

## [MS-MSBD-Diff]:

# Media Stream Broadcast Distribution (MSBD) Protocol

---

### Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation (~~“this documentation”~~) for protocols, file formats, data portability, computer languages, and standards ~~as well as overviews of the interaction among each of these technologies~~ support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **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~~ can make copies of it in order to develop implementations of the technologies ~~that are~~ described in ~~the Open Specifications~~ this documentation and ~~may~~ can distribute portions of it in your implementations ~~using~~ that use these technologies or ~~in~~ your documentation as necessary to properly document the implementation. You ~~may~~ can also distribute in your implementation, with or without modification, any ~~schema, IDL's~~ schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications- documentation.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that ~~may~~ might cover your implementations of the technologies described in the Open Specifications- documentation. Neither this notice nor Microsoft's delivery of ~~the~~ this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open ~~Specification may~~ Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in ~~the Open Specifications~~ this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting [iplg@microsoft.com](mailto:iplg@microsoft.com).
- **Trademarks.** The names of companies and products contained in this documentation ~~may~~ might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit [www.microsoft.com/trademarks](http://www.microsoft.com/trademarks).
- **Fictitious Names.** The example companies, organizations, products, domain names, ~~e-mail~~ email addresses, logos, people, places, and events ~~that are~~ 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 as specifically described above, whether by implication, estoppel, or otherwise.

**Tools.** The Open Specifications ~~documentation~~ does 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 documents are intended for use in conjunction with publicly available ~~standard~~ standards specifications and network programming art, ~~and~~ assumes, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

## Revision Summary

Date	Revision History	Revision Class	Comments
10/22/2006	0.01	<u>New</u>	Version 0.01 release
1/19/2007	1.0	<u>Major</u>	Version 1.0 release
3/2/2007	1.1	<u>Minor</u>	Version 1.1 release
4/3/2007	1.2	<u>Minor</u>	Version 1.2 release
5/11/2007	1.3	<u>Minor</u>	Version 1.3 release
6/1/2007	2.0	Major	Updated and revised the technical content.
7/3/2007	2.0.1	Editorial	Changed language and formatting in the technical content.
7/20/2007	3.0	Major	Updated and revised the technical content.
8/10/2007	4.0	Major	Updated and revised the technical content.
9/28/2007	4.1	Minor	Clarified the meaning of the technical content.
10/23/2007	5.0	Major	Made revisions for maximum field.
11/30/2007	5.1	Minor	Clarified the meaning of the technical content.
1/25/2008	5.1.1	Editorial	Changed language and formatting in the technical content.
3/14/2008	5.1.2	Editorial	Changed language and formatting in the technical content.
5/16/2008	5.1.3	Editorial	Changed language and formatting in the technical content.
6/20/2008	5.1.4	Editorial	Changed language and formatting in the technical content.
7/25/2008	5.1.5	Editorial	Changed language and formatting in the technical content.
8/29/2008	5.1.6	Editorial	Changed language and formatting in the technical content.
10/24/2008	5.1.7	Editorial	Changed language and formatting in the technical content.
12/5/2008	5.1.8	Editorial	Changed language and formatting in the technical content.
1/16/2009	5.1.9	Editorial	Changed language and formatting in the technical content.
2/27/2009	5.1.10	Editorial	Changed language and formatting in the technical content.
4/10/2009	5.1.11	Editorial	Changed language and formatting in the technical content.
5/22/2009	5.1.12	Editorial	Changed language and formatting in the technical content.
7/2/2009	5.1.13	Editorial	Changed language and formatting in the technical content.
8/14/2009	5.1.14	Editorial	Changed language and formatting in the technical content.
9/25/2009	5.2	Minor	Clarified the meaning of the technical content.
11/6/2009	5.2.1	Editorial	Changed language and formatting in the technical content.
12/18/2009	5.2.2	Editorial	Changed language and formatting in the technical content.
1/29/2010	5.3	Minor	Clarified the meaning of the technical content.

<b>Date</b>	<b>Revision History</b>	<b>Revision Class</b>	<b>Comments</b>
3/12/2010	5.3.1	Editorial	Changed language and formatting in the technical content.
4/23/2010	5.3.2	Editorial	Changed language and formatting in the technical content.
6/4/2010	5.3.3	Editorial	Changed language and formatting in the technical content.
7/16/2010	5.3.3	None	No changes to the meaning, language, or formatting of the technical content.
8/27/2010	5.3.3	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2010	5.3.3	None	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	5.3.3	None	No changes to the meaning, language, or formatting of the technical content.
1/7/2011	5.3.3	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2011	5.3.3	None	No changes to the meaning, language, or formatting of the technical content.
3/25/2011	5.3.3	None	No changes to the meaning, language, or formatting of the technical content.
5/6/2011	5.3.3	None	No changes to the meaning, language, or formatting of the technical content.
6/17/2011	5.4	Minor	Clarified the meaning of the technical content.
9/23/2011	5.4	None	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	5.4	None	No changes to the meaning, language, or formatting of the technical content.
3/30/2012	5.4	None	No changes to the meaning, language, or formatting of the technical content.
7/12/2012	5.4	None	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	5.4	None	No changes to the meaning, language, or formatting of the technical content.
1/31/2013	5.4	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	5.4	None	No changes to the meaning, language, or formatting of the technical content.
11/14/2013	5.4	None	No changes to the meaning, language, or formatting of the technical content.
2/13/2014	5.4	None	No changes to the meaning, language, or formatting of the technical content.
5/15/2014	5.4	None	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
6/30/2015	5.4	<del>No</del> ChangeNone	No changes to the meaning, language, or formatting of the technical content.
10/16/2015	5.4	<del>No</del> ChangeNone	No changes to the meaning, language, or formatting of the technical content.

# Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>7</b>
1.1	Glossary .....	7
1.2	References .....	7
1.2.1	Normative References .....	7
1.2.2	Informative References .....	8
1.3	Overview .....	8
1.4	Relationship to Other Protocols .....	8
1.5	Prerequisites/Preconditions .....	8
1.6	Applicability Statement .....	8
1.7	Versioning and Capability Negotiation .....	8
1.8	Vendor-Extensible Fields .....	9
1.9	Standards Assignments.....	9
<b>2</b>	<b>Messages.....</b>	<b>10</b>
2.1	Transport.....	10
2.2	Message Syntax.....	10
2.2.1	MSBMSGBASE.....	10
2.2.2	MSB_MSG_RES_PING .....	11
2.2.3	MSB_MSG_REQ_PING .....	11
2.2.4	MSB_MSG_REQ_STREAMINFO.....	12
2.2.5	MSB_MSG_RES_STREAMINFO .....	12
2.2.6	MSB_MSG_IND_STREAMINFO .....	14
2.2.7	MSB_MSG_REQ_CONNECT.....	16
2.2.8	MSB_MSG_RES_CONNECT.....	16
2.2.9	MSB_MSG_IND_EOS.....	17
2.2.10	MSB_MSG_IND_PACKET.....	18
<b>3</b>	<b>Protocol Details.....</b>	<b>20</b>
3.1	Server Details.....	20
3.1.1	Abstract Data Model.....	20
3.1.2	Timers .....	20
3.1.3	Initialization.....	20
3.1.4	Higher-Layer Triggered Events .....	21
3.1.5	Message Processing Events and Sequencing Rules .....	21
3.1.5.1	Receiving an MSB_MSG_REQ_CONNECT Packet.....	21
3.1.5.2	Receiving an MSB_MSG_REQ_STREAMINFO Packet .....	21
3.1.5.3	Reaching the End of the Stream.....	21
3.1.6	Timer Events.....	21
3.1.7	Other Local Events.....	21
3.2	Client Details.....	21
3.2.1	Abstract Data Model.....	22
3.2.2	Timers .....	22
3.2.3	Initialization.....	22
3.2.4	Higher-Layer Triggered Events .....	22
3.2.5	Message Processing Events and Sequencing Rules .....	23
3.2.5.1	Receiving an MSB_MSG_REQ_PING Packet.....	23
3.2.5.2	Receiving an MSB_MSG_IND_STREAMINFO Packet.....	23
3.2.6	Timer Events.....	23
3.2.7	Other Local Events.....	23
<b>4</b>	<b>Protocol Example.....</b>	<b>24</b>
4.1	General MSBD Sequence.....	24
4.2	Using MSBD for Server-Side Playlist Streaming .....	25
<b>5</b>	<b>Security.....</b>	<b>27</b>
5.1	Security Considerations for Implementers .....	27

5.2	Index of Security Parameters .....	27
<b>6</b>	<b>Appendix A: Product Behavior .....</b>	<b>28</b>
<b>7</b>	<b>Change Tracking.....</b>	<b>29</b>
<b>8</b>	<b>Index.....</b>	<b>30</b>

# 1 Introduction

The Media Stream Broadcast Distribution (MSBD) Protocol is a protocol that is used for transferring an audio-visual content stream from a server to a single client or multiple clients.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative ~~and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in [RFC2119]. Sections 1.5 and 1.9 are also normative but do not contain those terms.~~ All other sections and examples in this specification are informative.

## 1.1 Glossary

~~The~~This document uses the following terms ~~are specific to this document:~~

**.nsc file:** A file that serves as an announcement for, and contains information about, a media stream broadcast. This file allows a client to tune in to a broadcast. The .nsc file was originally known as a NetShow Station Configuration file. Because the NetShow protocol suite is now obsolete, the original nomenclature is no longer applicable and is not used. Also known as a Windows Media Station file or an NSC file.

**Advanced Systems Format (ASF):** An extensible file format that is designed to facilitate streaming digital media data over a network. This file format is used by Windows Media.

**HRESULT:** An integer value that indicates the result or status of an operation. A particular HRESULT can have different meanings depending on the protocol using it. See [MS-ERREF] section 2.1 and specific protocol documents for further details.

**little-endian:** Multiple-byte values that are byte-ordered with the least significant byte stored in the memory location with the lowest address.

**parity packet:** An **ASF** data packet that contains parity data and is used for reconstructing other lost packets. Unlike other **ASF** data packets, parity packets always have the Opaque Data Present bit set to 1 in the **ASF** data packet header.

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

## 1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

### 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.

[ASF] Microsoft Corporation, "Advanced Systems Format Specification", December 2004, [http://download.microsoft.com/download/7/9/0/790feca-f64a-4a5e-a430-0bccdab3f1b4/ASF\\_Specification.doc](http://download.microsoft.com/download/7/9/0/790feca-f64a-4a5e-a430-0bccdab3f1b4/ASF_Specification.doc)

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

[MS-ERREF] Microsoft Corporation, "Windows Error Codes".

[MS-MSB] Microsoft Corporation, "Media Stream Broadcast (MSB) Protocol".

[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>

## 1.2.2 Informative References

None.

## 1.3 Overview

The MSBD Protocol is used for transferring a live stream of audio-visual content from a server to a single client or to multiple clients. For example, the MSBD Protocol might be used when transmitting the digitized sound and images of a lecture from a computer that is encoding the lecture in real time to another computer that is running the appropriate streaming media server software, such as Windows Media Services version 4.1.

The MSBD Protocol can also be used for transmitting a live stream from one Windows Media Services server installation to another. Certain versions of Windows Media Player<1> also support the MSBD Protocol and can use it to monitor a live stream by connecting to a server that is running Windows Media Services or by connecting directly to a real-time encoder.

## 1.4 Relationship to Other Protocols

MSBD packets are encapsulated in TCP. However, one MSBD packet type, MSB\_MSG\_IND\_PACKET, can also be encapsulated in the User Datagram Protocol (UDP). UDP encapsulation mode is only used for transmitting packets to an IPv4 multicast group.<2>

## 1.5 Prerequisites/Preconditions

The MSBD Protocol does not provide a mechanism for a client to discover the URL to the server. Therefore, it is a prerequisite that the client obtain a URL to the server before this protocol can be used.

## 1.6 Applicability Statement

The MSBD Protocol can be used to distribute **Advanced Systems Format (ASF)** packets over an IP-based network. The MSBD Protocol is designed for distribution of content between servers or from an encoder to a server.<3> The MSBD Protocol is not intended for use by an end-user application, such as Windows Media Player, except for troubleshooting use.

The UDP encapsulation mode of this protocol might not be suitable for content that uses large ASF data packets. Large ASF data packets **may** cause the UDP packets to be fragmented into multiple IP datagrams, and fragmentation of IP datagrams might be undesirable. When the content uses these large packets, TCP is the recommended encapsulation mode.

## 1.7 Versioning and Capability Negotiation

The MSBD Protocol does not support negotiation of versioning or capabilities. However, server implementations of the MSBD Protocol are not required to support delivery of MSB\_MSG\_IND\_PACKET packets over UDP. If a client requests such delivery, and the server does not support it, the server sends an MSB\_MSG\_RES\_CONNECT packet to indicate that the operation failed.



## 1.8 Vendor-Extensible Fields

This protocol uses **HRESULT** values as defined in [MS-ERREF]. Vendors can define their own HRESULT values, provided that they set the C bit (0x20000000) for each vendor-defined value, indicating that the value is a customer code.

## 1.9 Standards Assignments

The MSBD Protocol has no applicable standards assignments.

## 2 Messages

The following sections specify how MSBD Protocol messages are transported and give details on MSBD Protocol message syntax.

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

### 2.1 Transport

MSB\_MSG\_IND\_PACKET packets are encapsulated in UDP packets if IP multicasting is used to deliver those packets. Otherwise, the MSB\_MSG\_IND\_PACKET packets are encapsulated in TCP. All other MSBD Protocol packets MUST be encapsulated in TCP. The UDP and TCP packets MUST be encapsulated in the IP version 4 (IPv4).

There is no officially assigned TCP port for the MSBD Protocol; however, MSBD servers often listen on TCP port 7007 for client requests.

### 2.2 Message Syntax

All 16-bit and 32-bit integer fields MUST be transmitted in **little-endian** byte order, unless otherwise specified.

All fields that are unsigned 16-bit integers have valid ranges from 0x0000 to 0xFFFF, inclusive, unless otherwise specified.

All fields that are unsigned 32-bit integers have valid ranges from 0x00000000 to 0x0000FFFF, inclusive, unless otherwise specified.

#### 2.2.1 MSBMSGBASE

The MSBMSGBASE header defines the packet header.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
dwSignature																															
wVersion																wMessageId															
cbMessage																															
hr																															

**dwSignature (4 bytes):** An unsigned 32-bit integer that identifies the protocol as the MSBD Protocol. This value MUST be 0x2042534D, which is the little-endian representation of the ASCII character sequence "MSB[space]".

**wVersion (2 bytes):** An unsigned 16-bit integer that represents the version number of the MSBD Protocol. This value is currently 0x0106.

**wMessageId (2 bytes):** An unsigned 16-bit integer that designates the MSBD packet type. This value MUST be one of the following.

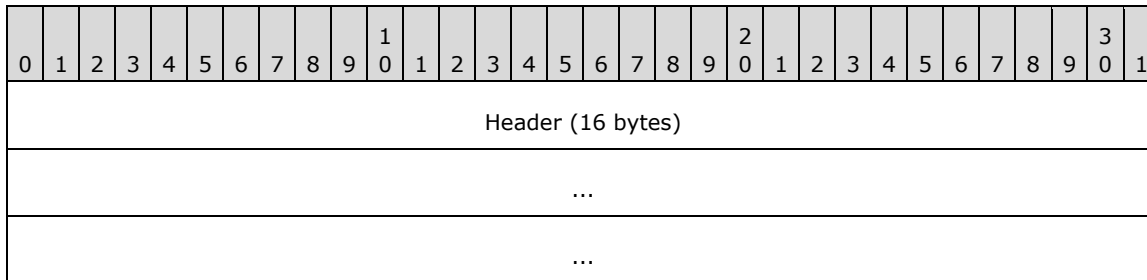
Value	Meaning
0x0001	Designates an MSB_MSG_REQ_PING packet.
0x0002	Designates an MSB_MSG_RES_PING packet.
0x0003	Designates an MSB_MSG_REQ_STREAMINFO packet.
0x0004	Designates an MSB_MSG_RES_STREAMINFO packet.
0x0005	Designates an MSB_MSG_IND_STREAMINFO packet.
0x0007	Designates an MSB_MSG_REQ_CONNECT packet.
0x0008	Designates an MSB_MSG_RES_CONNECT packet.
0x0009	Designates an MSB_MSG_IND_EOS packet.
0x000A	Designates an MSB_MSG_IND_PACKET packet.

**cbMessage (4 bytes):** An unsigned 32-bit integer that contains the byte length of the packet, including the header. The **cbMessage** field MUST be set to a value in the range 0x0010 to 0xFFFF, inclusive.

**hr (4 bytes):** An unsigned 32-bit integer field representing the status of an operation. The **hr** field MUST be set to an HRESULT code. For HRESULT codes, see [MS-ERREF] section 2.1.

### 2.2.2 MSB\_MSG\_RES\_PING

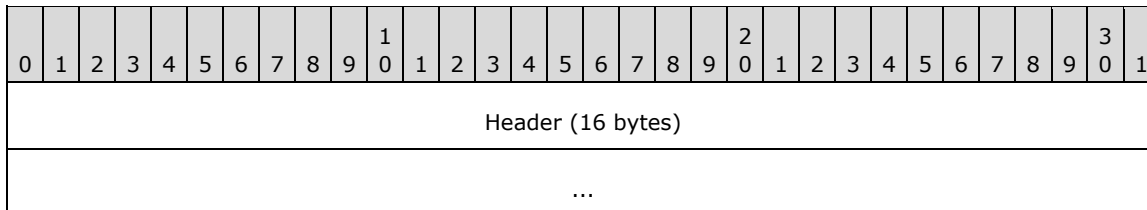
The client MUST respond with the MSB\_MSG\_RES\_PING packet to the MSB\_MSG\_REQ\_PING packet that is sent by the server. This packet informs the server that the client is still active.



**Header (16 bytes):** An MSBMSGBASE packet that defines the packet header. The **wMessageId** field MUST be set to 0x0002.

### 2.2.3 MSB\_MSG\_REQ\_PING

The server MAY send an MSB\_MSG\_REQ\_PING packet to confirm that the client is still active. It is recommended that the server send this packet approximately every 2 minutes. If active, the client MUST respond with an MSB\_MSG\_RES\_PING packet; otherwise, the server SHOULD end the session.

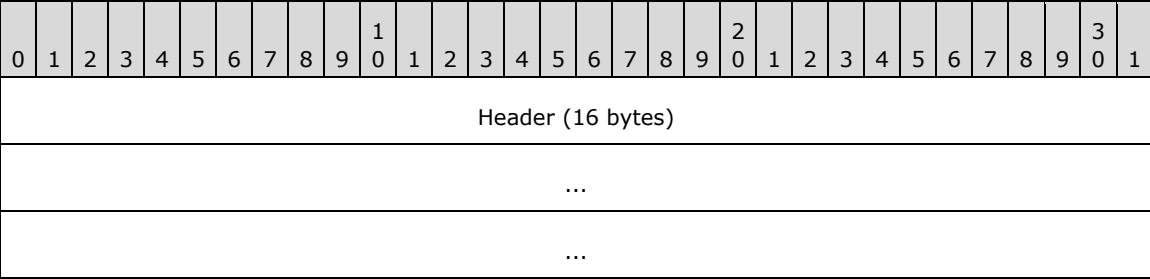


...

**Header (16 bytes):** An MSBMSGBASE packet that defines the packet header. The **wMessageId** field MUST be set to 0x0001.

**2.2.4 MSB\_MSG\_REQ\_STREAMINFO**

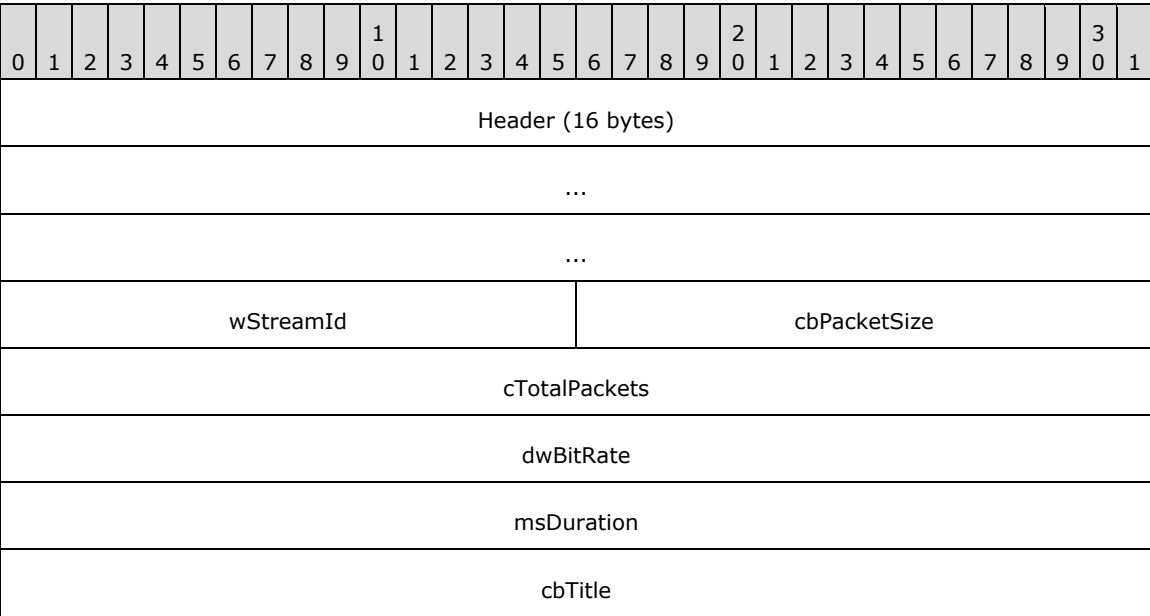
The client MAY send the MSB\_MSG\_REQ\_STREAMINFO packet to request information about the media stream. This request is optional because the server always sends the stream information in an MSB\_MSG\_IND\_STREAMINFO packet immediately after sending an MSB\_MSG\_RES\_CONNECT packet. The MSB\_MSG\_REQ\_STREAMINFO packet is deprecated and SHOULD NOT be sent by clients.



**Header (16 bytes):** An MSBMSGBASE packet that defines the packet header. The **wMessageId** field MUST be set to 0x0003.

**2.2.5 MSB\_MSG\_RES\_STREAMINFO**

The server MUST respond with an MSB\_MSG\_RES\_STREAMINFO packet to a client MSB\_MSG\_REQ\_STREAMINFO packet that requests media stream information. The structure and purpose of this packet are identical to those of the MSB\_MSG\_IND\_STREAMINFO packet. The client requests media stream information from the server only if the server omits sending the MSB\_MSG\_IND\_STREAMINFO packet immediately after the MSB\_MSG\_RES\_CONNECT packet.



cbDescription
cbLink
cbHeader
bBinaryData (variable)
...

**Header (16 bytes):** An MSBMSGBASE packet that defines the packet header. The **wMessageId** field MUST be set to 0x0004.

**wStreamId (2 bytes):** An unsigned 16-bit integer that uniquely identifies the media stream that is currently being transmitted. The value of this field MUST be identical to the value of **wStreamId** that is used in the MSB\_MSG\_IND\_STREAMINFO packet that the server transmitted for the current stream. If the server has not transmitted an MSB\_MSG\_IND\_STREAMINFO packet for the current stream, the value of **wStreamId** MUST be chosen in accordance with the rules for choosing a **wStreamId** value for an MSB\_MSG\_IND\_STREAMINFO packet. The value of the **wStreamId** field MUST be in the range 0x0000 to 0xFFFF, inclusive.

**cbPacketSize (2 bytes):** An unsigned 16-bit integer that specifies the maximum size, in bytes, of the payload in each MSB\_MSG\_IND\_PACKET packet. The value of the **cbPacketSize** field MUST be in the range 0x0000 to 0xFFFF, inclusive.

**cTotalPackets (4 bytes):** An unsigned 32-bit integer that MUST be set to the total number of MSB\_MSG\_IND\_PACKET packets that are needed to transmit the entire media stream—if this information is known. Otherwise, the field MUST be set to zero. The value of the **cTotalPackets** field MUST be in the range 0x00000000 to 0x0000FFFF, inclusive.

**dwBitRate (4 bytes):** An unsigned 32-bit integer that contains the bit rate, in bits per second, of the media stream that is measured. For an ASF stream, the value of the **dwBitRate** field corresponds to the value of the Maximum Bitrate field in the ASF File Properties Object. The value of the **dwBitRate** field MUST be in the range 0x00000000 to 0x0000FFFF, inclusive.

**msDuration (4 bytes):** An unsigned 32-bit integer that MUST be set to the duration, in milliseconds, of the media stream, if this information is known. Otherwise, this value MUST be set to -1 (FFFFFFFF hexadecimal). For an ASF stream, the value of the **msDuration** field corresponds to the value of the Play Duration field in the ASF File Properties Object. The value of the **msDuration** field MUST be in the range 0x00000000 to 0x0000FFFF, inclusive.

**cbTitle (4 bytes):** An unsigned 32-bit integer that contains the size, in bytes, of the media stream title. The stream title MUST be first in the **bBinaryData** field. The value of the **cbTitle** field MUST be in the range 0x00000000 to 0x0000FFCF, inclusive.

**cbDescription (4 bytes):** An unsigned 32-bit integer that contains the size, in bytes, of the media stream description. The stream description MUST follow the stream title in the **bBinaryData** field. The value of the **cbDescription** field MUST be in the range 0x00000000 to 0x0000FFCF, inclusive.

**cbLink (4 bytes):** An unsigned 32-bit integer that contains the size, in bytes, of the media stream link (an arbitrary moniker for the connection over which the media stream is to be transmitted). The stream link MUST follow the stream description in the **bBinaryData** field. The value of the **cbLink** field MUST be in the range 0x00000000 to 0x0000FFCF, inclusive.

**cbHeader (4 bytes):** An unsigned 32-bit integer that contains the size, in bytes, of the media stream header. The stream header MUST follow the stream link in the **bBinaryData** field. If the **szChannel** field of the MSB\_MSG\_REQ\_CONNECT packet is set to NetShow, the stream is in ASF,

as specified in [ASF] section 2, and the stream header is an ASF header. In the case of an ASF Header, the value of the **cbHeader** field is the size of the entire ASF Header Object, as described in [ASF] section 3.1, plus 50 bytes (which represents the fixed initial portion of the ASF Data Object, as described in [ASF] section 5.1). The value of the **cbHeader** field MUST be in the range 0x00000000 to 0x0000FFCF, inclusive.

**bBinaryData (variable):** The binary data that contains the title, description, link, and header of the media stream, in that order. The method for generating the binary data comprising the title, description, link or header (if not an ASF header) of the media stream is implementation-specific and outside the scope of this documentation. The size of the **bBinaryData** field is equal to the sum of the values of the **cbTitle**, **cbDescription**, **cbLink**, and **cbHeader** fields. The size of the **bBinaryData** field MUST be in the range of 0 to 65,487 bytes. ~~It should be noted~~**Note** that these requirements impose a restriction on the values that the **cbTitle**, **cbDescription**, **cbLink**, and **cbHeader** fields can assume. For example, **cbTitle** and **cbDescription** cannot both have a value of 0x8000 at the same time.

## 2.2.6 MSB\_MSG\_IND\_STREAMINFO

The server MUST send the MSB\_MSG\_IND\_STREAMINFO packet, which describes the media stream, in the following circumstances:

- Following an MSB\_MSG\_RES\_CONNECT packet.
- When a stream switch occurs while playing a server playlist.
- Following an MSB\_MSG\_IND\_EOS packet. For more information about sending an MSB\_MSG\_IND\_STREAMINFO packet in this case, see section 3.1.5.3.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Header (16 bytes)																															
...																															
...																															
wStreamId																cbPacketSize															
cTotalPackets																															
dwBitRate																															
msDuration																															
cbTitle																															
cbDescription																															
cbLink																															
cbHeader																															
bBinaryData (variable)																															

...

**Header (16 bytes):** An MSBMSGBASE packet that defines the packet header. The **wMessageId** field MUST be set to 0x0005.

**wStreamId (2 bytes):** An unsigned 16-bit integer that uniquely identifies the media stream that will be transmitted, within the current server playlist. The value of this field can be chosen arbitrarily by the server; however, it MUST be different from the wStreamId values that are used for any other streams in the same server playlist. This allows the client to distinguish MSB\_MSG\_IND\_PACKET packets that belong to the new stream from MSB\_MSG\_IND\_PACKET packets that belong to previous streams (if any). The value of the wStreamId field MUST be in the range 0x0000 to 0x07FF, inclusive, or in the range 0x8000 to 0x87FF, inclusive.

**cbPacketSize (2 bytes):** An unsigned 16-bit integer that specifies the maximum size, in bytes, of the payload in each MSB\_MSG\_IND\_PACKET packet. The value of the **cbPacketSize** field MUST be in the range 0x0000 to 0xFFF7, inclusive.

**cTotalPackets (4 bytes):** An unsigned 32-bit integer that MUST be set to the total number of MSB\_MSG\_IND\_PACKET packets that are needed to transmit the entire media stream—if this information is known. Otherwise, the field MUST be set to 0. The value of the **cTotalPackets** field MUST be in the range 0x00000000 to 0xFFFFFFFF, inclusive.

**dwBitRate (4 bytes):** An unsigned 32-bit integer that represents the bit rate, measured in bits per second, of the media stream. For an ASF stream, the value of the **dwBitRate** field corresponds to the value of the Maximum Bitrate field in the ASF File Properties Object. The value of the **dwBitRate** field MUST be in the range 0x00000000 to 0xFFFFFFFF, inclusive.

**msDuration (4 bytes):** An unsigned 32-bit integer that MUST be set to the duration, in milliseconds, of the media stream—if this information is known. Otherwise, this value MUST be set to -1 (FFFFFFFF hexadecimal). For an ASF stream, the value of the **msDuration** field corresponds to the value of the Play Duration field in the ASF File Properties Object. The value of the **msDuration** field MUST be in the range 0x00000000 to 0xFFFFFFFF, inclusive.

**cbTitle (4 bytes):** An unsigned 32-bit integer that contains the size, in bytes, of the media stream title. The stream title MUST be first in the **bBinaryData** field. The value of the **cbTitle** field MUST be in the range 0x00000000 to 0x0000FFCF, inclusive.

**cbDescription (4 bytes):** An unsigned 32-bit integer that represents the size, in bytes, of the media stream description. The stream description MUST follow the stream title in the **bBinaryData** field. The value of the **cbDescription** field MUST be in the range 0x00000000 to 0x0000FFCF, inclusive.

**cbLink (4 bytes):** An unsigned 32-bit integer that contains the size, in bytes, of the media stream link (an arbitrary moniker for the connection over which the media stream is to be transmitted). The stream link MUST follow the stream description in the **bBinaryData** field. The value of the **cbLink** field MUST be in the range 0x00000000 to 0x0000FFCF, inclusive.

**cbHeader (4 bytes):** An unsigned 32-bit integer that contains the size, in bytes, of the media stream header. The stream header MUST follow the stream link in the **bBinaryData** field. If the **szChannel** field of the MSB\_MSG\_REQ\_CONNECT packet is set to NetShow, the stream is in the ASF, as specified in [ASF] section 2, and the stream header is an ASF header. In the case of an ASF Header, the value of the **cbHeader** field is the size of the entire ASF Header Object, as described in [ASF] section 3.1, plus 50 bytes (which represents the fixed initial portion of the ASF Data Object, as described in [ASF] section 5.1). The value of the **cbHeader** field MUST be in the range 0x00000000 to 0x0000FFCF, inclusive.

**bBinaryData (variable):** The binary data that contains the title, description, link, and header of the media stream, in that order. The method for generating the binary data comprising the title, description, link or header (if not an ASF header) of the media stream is implementation-specific

and outside the scope of this documentation. The size of the **bBinaryData** field is equal to the sum of the values of the **cbTitle**, **cbDescription**, **cbLink**, and **cbHeader** fields. The size of the **bBinaryData** field MUST be in the range of 0 to 65,487 bytes. ~~It should be noted~~**Note** that these requirements impose a restriction on the values that the **cbTitle**, **cbDescription**, **cbLink**, and **cbHeader** fields can assume. For example, **cbTitle** and **cbDescription** cannot both have a value of 0x8000 at the same time.

### 2.2.7 MSB\_MSG\_REQ\_CONNECT

The client sends an MSB\_MSG\_REQ\_CONNECT packet to request a media stream from the server.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Header (16 bytes)																															
...																															
...																															
dwFlags																															
szChannel (variable)																															
...																															

**Header (16 bytes):** An MSBMSGBASE packet that defines the packet header. The **wMessageId** field MUST be set to 0x0007.

**dwFlags (4 bytes):** An unsigned 32-bit integer that represents the connection type. The value MUST be one of the following.

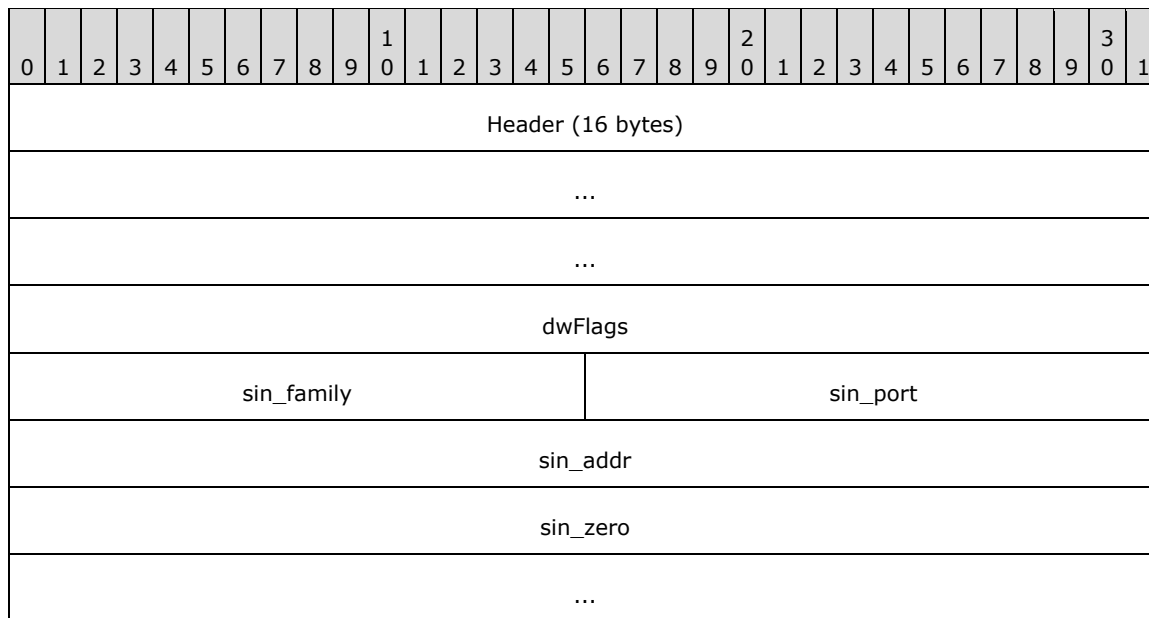
Value	Meaning
0x0002	The media stream is sent to multiple clients simultaneously by using IP multicasting; each client maintains an individual TCP control connection to the server.
0x0001	The media stream is sent to a single client using the current TCP connection.

**szChannel (variable):** This field SHOULD be set to the word NetShow, which MUST be represented as a string of Unicode characters. The terminating null character SHOULD NOT be included. The size of the string can be determined by subtracting the sizes of all other fields from the value of the **cbMessage** field in the packet header MSBMSGBASE. The size of the **szChannel** field MUST be between 0 and 65,514 bytes. Because the **szChannel** field consists of a Unicode character string, the size of the field must be an even number of bytes.

### 2.2.8 MSB\_MSG\_RES\_CONNECT

The server MUST send the MSB\_MSG\_RES\_CONNECT packet in response to the MSB\_MSG\_REQ\_CONNECT packet that is sent by the client.





**Header (16 bytes):** An MSBMSGBASE packet that defines the packet header. The **wMessageId** field MUST be set to 0x0008.

**dwFlags (4 bytes):** An unsigned 32-bit integer that specifies ASF information. This field SHOULD be set to 0x00000002 if the ASF header is present in a Windows Media Station **.nsc file**; otherwise, this field MUST be set to 0x00000000. For information about .nsc files, see [MS-MSB] section 2.2.1.

Value	Meaning
0x00000000	ASF header NOT present in .nsc file
0x00000002	ASF header present in .nsc file

**sin\_family (2 bytes):** An unsigned 16-bit integer. The **sin\_family** field MUST be set to 2 if the media stream is being multicast; otherwise, it SHOULD be set to 0.

**sin\_port (2 bytes):** An unsigned 16-bit integer transmitted in big-endian byte order. If the media stream is being multicast, the **sin\_port** field MUST be set to the UDP port number for the requested channel; otherwise, it SHOULD be set to 0. The value of the **sin\_port** field MUST be in the range 0x0000 to 0xFFFF, inclusive.

**sin\_addr (4 bytes):** An unsigned 32-bit integer that is transmitted in big-endian byte order. If the media stream is being multicast, the **sin\_addr** field MUST be set to a valid IPv4 multicast address for the requested channel; otherwise, it SHOULD be set to 0. The value of the **sin\_addr** field MUST be in the range 0x00000000 to 0xFFFFFFFF, inclusive.

**sin\_zero (8 bytes):** An array of 8 bytes. Each byte in the array SHOULD be set to 0 and MUST be ignored on receipt.

### 2.2.9 MSB\_MSG\_IND\_EOS

The server MUST send the MSB\_MSG\_IND\_EOS packet to indicate the end of the media stream. This packet always precedes an empty MSB\_MSG\_IND\_STREAMINFO packet.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Header (16 bytes)																															
...																															
...																															

**Header (16 bytes):** An MSBMSGBASE packet that defines the packet header. The **wMessageId** field MUST be set to 0x0009.

## 2.2.10 MSB\_MSG\_IND\_PACKET

The server transmits the media stream by using MSB\_MSG\_IND\_PACKET packets. Each ASF data packet in the media stream MUST be encapsulated in a separate MSB\_MSG\_IND\_PACKET packet. The server MUST NOT combine multiple ASF packets in a single MSB\_MSG\_IND\_PACKET and MUST NOT fragment an ASF packet across multiple MSB\_MSG\_IND\_PACKET packets.

The MSB\_MSG\_IND\_PACKET packets MUST be transmitted over the MSBD TCP connection (that is, the same TCP connection on which the server transmitted the MSB\_MSG\_RES\_CONNECT packet, except when the server has agreed to use IP multicasting to deliver the MSB\_MSG\_IND\_PACKET packets). When IP multicasting is used, the server MUST transmit the MSB\_MSG\_IND\_PACKET packets to the IP multicast address/UDP port number combination indicated in the MSB\_MSG\_RES\_CONNECT packet.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Header (16 bytes)																															
...																															
...																															
dwPacketId																															
wStreamId																wPacketSize															
bPayload (variable)																															
...																															

**Header (16 bytes):** An MSBMSGBASE packet that defines the packet header. The **wMessageId** field MUST be set to 0x000A.

**dwPacketId (4 bytes):** An unsigned 32-bit integer that identifies the MSB\_MSG\_IND\_PACKET packet. This value SHOULD start with 0 for the first MSB\_MSG\_IND\_PACKET packet that is transmitted; alternatively, the starting value MAY be chosen randomly. The value of the **dwPacketId** field MUST be incremented for each new MSB\_MSG\_IND\_PACKET packet that is transmitted. The value of the **dwPacketId** field MUST fall within the range 0x00000000 to 0xFFFFFFFF, inclusive.

**wStreamId (2 bytes):** An unsigned 16-bit integer that identifies the content. This value MUST be the same as the **wStreamId** field in the MSB\_MSG\_IND\_STREAMINFO packet. The value of the

**wStreamId** field MUST be in the range 0x0000 to 0x07FF, inclusive, or in the range 0x8000 to 0x87FF, inclusive.

**wPacketSize (2 bytes):** An unsigned 16-bit integer that contains the size, in bytes, of the **bPayload** field plus the size of the **wStreamId**, **dwPacketId**, and **wPacketSize** fields. The **wPacketSize** field MUST be set to a value in the range 0x0008 to 0xFFEF, inclusive.

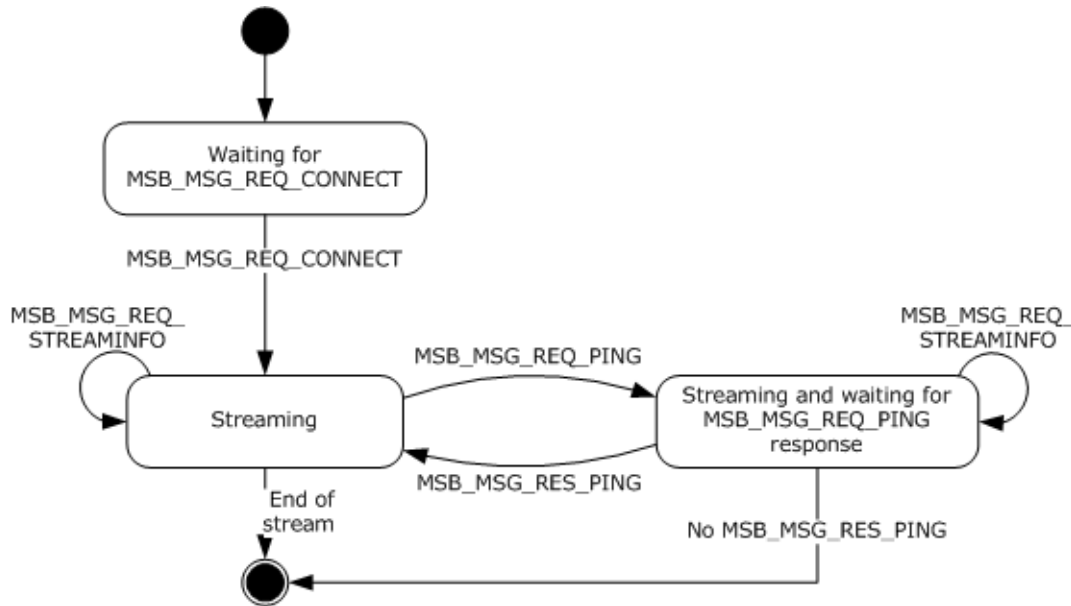
**bPayload (variable):** One ASF packet. If the ASF packet contains a nonzero **Padding Data** field, this field MUST be included in **bPayload**. Details on the **Padding Data** field are as specified in [ASF] section 5.2.4. Error correction, as specified in [MS-MSB] section 2.2.2, MAY be used, in which case some of the ASF packets are **parity packets**, as specified in [MS-MSB] section 2.2.2. If the **bPayload** field contains a parity packet, the **dwPacketId** field MUST be identical to that of the previously transmitted MSB\_MSG\_IND\_PACKET.

### 3 Protocol Details

The following sections specify details of the MSBD Protocol, including abstract data models and message processing rules for both server and client.

#### 3.1 Server Details

This section specifies details for server implementations of the MSBD Protocol. The state machine for MSBD on a server is depicted in the following diagram.



**Figure 1: State machine for MSBD on a server**

##### 3.1.1 Abstract Data Model

Not applicable for this protocol specification.

##### 3.1.2 Timers

An implementation of the MSBD Protocol on a server can use a timer to schedule the transmission of MSB\_MSG\_REQ\_PING packets.

##### 3.1.3 Initialization

If a timer is used to schedule the transmission of MSB\_MSG\_REQ\_PING packets, that timer **should not** be running initially.

The server initializes by listening for incoming TCP connections on the TCP port that is reserved by the server for communication using the MSBD Protocol. The server **MUST** then wait for an MSB\_MSG\_REQ\_CONNECT packet to be received on a new TCP connection.

For more information about how to process the MSB\_MSG\_REQ\_CONNECT packet, see section 3.1.5.1.

### 3.1.4 Higher-Layer Triggered Events

Not applicable for this protocol specification.

### 3.1.5 Message Processing Events and Sequencing Rules

#### 3.1.5.1 Receiving an MSB\_MSG\_REQ\_CONNECT Packet

If the value of the **dwFlags** field in the MSB\_MSG\_REQ\_CONNECT packet is 0x0002 and the server does not support delivery of MSB\_MSG\_IND\_PACKET packets over UDP, then the server MUST reject the connection request. In that case, the server MUST respond with an MSB\_MSG\_RES\_CONNECT packet where the **hr** field is set to an HRESULT failure code. The value of the **hr** field SHOULD then be set either to 0xC00D001A or to 0x80070057, but the server MAY set the **hr** field to a different value as long as it corresponds to an HRESULT failure code. For more information about HRESULT codes, see [MS-ERREF].

If the connection is accepted, the server MUST send an MSB\_MSG\_RES\_CONNECT packet with the **hr** field set to 0x00000000 and MUST follow it with an MSB\_MSG\_IND\_STREAMINFO packet. If the connection is rejected, the server MUST close the TCP connection after having sent the MSB\_MSG\_RES\_CONNECT packet.

If the connection is accepted, then the server transmits the stream as a sequence of MSB\_MSG\_IND\_PACKET packets.

#### 3.1.5.2 Receiving an MSB\_MSG\_REQ\_STREAMINFO Packet

The server MUST send an MSB\_MSG\_RES\_STREAMINFO packet.

#### 3.1.5.3 Reaching the End of the Stream

When the server transmits the last MSB\_MSG\_IND\_PACKET packet for a stream, the server MUST send an MSB\_MSG\_IND\_EOS packet that is followed by an MSB\_MSG\_IND\_STREAMINFO packet. The MSB\_MSG\_IND\_STREAMINFO packet MUST contain information about the next stream to be transmitted. If there are no additional streams to transmit, the MSB\_MSG\_IND\_STREAMINFO packet MUST be "empty". In an empty MSB\_MSG\_IND\_STREAMINFO packet, the size of the **bBinaryData** field MUST be 0 bytes, and the **hr** field SHOULD be set to 0xC00D0033. The server SHOULD set all the remaining fields in the empty MSB\_MSG\_IND\_STREAMINFO packet to 0.<4>

### 3.1.6 Timer Events

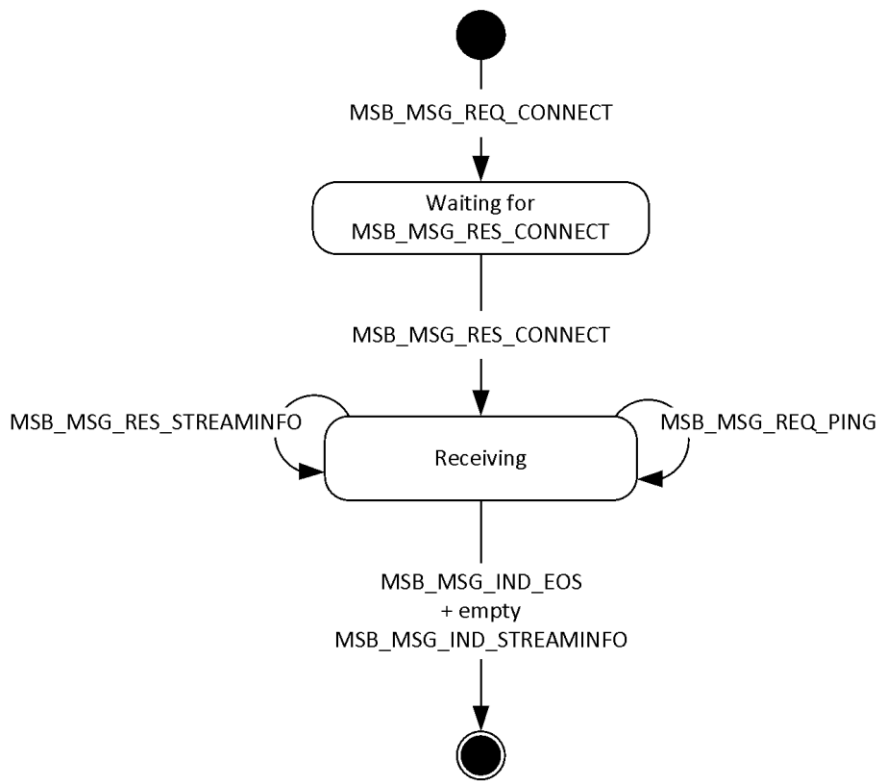
If the server uses a timer to schedule the transmission of the MSB\_MSG\_REQ\_PING packets, when that timer expires, the server SHOULD send an MSB\_MSG\_REQ\_PING packet. If after sending an MSB\_MSG\_REQ\_PING packet, the server does not receive an MSB\_MSG\_RES\_PING packet within 2 minutes, the server SHOULD end the session by closing the TCP connection to the client.

### 3.1.7 Other Local Events

Not applicable for this protocol specification.

## 3.2 Client Details

This section specifies details for client implementations of the MSBD Protocol. The state machine for MSBD on a client is depicted in the following diagram.



**Figure 2: State machine for MSBD on a client**

### 3.2.1 Abstract Data Model

Not applicable for this protocol specification.

### 3.2.2 Timers

Not applicable for this protocol specification.

### 3.2.3 Initialization

The client initializes an MSBD streaming session by establishing a TCP connection to the TCP port that the server reserves for communication by using the MSBD Protocol. The client and server **MUST** agree on the TCP port number that is used for the MSBD Protocol through some mechanism not addressed in the MSBD Protocol, because the port number **MUST** be known prior to the transmission of the first MSBD packet.

After the TCP connection is established, the client **MUST** send an MSB\_MSG\_REQ\_CONNECT packet.

### 3.2.4 Higher-Layer Triggered Events

Not applicable for this protocol specification.

## **3.2.5 Message Processing Events and Sequencing Rules**

### **3.2.5.1 Receiving an MSB\_MSG\_REQ\_PING Packet**

The client MUST send an MSB\_MSG\_RES\_PING packet.

### **3.2.5.2 Receiving an MSB\_MSG\_IND\_STREAMINFO Packet**

If the client receives an "empty" MSB\_MSG\_IND\_STREAMINFO packet (as specified in section 3.1.5.3), the portion of the packet that does not include the **Header** field MUST be ignored.

### **3.2.6 Timer Events**

Not applicable for this protocol specification.

### **3.2.7 Other Local Events**

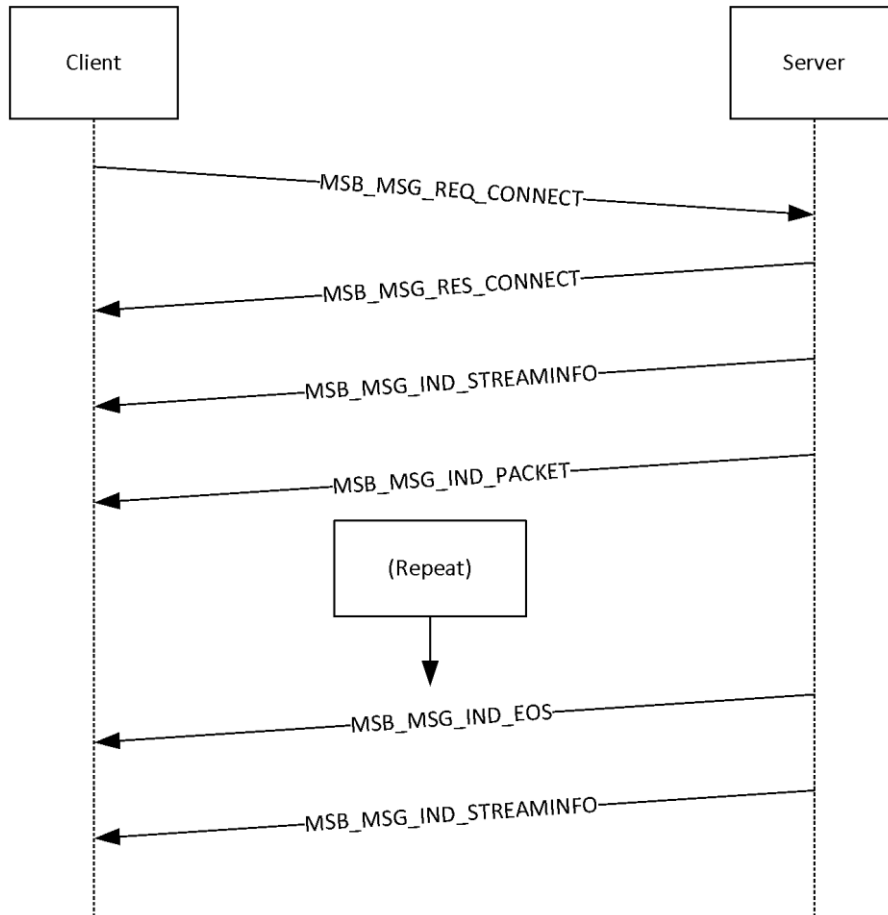
Not applicable for this protocol specification.

## 4 Protocol Example

The following sections describe several operations as used in common scenarios to illustrate the function of the MSBD Protocol.

### 4.1 General MSBD Sequence

The following diagram shows a typical communication sequence between the client and server.



**Figure 3: Client/server communication sequence**

1. The client opens a TCP connection to the server and sends an MSB\_MSG\_REQ\_CONNECT packet.
2. The server responds with an MSB\_MSG\_RES\_CONNECT packet followed by an MSB\_MSG\_IND\_STREAMINFO packet that describes the media stream.
3. The server sends one or more MSB\_MSG\_IND\_PACKET packets, each of which contains one ASF packet. The packets are sent either over the TCP connection or by using UDP to an IP multicast group.
4. When the stream ends, the server sends an MSB\_MSG\_IND\_EOS packet that is followed by an empty MSB\_MSG\_IND\_STREAMINFO packet.

During the previous message exchange, the server periodically pings the client by using an MSB\_MSG\_REQ\_PING packet. If the client does not respond to the MSB\_MSG\_REQ\_PING packet with

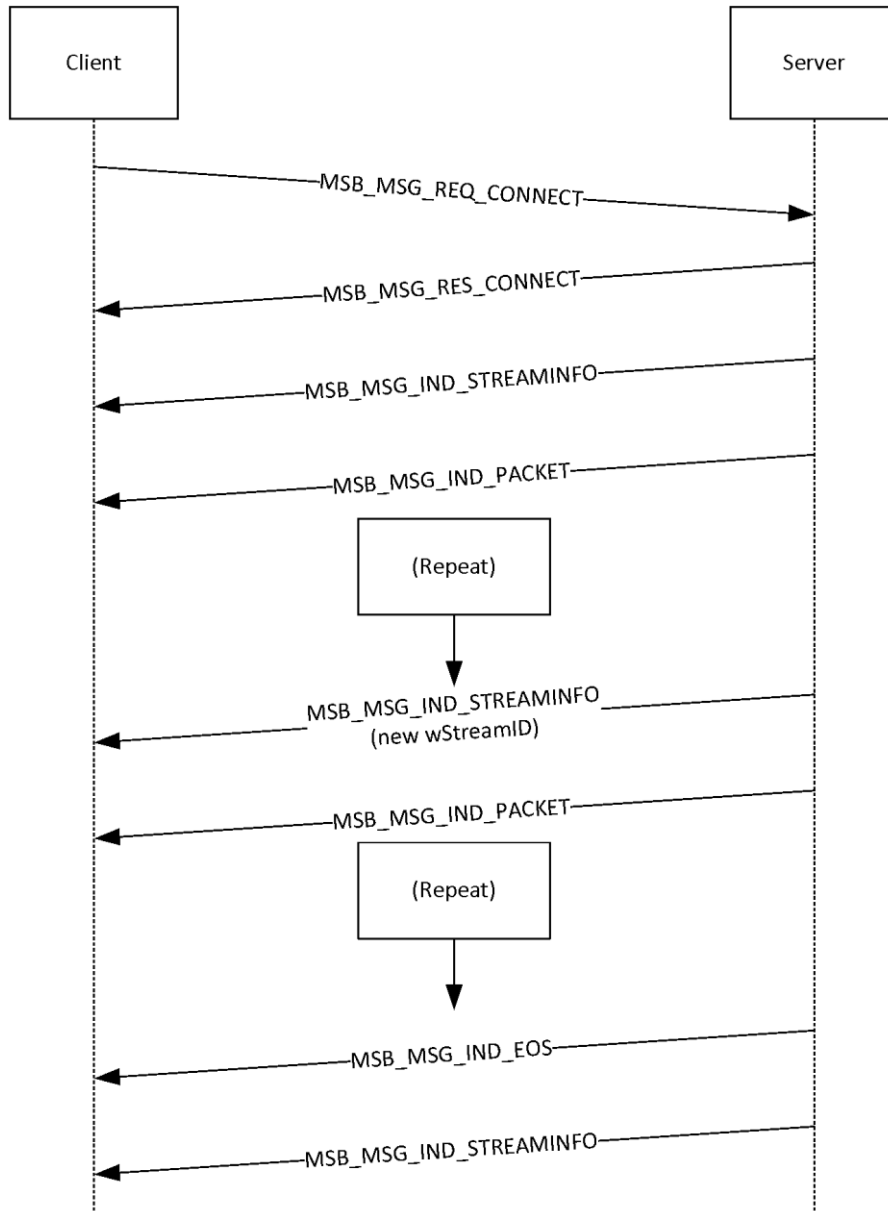


an MSB\_MSG\_RES\_PING packet, the server closes the TCP connection. In other cases, the client is responsible for ending the session by closing the TCP connection.

## 4.2 Using MSBD for Server-Side Playlist Streaming

The following diagram shows a typical communication sequence between a client and server when streaming content from a server-side playlist.

**Note** When using the MSBD Protocol, the client does not have the capability to seek or skip to a new entry in a server-side playlist.



**Figure 4: Client/server communication sequence when streaming from a server-side playlist**

1. The client opens a TCP connection to the server and sends an MSB\_MSG\_REQ\_CONNECT packet.

2. The server responds with an MSB\_MSG\_RES\_CONNECT packet followed by an MSB\_MSG\_IND\_STREAMINFO packet that describes the media stream.
3. The server sends one or more MSB\_MSG\_IND\_PACKET packets, each of which contains one ASF packet. The packets are sent either over the TCP connection or by using UDP to an IP multicast group.
4. When the entry changes in the server-side playlist, the server sends a MSB\_MSG\_IND\_STREAMINFO packet with a new **wStreamId** value. For more information, see section 2.2.6.
5. The server sends one or more MSB\_MSG\_IND\_PACKET packets for the new playlist entry, each of which contains one ASF packet. The packets are sent either over the TCP connection or by using UDP to an IP multicast group.
6. The preceding two steps are repeated for all entries in the server-side playlist.
7. When the streams end, the server sends an MSB\_MSG\_IND\_EOS packet that is followed by an empty MSB\_MSG\_IND\_STREAMINFO packet.

During the preceding message exchange, the server periodically pings the client by using an MSB\_MSG\_REQ\_PING packet. If the client does not respond to the MSB\_MSG\_REQ\_PING packet with an MSB\_MSG\_RES\_PING packet, the server closes the TCP connection. In other cases, the client is responsible for ending the session by closing the TCP connection.

## 5 Security

The following sections specify security considerations for implementers of the MSBD Protocol.

### 5.1 Security Considerations for Implementers

For any packet that contains fields that specify other field lengths, implementers ~~SHOULD~~ ensure that the length values do not exceed the size of the packet itself. For example, MSB\_MSG\_IND\_STREAMINFO is one packet that has multiple length fields.

Implementers ~~SHOULD~~do not assume that the ASF data in the MSB\_MSG\_IND\_STREAMINFO and MSB\_MSG\_IND\_PACKET packets can be trusted. Use caution when parsing variable-length fields, ensuring that length information, if any, does not cause the size of buffers to be exceeded.

### 5.2 Index of Security Parameters

The MSBD Protocol has no applicable security parameters.

## 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.

- Windows NT operating system
- Windows 2000 operating system
- Windows XP operating system
- Windows Server 2003 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 1.3: The MSBD Protocol is only supported by Windows Media Player 6.4.

<2> Section 1.4: The IP multicast is supported only by Windows Media Services implementations in Windows NT 4.0 operating system and Windows 2000 Server operating system.

<3> Section 1.6: The server role is only supported in Windows NT 4.0 and Windows 2000 Server.

<4> Section 3.1.5.3: Windows Media Services on Windows NT 4.0, Windows 2000 Server, and Windows Server 2003 does not assign values to the **wStreamId**, **cbPacketSize**, **cTotalPackets**, **dwBitRate**, **msDuration**, **cbTitle**, **cbDescription**, **cbLink**, or **cbHeader** fields.

## 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

### A

Abstract data model  
  client 22  
  server 20  
Applicability 8  
Applicability statement 8

### C

Capability negotiation 8  
Change tracking 29  
Client  
  abstract data model 22  
  higher-layer triggered events 22  
  initialization 22  
  local events 23  
  message processing  
    MSB\_MSG\_IND\_STREAMINFO 23  
    MSB\_MSG\_REQ\_PING 23  
  other local events 23  
  overview 21  
  sequencing rules  
    MSB\_MSG\_IND\_STREAMINFO 23  
    MSB\_MSG\_REQ\_PING 23  
  timer events 23  
  timers 22

### D

Data model - abstract  
  client 22  
  server 20  
Data model – abstract  
  client 22  
  server 20

### E

Examples (section 4 24, section 4.1 24, section 4.2 25)

### F

Fields - vendor-extensible 9  
Fields – vendor-extensible 9

### G

Glossary 7

### H

Higher-layer triggered events  
  client 22  
  server 21

### I

Implementer - security considerations 27  
Implementers – security considerations 27  
Index of security parameters 27  
Informative references 8

Initialization  
  client 22  
  server 20  
Introduction 7

## **L**

Local events  
  client 23  
  server 21

## **M**

Message processing  
  client  
    MSB\_MSG\_IND\_STREAMINFO 23  
    MSB\_MSG\_REQ\_PING 23  
  server  
    end of stream 21  
    MSB\_MSG\_REQ\_CONNECT 21  
    MSB\_MSG\_REQ\_STREAMINFO 21

### Messages

MSB\_MSG\_IND\_EOS 17  
MSB\_MSG\_IND\_PACKET 18  
MSB\_MSG\_IND\_STREAMINFO 14  
MSB\_MSG\_REQ\_CONNECT 16  
MSB\_MSG\_REQ\_PING 11  
MSB\_MSG\_REQ\_STREAMINFO 12  
MSB\_MSG\_RES\_CONNECT 16  
MSB\_MSG\_RES\_PING 11  
MSB\_MSG\_RES\_STREAMINFO 12  
MSBMSGBASE 10  
syntax 10  
transport 10  
MSB\_MSG\_IND\_EOS message 17  
MSB\_MSG\_IND\_EOS packet 17  
MSB\_MSG\_IND\_PACKET message 18  
MSB\_MSG\_IND\_PACKET packet 18  
MSB\_MSG\_IND\_STREAMINFO message 14  
MSB\_MSG\_IND\_STREAMINFO packet 14  
MSB\_MSG\_REQ\_CONNECT message 16  
MSB\_MSG\_REQ\_CONNECT packet 16  
MSB\_MSG\_REQ\_PING message 11  
MSB\_MSG\_REQ\_PING packet 11  
MSB\_MSG\_REQ\_STREAMINFO message 12  
MSB\_MSG\_REQ\_STREAMINFO packet 12  
MSB\_MSG\_RES\_CONNECT message 16  
MSB\_MSG\_RES\_CONNECT packet 16  
MSB\_MSG\_RES\_PING message 11  
MSB\_MSG\_RES\_PING packet 11  
MSB\_MSG\_RES\_STREAMINFO message 12  
MSB\_MSG\_RES\_STREAMINFO packet 12  
MSBMSGBASE message 10  
MSBMSGBASE packet 10

## **N**

Normative references 7

## **O**

Other local events  
  client 23  
  server 21  
Overview (synopsis) 8

## **P**

- Parameters – security 27
- Parameters - security index 27
- Preconditions 8
- Prerequisites 8
- Product behavior 28
- Protocol Details
  - overview 20

## **R**

- References 7
  - informative 8
  - normative 7
- Relationship to other protocols 8

## **S**

- Security 27
  - implementer considerations 27
  - parameter index 27
- Sequencing rules
  - client
    - MSB\_MSG\_IND\_STREAMINFO 23
    - MSB\_MSG\_REQ\_PING 23
  - server
    - end of stream 21
    - MSB\_MSG\_REQ\_CONNECT 21
    - MSB\_MSG\_REQ\_STREAMINFO 21
- Server
  - abstract data model 20
  - higher-layer triggered events 21
  - initialization 20
  - local events 21
  - message processing
    - end of stream 21
    - MSB\_MSG\_REQ\_CONNECT 21
    - MSB\_MSG\_REQ\_STREAMINFO 21
  - other local events 21
  - overview 20
  - sequencing rules
    - end of stream 21
    - MSB\_MSG\_REQ\_CONNECT 21
    - MSB\_MSG\_REQ\_STREAMINFO 21
  - timer events 21
  - timers 20
- Standards assignments 9
- Syntax – message 10

## **T**

- Timer events
  - client 23
  - server 21
- Timers
  - client 22
  - server 20
- Tracking changes 29
- Transport 10
- Transport – message 10
- Triggered events – higher layer
  - client 22
  - server 21



Triggered events - higher-layer  
client 22  
server 21

## **V**

Vendor-extensible fields 9  
Versioning 8