[MC-DPL4R]: DirectPlay 4 Protocol: Reliable Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Copyright © 2011 Microsoft Corporation.

Release: Friday, June 10, 2011

Revision Summary

Date	Revision History	Revision Class	Comments
08/10/2007	0.1	Major	Initial Availability
09/28/2007	0.2	Minor	Updated the technical content.
10/23/2007	0.2.1	Editorial	Revised and edited the technical content.
11/30/2007	1.0	Major	Updated and revised the technical content.
01/25/2008	2.0	Major	Updated and revised the technical content.
03/14/2008	3.0	Major	Updated and revised the technical content.
05/16/2008	3.0.1	Editorial	Revised and edited the technical content.
06/20/2008	3.1	Minor	Updated the technical content.
07/25/2008	3.1.1	Editorial	Revised and edited the technical content.
08/29/2008	3.1.2	Editorial	Revised and edited the technical content.
10/24/2008	3.2	Minor	Updated the technical content.
12/05/2008	3.2.1	Editorial	Editorial Update.
01/16/2009	3.2.2	Editorial	Revised and edited the technical content.
02/27/2009	4.0	Major	Updated and revised the technical content.
04/10/2009	4.0.1	Editorial	Revised and edited the technical content.
05/22/2009	4.1	Minor	Updated the technical content.
07/02/2009	4.1.1	Editorial	Revised and edited the technical content.
08/14/2009	4.1.2	Editorial	Revised and edited the technical content.
09/25/2009	4.2	Minor	Updated the technical content.
11/06/2009	4.2.1	Editorial	Revised and edited the technical content.
12/18/2009	4.2.2	Editorial	Revised and edited the technical content.
01/29/2010	5.0	Major	Updated and revised the technical content.
03/12/2010	5.0.1	Editorial	Revised and edited the technical content.
04/23/2010	5.0.2	Editorial	Revised and edited the technical content.
06/04/2010	6.0	Major	Updated and revised the technical content.
07/16/2010	7.0	Major	Significantly changed the technical content.

Copyright © 2011 Microsoft Corporation.

Date	Revision History	Revision Class	Comments
08/27/2010	7.1	Minor	Clarified the meaning of the technical content.
10/08/2010	7.1	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	7.1	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	7.1	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	7.1	No change	No changes to the meaning, language, or formatting of the technical content.
03/25/2011	7.1	No change	No changes to the meaning, language, or formatting of the technical content.
05/06/2011	7.1	No change	No changes to the meaning, language, or formatting of the technical content.
06/17/2011	7.2	Minor	Clarified the meaning of the technical content.

Copyright © 2011 Microsoft Corporation.

Contents

Copyright © 2011 Microsoft Corporation.

1 Introduction

This specification pertains to the DirectPlay 4 Protocol and describes functionality related to the reliable delivery of DirectPlay 4 messages. The DirectPlay 4 Protocol guarantees message delivery and provides throttling for applications that use DirectPlay 4.

1.1 Glossary

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

acknowledgment (ACK)
DirectPlay
DirectPlay 4
DirectX
DirectX runtime
DirectX Software Development Kit (DirectX SDK)
game
host
Internet Protocol security (IPsec)
Internetwork Packet Exchange (IPX)
maximum transmission unit (MTU)
peer
peer-to-peer mode
player
round-trip time (RTT)
sequence ID
server (3)
service provider
session layer
tick count
Transmission Control Protocol (TCP)
transport layer
User Datagram Protocol (UDP)
user message

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

game session
group
modem link
payload
serial link

The following terms are specific to this document:

throttling: The reduction in the rate of sending data when a network link saturation condition is detected.

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.

Copyright © 2011 Microsoft Corporation.

1.2 References

References to Microsoft Open Specification documents do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

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.

[MC-DPL4CS] Microsoft Corporation, "DirectPlay 4 Protocol: Core and Service Providers Specification".

[MS-DPDX] Microsoft Corporation, "DirectPlay DXDiag Usage Protocol Specification".

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

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

[US PATENT 6438603 B1] Ogus, A.W., "Methods and Protocol for Simultaneous Tuning of Reliable and Non-Reliable Channels of a Single Network Communication Link", 2002 Ogus http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=6438603.PN.&OS=PN/6438603&RS=PN/6438603

1.2.2 Informative References

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

1.3 Overview

This specification describes the reliable transport mechanism that may be used with the DirectPlay 4 Protocol. This mechanism provides for reliable delivery of messages, message **throttling**, and for applications written for the IDirectPlay4 interface. Early implementations of **DirectPlay 4** did not provide reliable delivery and throttling. As a result, in some scenarios, no reliable delivery services were available. The reliable transport mechanism combines the reliable and unreliable delivery of messages into a single channel of data. This facilitates a single set of flow control logic that provides both reliable and unreliable data delivery services, as specified in [US PATENT 6438603 B1].

The reliable transport mechanism used by the DirectPlay 4 Protocol is an envelope that encapsulates all messages sent between connected **peers** when active. That is, the mechanism becomes active when the normal DirectPlay 4 connection process has completed. All messages sent to the **host** during the connection process do not use the reliable transport mechanism.

The application determines whether the reliable transport mechanism is activated. Use of reliable transport is exclusive of the use of some of the normally available security functionality in DirectPlay 4. However, when reliable transport is activated, user-level security is not available.

Copyright © 2011 Microsoft Corporation.

1.4 Relationship to Other Protocols

The DirectPlay 4 Protocol is an envelope that wraps both operating system messages and **user messages**. This protocol is media-independent because it resides in the **session layer** of the protocol stack. **DirectPlay** typically can be run using **service providers** for **TCP**/IP, **Internetwork Package Exchange (IPX)**, modem, and serial links. From the perspective of service providers, this is an application-layer protocol. From the perspective of applications, the DirectPlay 4 Protocol is a session-layer protocol.

The DirectPlay 4 Protocol is transmitted via the TCP and **User Datagram Protocol (UDP)** protocols, as specified in the DirectPlay 4 Protocol: Core and Service Providers Specification [MC-DPL4CS]. At the discretion of the **game**, all of the messages listed in [MC-DPL4CS] may be transmitted via the DirectPlay 4 Protocol, as described in this specification [MC-DPL4R].

1.5 Prerequisites/Preconditions

The DirectPlay 4 Protocol requires the **DirectX 6 Runtime**.<1>

1.6 Applicability Statement

The DirectPlay 4 Protocol is activated only at the request of an application that is written for the IDirectPlay4 interface and written with the **DirectX 6 Software Development Kit** or a later version of the DirectX Software Development Kit (DirectX SDK). All of the functionality present in the DirectPlay 4 Protocol has been superseded by the **DirectPlay 8 Protocol** and, as such, the DirectPlay 4 Protocol is only to be used when the game has a requirement to interoperate with other DirectPlay 4 games.

1.7 Versioning and Capability Negotiation

There is only one version of the DirectPlay 4 Protocol. It is activated at the request of the application. It is assumed that the application has taken measures to ensure that the appropriate version of the DirectX Runtime has been installed on all peers involved in a DirectPlay **game session**.

If the versions of DirectPlay on all computing systems do not support the DirectPlay 4 Protocol, and the application has requested that the DirectPlay 4 Protocol be used, the connection process between peers fails gracefully. That is, a node with a version of the DirectX Runtime earlier than version 5 that is attempting to join a DirectPlay game session that has the DirectPlay 4 Protocol activated will be rejected during its join attempt.

For version negotiation between versions of DirectPlay, see [MC-DPL4CS].

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

Copyright © 2011 Microsoft Corporation.

2 Messages

In the DirectPlay 4 Protocol, the terms "frame" and "packet" are used interchangeably. Frames and packets refer to a single **payload** that is passed to a lower-layer transport, which is typically constrained by the **maximum transmission unit (MTU)** size of the network. Messages are higher-layer payloads that might be fragmented. Messages that do not fit in a single frame can span multiple frames.

The DirectPlay 4 Protocol supports three types of frames: Data, acknowledgment (ACK), and NACK.<2> The frame type is determined by the settings for the **ACK** (ACKNOWLEDGE) and **EXT** (EXTENSION) bits of the **flags** field, and the **nNACK** bits of the **extended flags** field in the packet.

Frame type	ACK bit	EXT bit	nNACK bits
Data Frame (section 2.2.1)	Not Set	Not Set	Not present
ACK Frame (section 2.2.3)	Set	Not Set	Not present
NACK Frame (section 2.2.2)	Set	Set	Non-zero

This protocol references commonly used data types as defined in [MS-DTYP].

2.1 Transport

Messages are transported over DirectPlay service providers. They can use UDP, IPX, serial, and modem, or a third-party service provider.

2.2 Message Syntax

2.2.1 Frame Format, Data Frame

Data frames are messages that deliver user-specified data. The Data frames format specifies message boundaries and sequencing. Each message is identified by a **messageid** and each part of the message is identified by a sequence number. Each unique instance of a piece of a message is uniquely identified by a serial number. Message serial numbers help statistics-gathering mechanisms to differentiate between original instances of a message fragment and their retries.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="8">flags</td><td colspan="8">messageid</td><td colspan="8">sequence</td><td colspan="8">serial</td></tr>
<tr><td colspan="8">extended flags (optional)</td><td colspan="24">NACK MASK (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">Data (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

flags (1 byte): A bitmask that contains values from the following table that are combined using the bitwise OR operation.

Copyright © 2011 Microsoft Corporation.

Note For a description of how these flags determine the frame type, see section 2.

Value	Meaning
0x80	(EXT or EXTENSION) Indicates that the extended flags field is present.
0x40	(BIG) Indicates that big frame formatted is used. This value MUST be 0 because the big frame format was never implemented.
0x20	(CMD or COMMAND) Indicates that this is a Data Frame. This bit MUST be set for Data Frames. If the EXT bit is also set, the packet MUST be ignored.
0x10	(STA or START) Indicates the start of a message that can span multiple protocol frames.
0x08	(EOM or END OF MESSAGE) Indicates the end of a message that can span multiple protocol frames.
0x04	(SAK or SEND ACKNOWLEDGE) Specifies a request for immediate **acknowledgment (ACK)** after this frame is received.
0x02	(ACK or ACKNOWLEDGE) Acknowledges receipt of some frames or the nonreceipt of frames if the extended flags are present. This bit MUST NOT be set for Data Frame messages.
0x01	(RLY or RELIABLE, ~UNRELIABLE) Indicates the message is reliable if set and unreliable if not set.

messageid (1 byte): A sequentially assigned value, starting at 1. The first message sent MUST be **messageid** 1. The second message sent is **messageid** 2, and so on. There MUST NOT be more than 24 outstanding **messageid**(s) on any link.

sequence (1 byte): The sequence number uniquely identifies the packet. It specifies the sequence space in which ACKs and NACKs are made.

serial (1 byte): For messages that do not span multiple packets and that set both the STA and EOM flags, the value of the serial field is the count of times this packet has been sent. If the value of serial is 0, this is the first instance of the packet. If the value of serial is 1, this is the first retry of this packet. For messages that span multiple packets, the value of serial begins at 0 for the first instance of the first packet in the message and is incremented for each additional packet in the message, as well as any retries of any of those packets.

extended flags (1 byte): This field is optional. It is present if the EXT bit is set in the **flags** field.<3>

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
COMMAND					nNACK		0																								

COMMAND (5 bits): MUST be 0.

nNACK (2 bits): The size, in bytes, of the **NACK MASK**.

Copyright © 2011 Microsoft Corporation.

Value	Meaning
0x00	There is no **NACK MASK** present.
0x01	There is 1 byte of **NACK MASK** present.
0x02	There are 2 bytes of **NACK MASK** present.
0x03	There are 3 bytes of **NACK MASK** present.

0 (1 bit): MUST be 0.

NACK MASK (variable): A **NACK MASK** is sent in any message header when packets expected to be in the message stream are not received. The mask of bits specifies which packets have not been received by the receiver.

Creation of the mask is based on the sequence number in the NACK packet, which identifies the first packet that was not received. The bits in the **NACK MASK** bitmask specify which other packets relative to that sequence number have also not been received. For example, if the sequence number is 7 and the **NACK MASK** is 0x03, then packets with sequence numbers 7, 8, and 9 have not been received at the receiving end of the link.

The size of the **NACK MASK** is specified in the **nNACK** bits of the **extended flags** field, if that field is present. When **nNACK** is greater than 0, there is a **NACK MASK**. When the **extended flags** field is not present, there is no **NACK MASK** field.

The **NACK MASK** field is optional.<4>

Data (variable): The higher-layer data payload. The length of this field MUST be inferred from the remaining size of the packet reported by the lower-level transport. The size is the total number of bytes in the packet minus the 4–8 bytes of previous Data frame fields.

2.2.2 Frame Format, NACK Frame

NACK, or Negative Acknowledge, frames specify which data frames were expected but were not received by the receiving end of the link. Messages that are not sent reliably MUST NOT generate a NACK.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="8">flags</td><td colspan="8">extended flags (optional)</td><td colspan="8">messageid</td><td colspan="8">sequence</td></tr>
<tr><td colspan="32">bytes received</td></tr>
<tr><td colspan="32">local tick count</td></tr>
<tr><td colspan="32">NACK MASK (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

flags (1 byte): A bitmask that contains values from the following table that are combined using the bitwise OR operation.

Copyright © 2011 Microsoft Corporation.

Note For a description of how these flags determine the frame type, see section 2.

Value	Meaning
0x80	(EXT or EXTENSION) Indicates that the extended flags field is present.
0x40	(BIG) Indicates that big frame format is used. This value MUST be 0 because the big frame format was never implemented.
0x20	(CMD or COMMAND) Indicates that this is a Data Frame. This bit MUST be set for Data Frames. If the EXT bit is also set, the packet MUST be ignored.
0x10	(STA or START) Indicates the start of a message that can span multiple protocol frames.
0x08	(EOM or END OF MESSAGE) Indicates the end of a message that can span multiple protocol frames.
0x04	(SAK or SEND ACKNOWLEDGE) Specifies a request for immediate acknowledgment after this frame is received.
0x02	(ACK or ACKNOWLEDGE) Acknowledges receipt of some frames; might specify the nonreceipt of frames if the extended flags are present.
0x01	(RLY or RELIABLE, ~UNRELIABLE) Indicates that the message is reliable if set and unreliable if not set.

extended flags (1 byte): This field is optional. It is present if the EXT bit is set in the **flags** field.<5>

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="5">COMMAND</td><td colspan="2">nNACK</td><td>0</td></tr>
</table>

COMMAND (5 bits): MUST be 0x00.

nNACK (2 bits): The size, in bytes, of the **NACK MASK**.

Value	Meaning
0x00	There is no **NACK MASK** present.
0x01	There is 1 byte of **NACK MASK** present.
0x02	There are 2 bytes of **NACK MASK** present.
0x03	There are 3 bytes of **NACK MASK** present.

0 (1 bit): MUST be 0.

messageid (1 byte): The **messageid** from the sent data frame.

sequence (1 byte): The **sequence ID** from the sent data frame. The sequence uniquely identifies which is the first nonreceived frame.

Copyright © 2011 Microsoft Corporation.

bytes received (4 bytes): Specifies the total number of bytes received on the link at the time that this NACK frame was sent.

local tick count (4 bytes): Specifies the **tick count** on the local tick clock when the NACK frame was sent.

NACK MASK (variable): A **NACK MASK** is sent in any message header when packets expected to be in the message stream are not received. The mask of bits specifies which packets have not been received by the receiver.

Creation of the mask is based on the sequence number in the NACK packet, which identifies the first packet that was not received. The bits in the **NACK MASK** bitmask specify which other packets relative to that sequence number have also not been received. For example, if the sequence number is 7 and the **NACK MASK** is 0x03, then packets with sequence numbers 7, 8, and 9 have not been received at the receiving end of the link.

The size of the **NACK MASK** is specified in the **nNACK** bits of the **extended flags** field, if that field is present. When **nNACK** is greater than 0, there is a **NACK MASK**. When the **extended flags** field is not present, there is no **NACK MASK** field.

The **NACK MASK** field is optional.<6>

2.2.3 Frame Format, ACK Frame

Acknowledgment (ACK) frames specify which data frames have successfully arrived at the receiving end of the link.

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
flags								messageid								sequence								serial							
bytes received																															
local tick count																															

flags (1 byte): A bitmask that contains values from the following table that are combined using the bitwise OR operation.

Note For a description of how these flags determine the frame type, see section 2.

Value	Meaning
0x80	(EXT or EXTENSION) This value MUST be set to 0.
0x40	(BIG) Indicates that big frame format is used. This value MUST be 0 because the big frame format was never implemented.
0x20	(CMD or COMMAND) Indicates that this is a Data Frame. This bit MUST be set for Data Frames. If the EXT bit is also set, the packet MUST be ignored.
0x10	(STA or START) Indicates the start of a message that can span multiple protocol frames.
0x08	(EOM or END OF MESSAGE) Indicates the end of a message that can span multiple

Copyright © 2011 Microsoft Corporation.

Value	Meaning
	protocol frames.
0x04	(SAK or SEND ACKNOWLEDGE) Specifies a request for immediate acknowledgment after this frame is received.
0x02	(ACK or ACKNOWLEDGE) Acknowledges receipt of some frames; might specify the nonreceipt of frames if the extended flags are present.
0x01	(RLY or RELIABLE, ~UNRELIABLE) Indicates that the message is reliable if set and unreliable if not set.

messageid (1 byte): The **messageid** from the sent data frame.

sequence (1 byte): The sequence ID from the sent data frame. The sequence uniquely identifies which is the last successfully received frame. Acknowledgment of a particular sequence also acknowledges receipt of all prior unacknowledged sequence numbers.

serial (1 byte): The number of times this packet has been sent. If the serial number is 0, this is the first instance of the packet. If the serial number is 1, this is the first retry of this packet.

bytes received (4 bytes): Specifies the total number of bytes received on the link at the time this ACK frame was sent.

local tick count (4 bytes): Specifies the tick count on the local tick clock when the ACK frame was sent.

Copyright © 2011 Microsoft Corporation.

3 Protocol Details

The DirectPlay 4 Protocol is inherently **peer-to-peer**. It was also designed to ride on top of existing connection and game session logic of the existing DirectPlay 4 implementation. Therefore, it needs no explicit connection logic or game session management logic because that is handled by the DirectPlay 4 core, as specified in [MC-DPL4CS].

3.1 Common Details

3.1.1 Abstract Data Model

First Message: The **messageid** of the first reliable outstanding message on the outbound link. A message is outstanding if it has been transmitted but not acknowledged.

Last Message: The **messageid** of the last reliable outstanding message on the outbound link.

Outstanding Message Mask: A mask of bits that specifies those messages between First Message and Last Message that have not been acknowledged.

First Unreliable Message: The **messageid** of the first outstanding message on the outbound link that is not marked as reliable.

Last Unreliable Message: The **messageid** of the last outstanding message on the outbound link that is not marked as reliable.

Unreliable Outstanding Message Mask: A mask of bits that specifies those messages between First Unreliable Message and Last Unreliable Message that have not been acknowledged.

Unreliable Receive Queue: A queue for assembling parts of unreliable messages before they are finally assembled and indicated.

Reliable Receive Queue: A queue for assembling reliable messages and maintaining their order for indication.

First Reliable Receive: The first **messageid** in the reliable receive queue.

Last Reliable Receive: The last **messageid** in the reliable receive queue.

Reliable Receive Mask: A mask of bits that indicates which **messageid**s relative to First Reliable Receive are reliable.

Bytes Received: Count of bytes received on this link.

Average Latency: Average latency of messages sent on this link.

3.1.2 Timers

The DirectPlay 4 Protocol utilizes the computing system specified in [US PATENT 6438603 B1] for estimating available bandwidth and tuning the link appropriately. However, using the scheme described in the patent is not required to achieve interoperability.

Timers for retries can be set according to the expected delivery time and acknowledge time, plus some amount of extra time to allow for inconsistencies in link operation.

The current implementation sets the retry time-out to the average **round-trip** plus three standard deviations in the average round-trip. A simpler method may also be used, such as the average

Copyright © 2011 Microsoft Corporation.

round-trip plus an extra percentage, such as 20%. When no messages have previously been sent, the initial starting value can be set to 1 second.

3.1.3 Initialization

All sequence numbers MUST start at 1. All serial numbers MUST start at 0.

The count of bytes received SHOULD correspond as closely as possible to the total bytes received from the sender on the receiver's link, and SHOULD include the duplicate (retransmitted) packets.

The tick count is a 32-bit value counting 1/1000ths of a second of elapsed time. It can start at any value but MUST increase at the specified rate.

NACKs are not triggered, but only occur when missing frames are detected.

Note However, an implementation can attempt to trigger NACKs using a timer-based algorithm approach.

ACKs are triggered by either the SAK bit in the message header or by the end of the message.

ACKs MAY wait for a timer before sending, unless the SAK bit has been specified by the sender.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Processing Events and Sequencing Rules

3.1.5.1 Data Frame Processing

When a Data frame (section 2.2.1) arrives, the value of its **messageid** field MUST be compared to the list of receiving messages using 8-bit unsigned integer math, and according to the following rules:

Note When processing Data frames, the receiver MUST keep track of the last **messageid** that was received reliably.

- If there are no ongoing receives or completed receives, then this **messageid** MUST become the first ongoing receive.
- If the value of the **messageid** field is outside the range of the **messageid** of the earliest ongoing or completed receive (and that **messageid**+23), the Data frame MUST be ignored.
- If no message with the **messageid** value is currently being received, the **messageid** of this message MUST be added to the list of receiving messages.

The only exception to this rule is when a reliable Data frame is received and the value of its **messageid** is the same as the last reliable message received. In this case, an acknowledgment (ACK) message MUST be sent back to the sender. This exception handles the situation where an ACK that was sent for a previously received message was lost. Because a message that was previously received is now being received again, it is possible that the ACK that was previously sent was lost and needs to be retransmitted.

When the **messageid** of a message is initially added to the list of receiving messages, a new message buffer MAY be allocated for the message. This buffer is used for assembling the message according to the sequence numbers associated with the **messageid** as necessary. When all of the

Copyright © 2011 Microsoft Corporation.

following Data frames have been received, the Data (variable) areas of those messages are concatenated in sequence order to make up the receive data for that message:

- A Data frame with a particular **messageid**, a starting sequence, and a STA bit set.
- A Data frame with the same **messageid**, an ending sequence, and an EOM set.
- All Data frames with the same **messageid** in the contiguous intervening sequence space.

Note Using 8-bit unsigned integer math, if a message is not received reliably and not all parts of the message have been received, and the currently receiving **messageid** is equal to the **messageid**+24 of this unreliable message, then the unreliable message SHOULD be discarded.

During assembly of the receive data, a NACK or ACK MUST be sent in the following circumstances:

- If the sequence number is not the current expected sequence number, a NACK packet MUST be sent back to the sender. It is at the discretion of the implementation to send the NACK immediately or to wait for more time or more receive data before sending the NACK.<7> Delaying the NACK may result in efficiency gains.
- If the Data frame has its SAK bit set, an ACK or NACK MUST be sent back to the sender without waiting for more time to elapse or more data to be received.<8>
- When the final part in a multipart message is received, an ACK MUST be sent back to the sender without waiting for more time to elapse or more data to be received.

After the receive data is fully assembled, the data is indicated to the higher layers, and the **messageid** of the message MUST be removed from the list of receiving messages.

3.1.5.2 ACK and NACK Processing

When an ACK frame (section 2.2.3) arrives, the data associated with the **messageid** in the acknowledgment (ACK) message and any earlier **messageids** MAY be discarded.

When a NACK frame (section 2.2.2) arrives, the data associated with the **messageid**-1 in the NACK frame (section 2.2.2) MAY be discarded. Any other data is indicated as received, but the presence of a 0 bit in the NACK mask may also be discarded. Any other data MUST be retransmitted.

3.1.5.3 Bytes Received Processing

Any time that data arrives from a remote sender, the size of the entire Data Frame (section 2.2.1) is accumulated in a per-link value called bytes received. This value is included in any acknowledgment (ACK) or NACK message on that link.

3.1.6 Timer Events

Timers are recommended as specified in [US PATENT 6438603 B1]. However, this scheme is not required to achieve interoperability.

3.1.7 Other Local Events

None.

Copyright © 2011 Microsoft Corporation.

4 Protocol Examples

4.1 One-Way Traffic Between Node A and Node B

In the following examples, one-way traffic between Node A and Node B is covered. Node A sends four messages to Node B:

- Message 1: Three Data frames (section 2.2.1) in length, marked as reliable.
- Message 2: One Data frame in length, not marked as reliable.
- Message 3: One Data frame in length, marked as reliable, will drop in first transmission.
- Message 4: Two Data frames in length, not marked as reliable.

4.1.1 Message 1

Node A sends Message 1 to the receiver, Node B. This message is three Data frames (section 2.2.1) in length and is marked as reliable.

```
Message 1:  DATA FRAME: 1
Flags:      STA, RLY messageid: 0, sequence: 0, serial: 0
Message 1:  DATA FRAME: 2
Flags:      RLY messageid: 0, sequence: 1, serial: 1
Message 1:  DATA FRAME: 3
Flags:      EOM, RLY messageid: 0, sequence: 2, serial: 2
```

All three Data frames arrive on the receiver, Node B, and the receiver indicates this to the higher layers.

The receiver sends back an ACK frame to the sender, Node A.

```
Flags:      ACK messageid: 0, sequence: 2, bytes received: 4500
```

4.1.2 Message 2

Node A sends Message 2 to the receiver, Node B. This message is one Data frame (section 2.2.1) in length and is not marked as reliable.

```
Message 2:  DATA FRAME
Flags:      STA, EOM messageid: 1, sequence: 3, serial: 0
```

It is possible that there can be an ACK by the receiver for a frame that is not marked as reliable; this is at the discretion of the implementation. However, sending an ACK allows the sender, Node A, to manage resources more efficiently.

4.1.3 Message 3

Node A sends Message 3 to the receiver, Node B. This message is one Data frame (section 2.2.1) in length, is marked as reliable, and will drop the first transmission.

Copyright © 2011 Microsoft Corporation.

```
Message 3:  DATA FRAME
Flags:      STA, EOM, RLY messageid: 2, sequence: 4, serial:0
```

If Message 3 is dropped, but the sender, Node A, still sends Message 4 (section 4.1.4) to the receiver, Node B, this would result in the following.

```
Message 4:  DATA FRAME: 1
Flags:      STA messageid: 3, sequence: 5, serial: 0
Message 4:  DATA FRAME: 2
Flags:      EOM messageid: 3, sequence: 6, serial: 1
```

This can trigger a NACK on the receiver, Node B.

The receiver, Node B, sends back a NACK to the sender, Node A.

```
Flags:      NACK messageid: 2, sequence: 4,
            nNACK: 0, no NACK mask present, bytes received 8954
```

In response to the NACK, the sender, Node A, attempts a retransmission of Message 3 to Node B.

```
Message 3:  DATA FRAME (retry)
Flags:      STA, EOM, RLY messageid: 2, sequence: 4, serial: 1
```

4.1.4 Message 4

The sender, Node A, sends Message 4 to the receiver, Node B. This message is two Data frames (section 2.2.1) in length and is not marked as reliable.

```
Message 4:  DATA FRAME: 1
Flags:      STA messageid: 3, sequence: 5, serial: 0
Message 4:  DATA FRAME: 2
Flags:      EOM messageid: 3, sequence: 6, serial: 0
```

This can trigger a NACK on the receiver. For more information, see section 4.1.3.

Copyright © 2011 Microsoft Corporation.

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

Copyright © 2011 Microsoft Corporation.

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 NT® operating system
- 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 1.5: The DirectPlay 4 Protocol is available only in Windows operating systems that have the DirectX 6 runtime (from the DirectX 6 Software Development Kit (DirectX SDK)) or a later version of the runtime installed. All Microsoft products listed above meet this requirement.

<2> Section 2: NACK frames are not implemented in Windows DirectPlay 4 implementation.

<3> Section 2.2.1: The Windows DirectPlay 4 implementation does not support extended flags and therefore does not set or expect to receive messages with the EXT bit set.

<4> Section 2.2.1: Because the extended flags field is not supported by Windows implementations, the **NACK MASK** field is also not supported

<5> Section 2.2.2: The Windows DirectPlay 4 implementation does not support extended flags and therefore does not set or expect to receive messages with the EXT bit set.

<6> Section 2.2.2: Because the extended flags field is not supported by Windows implementations, the **NACK MASK** field is also not supported.

<7> Section 3.1.5.1: The Windows DirectPlay 4 implementation sends NACKs without waiting for more time to elapse or more data to be received.

<8> Section 3.1.5.1: The Windows DirectPlay 4 implementation sets the SAK bit when any one of the following applies:

- The EOM bit is not set or the higher layer requested the message to be sent using reliable delivery, and the time since the last SAK request is greater than 50 milliseconds and greater than one quarter of the estimated round-trip latency of the connection.

Copyright © 2011 Microsoft Corporation.

- The packet is being retransmitted.
- There have been 24 packets since the last packet was sent with the SAK bit set.

Copyright © 2011 Microsoft Corporation.

7 Change Tracking

This section identifies changes that were made to the [MC-DPL4R] protocol document between the May 2011 and June 2011 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

Copyright © 2011 Microsoft Corporation.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1.2 References	Added explanatory statement regarding the removal of the publishing year from Microsoft Open Specification document references.	N	Content updated.

Copyright © 2011 Microsoft Corporation.

8 Index

Copyright © 2011 Microsoft Corporation.