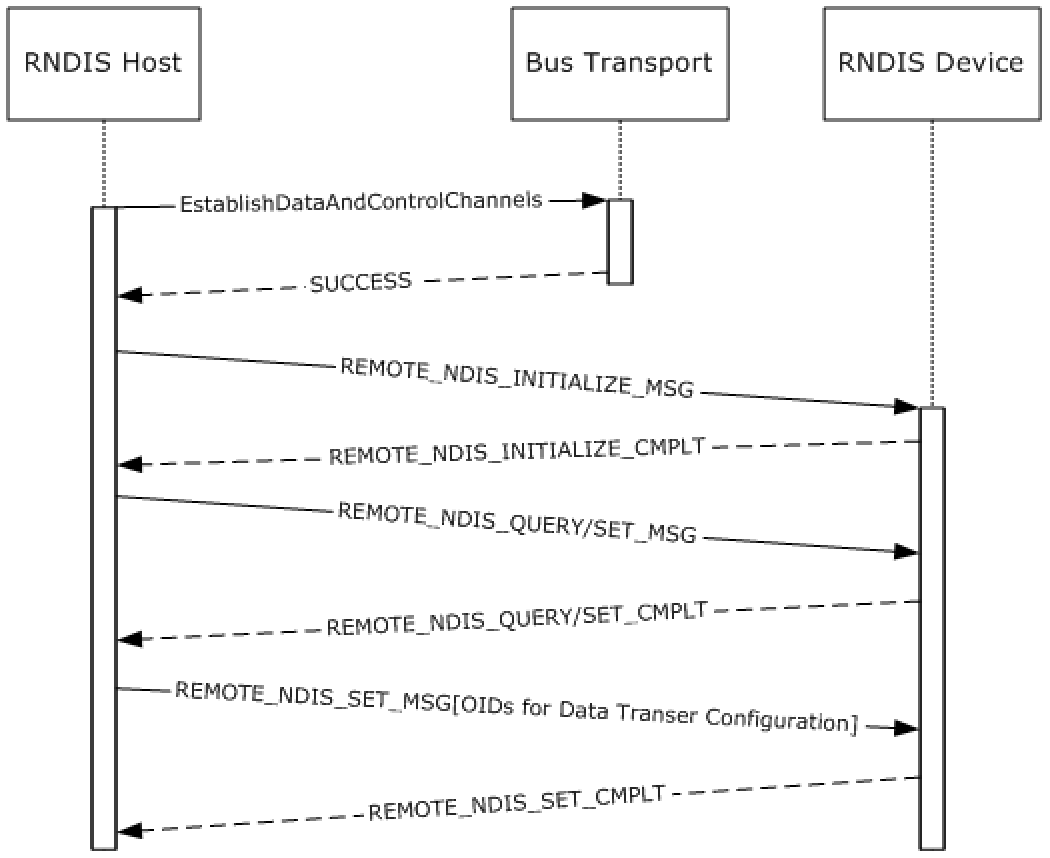

RNDIS Host
Bus Transport
RNDIS Device
EstablishDataAndControlChannels
SUCCESS
REMOTE_NDIS_INITIALIZE_MSG
REMOTE_NDIS_INITIALIZE_CMPLT
REMOTE_NDIS_QUERY/SET_MSG
REMOTE_NDIS_QUERY/SET_CMPLT
REMOTE_NDIS_SET_MSG[OIDs for Data Transer Configuration]
REMOTE_NDIS_SET_CMPLT

Figure 3: RNDIS Protocol initialization sequence diagram

3.1.4 Higher-Layer Triggered Events

Depending on the specification of the networking driver interface on the host operating system, any of the control or data messages can be sent from the host side of the protocol. In response, the host and the device will be in one of the appropriate states as specified in section 3.1.1.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Processing a REMOTE_NDIS_INITIALIZE_CMPLT Message

This message can be received only when the host and the device are in a **BUS_INITIALIZED** state. If the **Status** field of this message is set to **REMOTE_NDIS_STATUS_SUCCESS**, the host and the device move into **RNDIS_INITIALIZED** state. For any other status value, the host and the device move to **BUS_INITIALIZED** state. The host may attempt to initialize the device again or move into **UNINITIALIZED** state by undoing all operations previously made to reach **BUS_INITIALIZE** state.

The following parameters MUST be derived by the host while processing this message, to be adhered to for the rest of the communication:

- The highest-numbered version of the RNDIS Protocol that the host and the device can support as described in section 1.7.
- The maximum size of any RNDIS message that the device and host can support together per bus transaction.
- The limit on the number of network packets per bus transaction and byte alignment requirements, if multi-packet transfer is supported by the device and host,.
- The network media type per the NDIS specification that the device supports.

3.1.5.2 Processing a REMOTE_NDIS_QUERY_CMPLT Message

If the **Status** field is set to RNDIS_STATUS_SUCCESS, the query is assumed to have succeeded and the device returned value MUST be read from the **InformationBuffer** field as defined in REMOTE_NDIS_QUERY_CMPLT message syntax. The value corresponds to the request sent by the host with the matching **RequestID**. If the Status value is set to anything other than RNDIS_STATUS_SUCCESS, the action to be taken is determined by the network driver interface specification on the host. This message can be received only when the host and the device are in RNDIS_INITIALIZED or RNDIS_DATA_INITIALIZED states.

3.1.5.3 Processing a REMOTE_NDIS_SET_CMPLT Message

If the **Status** field is set to RNDIS_STATUS_SUCCESS, the corresponding set request as identified by the **RequestID** field is assumed to have succeeded. If the **Status** value is set to anything other than RNDIS_STATUS_SUCCESS, the action to be taken is determined by the network driver interface specification on the host. This message can be received only when the host and the device are in RNDIS_INITIALIZED or RNDIS_DATA_INITIALIZED states.

3.1.5.4 Processing a REMOTE_NDIS_KEEPALIVE_CMPLT Message

If the status field is set to RNDIS_STATUS_SUCCESS, the device is assumed to healthy, the KeepAliveTimer is reset, and no other action is taken. Otherwise, the host sends a REMOTE_NDIS_RESET_MSG message to the device. This can be received only when the host and the device are in the RNDIS_DATA_INITIALIZED or RNDIS_INITIALIZED state.

3.1.5.5 Processing a REMOTE_NDIS_RESET_CMPLT Message

If the **Status** field is set to RNDIS_STATUS_SUCCESS, the device is assumed to have completed the soft-reset successfully. If the **AddressingReset** field is set to 0x00000001, the host resends the appropriate REMOTE_NDIS_SET_MSG message to resend the addressing information.

3.1.5.6 Processing a REMOTE_NDIS_INDICATE_STATUS_MSG Message

The reason for the device indicating the status can be derived by examining the **Status** field. If the value is RNDIS_STATUS_INVALID_DATA, the request message that the device could not process MAY be examined and a corrective action MAY be taken by the host. If the value indicates a device status change, appropriate action as required by the host NDIS specification may be taken.

3.1.5.7 Processing a REMOTE_NDIS_HALT_MSG Message

If the host receives this message, irrespective of the value contained in the **RequestId** field, the host MUST free all the **resources** associated with this device, and the device and the host move to an **UNINITIALIZED** state. This message can be received only when the device and the host are in **RNDIS_INITIALIZED** or **RNDIS_DATA_INITIALIZED** states.

3.1.5.8 Processing a REMOTE_NDIS_PACKET_MSG Message

When the host receives this message, the packet data contained in the message, along with any out-of-band data and per-packet-info data, MUST be dealt with according to the host side network driver interface specification. The host MUST also examine if there is another REMOTE_NDIS_PACKET_MSG message immediately following this message, as multiple packets can be indicated together in a single bus transfer, and process the following message as if it is an independent data message until no more packets are found in the current bus message.

3.1.5.9 Processing a REMOTE_NDIS_KEEPALIVE_MSG Message

When the device implements the optional **KeepAliveTimer**, this message MAY be received when the device and the host are in **RNDIS_DATA_INITIALIZED** or **RNDIS_INITIALIZED** states and when there has not been any data or control message sent from host to device for the timeout period of the device's **KeepAliveTimer**. In response, the host MUST send a REMOTE_NDIS_KEEPALIVE_CMPLT message with the **Status** field set to RNDIS_STATUS_SUCCESS.

3.1.6 Timer Events

KeepAliveTimer Expiry: The host sends a REMOTE_NDIS_KEEPALIVE_MSG message to the device via the control channel in order to check the health of the device periodically. It is sent when the device is in **RNDIS_DATA_INITIALIZED** or **RNDIS_INITIALIZED** state and there has been no other control or data traffic from the device to the host for the **KeepAliveTimer's** timeout period. If the device does not respond within the timeout period, the host issues a REMOTE_NDIS_RESET_MSG message to the device. The **KeepAliveTimer's** timeout period is dependent on the underlying bus transport.

ControlTimeoutTimer Expiry: This timer is started on the host whenever a control message is sent to the device and the device MUST respond within the **ControlTimeoutTimer's** timeout period with the appropriate response message. If the host does not receive the corresponding response within the timeout period, it issues a REMOTE_NDIS_RESET_MSG message to the device. The timeout period depends on the characteristics of the underlying bus transport.

The host MUST also handle any underlying bus transport-mandated timer events.

3.1.7 Other Local Events

None.

3.2 Device Details

3.2.1 Abstract Data Model

The data model in section 3.1.1 is assumed. There is no additional device-specific data model.

3.2.2 Timers

KeepAliveTimer (optional): The device may optionally implement a **KeepAliveTimer** to check whether the host is active, when the device and the host are in **RNDIS_DATA_INITIALIZED** or **RNDIS_INITIALIZED** states, and when there has been no other control or data traffic from the host for the timeout period. The timeout period should suitably be OID-chosen, based on the underlying bus characteristics.

The device MUST also implement any timers mandated by the underlying bus transport.

3.2.3 Initialization

Device initialization happens in the context of processing a REMOTE_NDIS_INITIALIZE_MSG message. See section 3.2.5.1.

3.2.4 Higher-Layer Triggered Events

Change in the network link status results in the device sending a REMOTE_NDIS_INDICATE_STATUS_MSG message on the control channel. Also, the networking data arriving at the device to be sent to the host causes the device to send REMOTE_NDIS_PACKET_MSG messages on the data channel.

3.2.5 Message Processing Events and Sequencing Rules

Except for a REMOTE_NDIS_HALT_MSG message, the device MUST copy the **RequestID** field of the request message to the response message it sends to the host. In the event that the device fails to process a message or does not support a particular request, an appropriate REMOTE_NDIS_INDICATE_STATUS_MSG message MUST be sent to the host.

3.2.5.1 Processing a REMOTE_NDIS_INITIALIZE_MSG Message

This message is received only when the device and the host are in the **RNDIS_UNINITIALIZED** state. The device SHOULD initialize itself, adhere to the **MaxTransferSize** field value specified in this message in the subsequent communication with the host, and respond with appropriate the REMOTE_NDIS_INITIALIZE_CMPLT message.

3.2.5.2 Processing a REMOTE_NDIS_QUERY_MSG message

This message can be received by the device when the host and the device are in the **RNDIS_INITIALIZED** or **RNDIS_DATA_INITIALIZED** states. If the OID contained in the message is a mandatory OID or a device-supported optional OID, the device MUST respond with the appropriate REMOTE_NDIS_QUERY_CMPLT message, indicating the status of REMOTE_NDIS_QUERY_MSG request processing. Any input parameters required to service the OID MUST be read from the **InformationBuffer** field. Upon successful processing of the OID, the response data SHOULD be set in the **InformationBuffer** field of the REMOTE_NDIS_QUERY_CMPLT response message. If the OID is optional and not supported by the device, the **Status** field of the REMOTE_NDIS_QUERY_CMPLT response message MUST be set to the value RNDIS_STATUS_NOT_SUPPORTED.

3.2.5.3 Processing REMOTE_NDIS_SET_MSG message

This message can be received by the device when the host and the device are in **RNDIS_INITIALIZED** or **RNDIS_DATA_INITIALIZED** states. If the OID contained in the message is a mandatory OID or a device-supported optional OID, the device MUST respond with appropriate REMOTE_NDIS_SET_CMPLT message, indicating the status of processing the request. Any input parameters required to service the OID MUST be read from the **InformationBuffer** field. If the OID is optional and not supported by the device, the **Status** field of the REMOTE_NDIS_SET_CMPLT response message MUST be set to the value RNDIS_STATUS_NOT_SUPPORTED.

3.2.5.4 Processing a REMOTE_NDIS_KEEPALIVE_MSG Message

The host and the device MUST be in a **RNDIS_DATA_INITIALIZED** or **RNDIS_DATA_INITIALIZED** state when this message is received. The device MUST respond with a REMOTE_NDIS_KEEPALIVE_CMPLT message with the **Status** field value set to RNDIS_STATUS_SUCCESS if the device is working correctly or an error value to obtain a REMOTE_NDIS_RESET_MSG message from the host.

3.2.5.5 Processing a REMOTE_NDIS_RESET_MSG Message

This message may be received in either the **RNDIS_INITIALIZED** or **RNDIS_DATA_INITIALIZED** state. The device MUST soft-reset itself by discarding any outstanding requests or data packets but still keeping the communication channels intact. After the soft-reset is complete, the device MUST respond to the host with a REMOTE_NDIS_RESET_CMPLT message containing the appropriate status.

3.2.5.6 Processing a REMOTE_NDIS_HALT_MSG Message

This message can be received only when the host and the device are in **RNDIS_INITIALIZED** or **RNDIS_DATA_INITIALIZED** states. The device MUST terminate all communication with the host upon receiving this message. The device will not send any response to this message from the host, on processing this request. The host and the device enter a **RNDIS_UNINITIALIZED** state after receiving this message.

3.2.5.7 Processing a REMOTE_NDIS_PACKET_MSG Message

When the device receives this message, the packet data contained in the message, along with any out-of-band data and per-packet-info data, are consumed. The device MUST also examine if there is another REMOTE_NDIS_PACKET_MSG message immediately following this message, as multiple REMOTE_NDIS_PACKET_MSG messages MAY be bundled in a single bus transfer, and process the following message as if it is an independent data message, until no more packets are found in the current bus message.

3.2.5.8 Processing a REMOTE_NDIS_KEEPALIVE_CMPLT Message

When the device implements the optional KeepAliveTimer, this message SHOULD be processed as follows. The host and the device MUST be in **RNDIS_DATA_INITIALIZED** or

RNDIS_INITIALIZED states when this message is received. The device MUST reset its KeepAliveTimer in response.

3.2.6 Timer Events

KeepAliveTimer Expiry: The device periodically sends a REMOTE_NDIS_KEEPALIVE_MSG message to the host via the control channel to check if the host is active; in the **RNDIS_DATA_INITIALIZED** or **RNDIS_INITIALIZED** states if there has been no other control or data traffic from the host for the **KeepAliveTimer's** timeout period. The **KeepAliveTimer's** timeout period is dependent on the underlying bus.

The device MUST also handle any underlying bus transport mandated timer events.

3.2.7 Other Local Events

Device status change: Whenever there is a change in the state of the device, such as a link state change, the device indicates this change to the host by sending an appropriate REMOTE_NDIS_INDICATE_STATUS_MSG message.

Failure to interpret any message: If the device receives a message that is not a valid RNDIS message it can successfully receive, then it SHOULD send a REMOTE_NDIS_INDICATE_STATUS_MSG message with appropriate error code in the **Status** field along with a suitable **RNDIS_DIAGNOSTIC_INFO** with the received message in the **StatusBuffer** field.

4 Protocol Examples

4.1 Example of an OID Query and its Response

In the following example, the host has defined an OID named _OID_GEN_MEDIA_SUPPORTED(0x0000ABCD) to query the list of media the device supports, which can be a subset of the values {Medium802_3 (0x00000000), Medium802_5 (0x00000001), MediumWan (0x00000002), and MediumFDDI (0x00000003)}.

The device is required to return the supported media in the **InformationBuffer** field as an array of 32-bit values. Assuming that the device supports Medium802_3, the following packet diagrams represent the REMOTE_NDIS_QUERY_MSG message from the host and the corresponding successful REMOTE_NDIS_QUERY_CMPLT message from the device, respectively.

REMOTE_NDIS_QUERY_MSG for OID_GEN_MEDIA_SUPPORTED

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageType=0x00000004																															
MessageLength=0x0000001C																															
RequestID=0x0000012																															
OID=0x0000ABCD																															
InformationBufferLength=0x00000000																															
InformationBufferOffset=0x00000000																															
Reserved=0x00000000																															

REMOTE_NDIS_QUERY_CMPLT for OID_GEN_MEDIA_SUPPORTED

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageType=0x80000004																															
MessageLength=0x0000001C																															
RequestID=0x00000012																															
Status=0x00000000																															
InformationBufferLength=0x00000004																															
InformationBufferOffset=0x00000010																															

0x00000000

4.2 Example of a Multi-Packet Message

The following example shows the structure of a REMOTE_NDIS_PACKET_MSG message containing two packets, the first with a payload of 30 bytes and the second with a payload of 20 bytes. During initialization, through a REMOTE_NDIS_INITIALIZE_CMPLT message, the device specified the values of **MaxPacketsPerTransfer** to be 0x00000004, **MaxTransferSize** to be 0x00001000 (4096), and **PacketAlignmentFactor** to be 0x00000004. The **PacketAlignmentFactor** field indicates that 16-byte alignment is required between the multiple-packets, excluding the last, which requires 6-byte padding at the end of the first REMOTE_NDIS_PACKET_MSG message in the example.

Multi-packet REMOTE_NDIS_PACKET_MSG

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageType=0x00000001																															
MessageLength=0x00000050																															
DataOffset=0x00000024																															
DataLength=0x0000001E																															
OutOfBandDataOffset=0x00000000																															
OutOfBandDataLength=0x00000000																															
NumOutOfBandDataElements=0x00000000																															
PerPacketInfoOffset=0x00000000																															
PerPacketInfoLength=0x00000000																															
Reserved(low 32 bit)=0x00000000																															
Reserved(high 32 bit)=0x00000000																															
Payload(30 bytes)																															
Padding(6 bytes)																															
MessageType=0x00000001																															
MessageLength=0x00000040																															

DataOffset=0x00000024
DataLength=0x00000014
OutOfBandDataOffset=0x00000000
OutOfBandDataLength=0x00000000
NumOutOfBandDataElements=0x00000000
PerPacketInfoOffset=0x00000000
PerPacketInfoLength=0x00000000
Reserved(low 32 bit)=0x00000000
Reserved(high 32 bit)=0x00000000
Payload(20 bytes)

4.3 Example of RNDIS over USB

For an example of how RNDIS is mapped over the USB protocol, see [MSDN-RNDISUSB].

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft Windows® 2000 operating system
- Windows® XP operating system
- Windows Server® 2003 operating system
- Windows Vista® operating system
- Windows Server® 2008 operating system
- Windows® 7 operating system
- Windows Server® 2008 R2 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 2.2.2: Windows currently sets this value to 0x00000001.

<2> Section 2.2.2: Windows currently sets this value to 0x00000000.

<3> Section 2.2.2: Windows currently sets this value to 0x00004000.

<4> Section 2.2.4: See [MSDN-OIDs] for Windows-defined OIDs.

<5> Section 2.2.4: See [MSDN-OIDs] for the value to be used for the OID with Windows.

<6> Section 2.2.4: See [MSDN-OIDs] for the structure to be used for the OID with Windows.

<7> Section 2.2.5: See [MSDN-OIDs] for Windows-defined OIDs.

<8> Section 2.2.5: See [MSDN-OIDs] for the value to be used for the OID with Windows.

<9> Section 2.2.5: See [MSDN-OIDs] for the structure to be used for the OID with Windows.

<10> Section 2.2.7: See [MSDN-NDIS] for the value to be used for a specific status indication for device state change in Windows.

<11> Section 2.2.7: See [MSDN-NDIS] for the structure to be used for a specific status indication for device state change in Windows.

<12> Section 2.2.10: See [MSDN-OIDs] for the value to be used for the OID with Windows.

<13> Section 2.2.10: See [MSDN-OIDs] for the structure to be used for the OID with Windows.

<14> Section 2.2.14.1: See the documentation of NDIS_PACKET_OOB_DATA structure on [MSDN-NDIS] for Windows-specific details.

<15> Section 2.2.14.2: See the documentation of NDIS_PACKET_EXTENSION on [MSDN-NDIS] for Windows-specific details.

<16> Section 3.1.2: Windows uses 5 seconds as the timeout value on all supported versions of USB

<17> Section 3.1.2: Windows uses 8 seconds as the timeout value on all supported versions of the USB bus.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index