[MS-MSSO]:
Media Streaming Server System Overview

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.

This document provides an overview of the Media Streaming Server System Overview Protocol Family. It is intended for use in conjunction with the Microsoft Protocol Technical Documents,

publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts. It assumes that the reader is either familiar with the aforementioned material or has immediate access to it.

A Protocol Family System Document does not require the use of Microsoft programming tools or programming environments in order to implement the Protocols in the System. Developers who have access to Microsoft programming tools and environments are free to take advantage of them.

Abstract

A streaming media system is a system designed to distribute digital media content from an encoder or a capture application to a media server, and finally, to a media player for rendering or playback of that content. The Media Streaming Server System protocols are a series of protocols designed to achieve that task.

This document serves as a companion document to the protocol and data structure specifications included in the Media Streaming Server System protocol documentation set. It provides an overview of the Windows Media protocols that are implemented in the Windows Server operating systems and that are used to interoperate or communicate with Windows client operating systems.

This document describes the intended functionality of the Media Streaming Server System and how it interacts with systems or applications that need to stream media. As mentioned, the Media Streaming Server System supports multiple protocols for streaming digital media to and from a media server. This document identifies those supported protocols and how they interact in a combined system.

Revision Summary

Date	Revision History	Revision Class	Comments
02/27/2009	0.1	Major	First Release.
04/10/2009	1.0	Major	Updated and revised the technical content.
05/22/2009	1.0.1	Editorial	Revised and edited the technical content.
07/02/2009	1.0.2	Editorial	Revised and edited the technical content.
08/14/2009	2.0	Major	Updated and revised the technical content.
09/25/2009	3.0	Major	Updated and revised the technical content.
11/06/2009	4.0	Major	Updated and revised the technical content.
12/18/2009	5.0	Major	Updated and revised the technical content.
01/29/2010	6.0	Major	Updated and revised the technical content.
03/12/2010	6.0.1	Editorial	Revised and edited the technical content.
04/23/2010	6.0.2	Editorial	Revised and edited the technical content.
06/04/2010	6.0.3	Editorial	Revised and edited the technical content.

Date	Revision History	Revision Class	Comments
07/16/2010	6.0.3	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	6.1	Minor	Clarified the meaning of the technical content.
10/08/2010	6.1	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	6.1	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	6.1	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	6.1	No change	No changes to the meaning, language, or formatting of the technical content.
03/25/2011	6.1	No change	No changes to the meaning, language, or formatting of the technical content.
05/06/2011	6.1	No change	No changes to the meaning, language, or formatting of the technical content.
06/17/2011	6.2	Minor	Clarified the meaning of the technical content.

Contents

1 Introduction 7
1.1 Glossary 7
1.2 References 8
1.2.1 Normative References 8
1.2.2 Informative References 9

2 Overview 10
2.1 System Summary 10
2.2 List of Member Protocols 10
2.3 Relevant Standards 11

3 Foundation 12
3.1 Background Knowledge and System-Specific Concepts 12
3.1.1 Digital Rights Management 12
3.1.2 Encoders 12
3.1.3 Live Broadcast 13
3.1.4 Media Player Clients 13
3.1.5 On Demand Broadcast 13
3.1.6 Playlists 13
3.1.7 Origin Servers 14
3.1.8 Distribution Server 14
3.1.9 Proxy Servers 15
3.1.10 Packet-Pair Bandwidth Estimation 16
3.1.11 Fast Start 16
3.1.12 Advanced Fast Start 17
3.1.13 Unicast Streaming 18
3.1.14 Multicast Streaming 19
3.1.15 UDP Packets 19
3.1.16 TCP Packets 20
3.2 System Purposes 20
3.3 System Use Cases 20
3.3.1 Stakeholders and Interests Summary 20
3.3.2 Supporting Actors and System Interests Summary 21
3.3.3 Use Case Diagrams 21
3.3.4 Use Case Descriptions 23
3.3.4.1 Publish Content to Media Server—Encoder 23
3.3.4.2 Publish Secure Content to Media Server—DRM Packager 25
3.3.4.3 Stream Content from Media Server - Media Player Client 25
3.3.4.4 Request License from License Server - Media Player Client 26
3.3.4.5 Log Statistics to Servers - Media Player Client 27
3.3.4.6 Discover Media Server URLs – Media Player Client 28

4 System Context 30
4.1 System Environment 30
4.1.1 Authentication 31
4.1.2 Media Player Client 31
4.1.3 Encoder 32
4.2 System Assumptions and Preconditions 32
4.3 System Relationships 33
4.3.1 Black Box Relationship Diagram 33

4.3.2 System Dependencies 35
4.3.2.1 Dependencies on MSS 35
4.3.3 System Influences 36
4.4 System Applicability 36
4.5 System Versioning and Capability Negotiation 36
4.6 System Vendor-Extensible Fields 37

5 System Architecture 38
5.1 Abstract Data Model 38
5.2 White Box Relationships 38
5.2.1 Encoder Push Diagram 38
5.2.2 Encoder Pull Diagram 39
5.2.3 Unicast Playback Diagram 40
5.2.4 Multicast Playback Diagram 41
5.3 Member Protocol Functional Relationships 41
5.3.1 Member Protocol Roles 41
5.3.1.1 RTSP-WME Overview 43
5.3.1.1.1 RTSP-WME: Logical Dependencies and Relationship to Other Protocols 44
5.3.1.2 Microsoft Media Server Protocol (MMSP) 44
5.3.1.2.1 MMSP Logical Dependencies and Relationship to Other Protocols 45
5.3.1.3 Media Stream Broadcast Protocol (MSB) 45
5.3.1.3.1 MSB Relationship to Other Protocols 46
5.3.1.4 Media Stream Broadcast Distribution Protocol (MSBD) 46
5.3.1.4.1 MSBD Logical Dependencies and Relationship to Other Protocols 46
5.3.1.5 Windows Media HTTP Streaming Protocol (WMSP) 46
5.3.1.5.1 WMSP Logical Dependencies and Relationship to Other Protocols 47
5.3.1.6 Windows Media HTTP Push Distribution Protocol (WMHTTP) 47
5.3.1.6.1 WMHTTP Logical Dependencies and Relationship to Other Protocols 47
5.3.2 Member Protocol Groups 48
5.4 System Internal Architecture 48
5.5 Failure Scenarios 50

6 System Details 51
6.1 Architectural Details 51
6.1.1 Encoder Push to Media Server 51
6.1.2 Media Server Pull from Encoder 51
6.1.3 Server to Client Streaming 53
6.1.4 Multicast Playback 53
6.1.5 Unicast Playback 53
6.1.6 Packet-Pair Bandwidth Estimation 57
6.1.7 Advanced Fast Start/Fast Start 59
6.1.8 Logging 61
6.1.9 Integrating DRM 63
6.2 Communication Details 65
6.3 Transport Requirements 65
6.4 Timers 65
6.5 Non-Timer Events 65
6.6 Initialization and Reinitialization Procedures 65
6.7 Status and Error Returns 65

7 Security 66

8 Appendix A: Product Behavior 67

9 Change Tracking.....68

10 Index 70

1 Introduction

This Protocol Family System Document (PFSD) is primarily intended to cover the Protocol Family as a whole. In conjunction with Member Protocol Technical Documents (TDs), which are intended to cover Member Protocols, it presents the rules for information exchange relevant to those Member Protocols and the Protocol Family that are used to interoperate or communicate with a Windows operating system in its various environments.

The Media Streaming Server System is a platform for **streaming** audio and video content to clients over the Internet or an intranet. These clients can be other computers or devices that play back the content using a media player, or they can be other computers running media servers that proxy, cache, or redistribute content.

1.1 Glossary

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

Advanced Systems Format (ASF)
client
encryption
key
multicast
server
unicast

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

content
playlist
session
stream
streaming

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

.nsc file

The following terms are specific to this document:

Advanced Fast Start: A process whereby a receiver uses information provided by the sender to decide when playback of **streaming content** should be initiated.

.asx file: A file which contains the URL or set of URLs to the streaming media files.

Fast Start: Streaming content in a quick fashion as requested by the receiver.

MMSP: Microsoft Media Server (MMS) Protocol.

MSB: Media Stream Broadcast (MSB) Protocol.

MSBD: Media Stream Broadcast Distribution (MSBD) Protocol.

publishing point: An organized memory location that is identified by a name on a Media Streaming Server. The name is part of the URL used by a **client** when requesting content from the **server**.

redirector file: A file that is designed to provide the player with information on how to access the streaming media file. Redirector files can be **ASX Files** or **NSC files**.

RTSP-WME: Real-Time Streaming Protocol (RTSP) Windows Media Extensions.

split stream: A **stream** that is being split by a distribution **server** in order to forward to multiple recipients.

WMHTTP: Windows Media HTTP Push Distribution Protocol.

WMLOG: Windows Media Log Data Structure.

WMSP: Windows Media HTTP Streaming Protocol.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. Note that in [RFC2119] terms, most of these specifications should be imperative, to ensure interoperability. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

Any specification that does not explicitly use one of these terms is mandatory, exactly as if it used MUST.

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.

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

If you have any trouble finding [ASF], please check here.

[MS-DRM] Microsoft Corporation, "Digital Rights Management License Protocol Specification".

[MS-MMSP] Microsoft Corporation, "Microsoft Media Server (MMS) Protocol Specification".

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

[MS-MSBD] Microsoft Corporation, "Media Stream Broadcast Distribution (MSBD) Protocol Specification".

[MS-NLMP] Microsoft Corporation, "NT LAN Manager (NTLM) Authentication Protocol Specification".

[MS-RTSP] Microsoft Corporation, "Real-Time Streaming Protocol (RTSP) Windows Media Extensions".

[MS-WMHTTP] Microsoft Corporation, "Windows Media HTTP Push Distribution Protocol Specification".

[MS-WMLOG] Microsoft Corporation, "Windows Media Log Data Structure".

[MS-WMSP] Microsoft Corporation, "Windows Media HTTP Streaming Protocol Specification".

[RFC1945] Berners-Lee, T., Fielding, R., and Frystyk, H., "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996, http://www.ietf.org/rfc/rfc1945.txt

[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

[RFC2326] Schulzrinne, H., Rao, A., and Lanphier, R., "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998, http://www.ietf.org/rfc/rfc2326.txt

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., et al., "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999, http://www.ietf.org/rfc/rfc2617.txt

[RFC3452] Luby, M., Vicisano, L., Gemmel, J., et al., "Forward Error Correction (FEC) Building Block", RFC 3452, December 2002, http://www.ietf.org/rfc/rfc3452.txt

1.2.2 Informative References

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

[MSDN-ASX] Microsoft Corporation, "ASX Elements Reference" http://msdn.microsoft.com/en-us/library/ms910265.aspx

[MSDN-LPS-WME] Microsoft Corporation, "Developing a License Provider Service for Windows Media Encoder", November 2002, http://msdn.microsoft.com/en-us/library/ms867146.aspx

[MSDN-MediaPlaylists] Microsoft Corporation, "Windows Media Playlist Elements Reference", http://msdn.microsoft.com/en-us/library/dd564688(VS.85).aspx

[MSFT-WMSDG] Microsoft Corporation, "Windows Media Services Deployment Guide", http://technet.microsoft.com/en-us/library/cc730848.aspx

[W3C] W3C, "World Wide Web Consortium (W3C)" http://www.w3.org

[WMESDK] Microsoft Corporation, "Windows Media Encoder 9 Series SDK", http://www.microsoft.com/downloads/details.aspx?familyid=000a16f5-d62b-4303-bb22-f0c0861be25b&displaylang=en

[WM9CSEB] Microsoft Corporation, "Creating a Successful Executive Broadcast using Windows Media 9 Series", http://download.microsoft.com/download/7/d/a/7dae51fa-b199-47a1-b583-c4c510535730/broadcastupdated.doc

2 Overview

Section 1, "Introduction," describes this Protocol Family System Document. This section introduces the system that is being documented.

2.1 System Summary

The Media Streaming Server (MSS) System is designed to deliver an end-to-end experience for components involved in the creation, distribution, and playback of audio and video **content**. The system provides administrators and content providers with the ability to create media solutions for corporate communications, training and education, e-commerce, commercial broadcast, and other uses. The Media Streaming Server System consists of a computer running a media encoder, a **server** running as a media server, and a number of **client** computers running media play clients. The encoder converts both live and prerecorded audio and video content to a media format. The server then distributes the content over a network or the Internet. The media player client then receives the content. In order to scale and meet network demands, the system can also included cache and proxy servers, and distribution servers.

For E-commerce scenarios, the Media Streaming Server System may require the support of Digital Rights Management components to provide the administrator with the ability to securely **encrypt** content broadcast and download.

Each of the components in the system uses the member protocols to enable scenarios ranging from live broadcast playback to on-demand playback.

2.2 List of Member Protocols

Media Stream Broadcast (**MSB**) Protocol, a Windows Media Protocol, as specified in [MS-MSB]. This protocol allows the multicast distribution of **Advanced Systems Format (ASF)** packets over a network for which Internet Protocol (IP) multicasting is enabled. MSB allows clients to tune in to a broadcast on a network, much like television and radio users can tune to a particular television or radio station.

Media Stream Broadcast Distribution (**MSBD**) Protocol, a Windows Media Protocol, as specified in [MS-MSBD]. This protocol is used to transfer a live stream of audiovisual content from a server to a single client or multiple clients. The MSBD Protocol can be used to transmit the digitized audio and video of a live event to another computer that is running appropriate streaming media server software such as Windows Media Services, or to distribute the stream to multiple clients.

Microsoft Media Server (**MMSP**) Protocol, a Windows Media Protocol, as specified in [MS-MMSP]. This protocol is used by the Media Streaming Server System to stream data from the Windows Media Server (WMS) to the Windows Media Player (WMP) using Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

Windows Media HTTP Streaming Protocol (**WMSP**), a Windows Media Protocol, as specified in [MS-WMSP]. This protocol is used to transfer real-time multimedia data (for example, audio and video). The protocol depends on Hypertext Transfer Protocol (HTTP) for the transfer of all protocol messages, including the transfer of multimedia data.

Real-Time Streaming Protocol Windows Media Extensions (**RTSP-WME**), a Windows Media Protocol, as specified in [MS-RTSP]. This protocol defines extensions to Real-Time Streaming Protocol (RTSP), Real-Time Transport Protocol (RTP), Session Description Protocol (SDP), and Real-Time Transport Control Protocol (RTCP) to enable the delivery of multimedia data that is encapsulated in ASF packets.

Windows Media HTTP Push Distribution Protocol (**WMHTTP**), a Windows Media Protocol, as specified in [MS-WMHTTP]. This protocol is used to transfer real-time multimedia data (for example, audio and video) from an encoder client to a server. Push distribution is ideal for broadcasting company meetings or live presentations.

In addition to the member protocols listed above, the following data structure is an integral part of the system:

Windows Media Log Data Structure (**WMLOG**), a structure, as specified in [MS-WMLOG]. The Windows Media Log Data Structure is a syntax for logging messages. The logging messages specify information about how a client received multimedia content from a streaming server.

2.3 Relevant Standards

The system does not require any standards beyond those described in the specifications of the protocols supported by the system, as listed in section 2.2.

3 Foundation

This section describes the theoretical and practical information needed to understand this document and this system.

3.1 Background Knowledge and System-Specific Concepts

This section summarizes:

- Background knowledge required to understand this document.
- Concepts that are specific to this system including:
  - **Advanced Fast Start**
  - Digital Rights Management
  - Distribution server
  - Encoders
  - **Fast Start**
  - Live Broadcast
  - Media player clients
  - Multicast streaming
  - On demand broadcast
  - Origin servers
  - Proxy servers
  - Packet pair bandwidth estimation
  - TCP packets
  - UDP packets
  - **Unicast** streaming

3.1.1 Digital Rights Management

Digital Rights Management (DRM) provides content providers with the means to protect their proprietary music or other data from unauthorized copying and other illegal uses. DRM technology protects digital content by encrypting it and attaching to it usage rules that determine the conditions under which a user can play back the content. Usage rules typically limit the number of computers or devices that have access to the content, or limit the number of times that content can be played.

3.1.2 Encoders

An encoder is a tool used to capture audio and video files and streams, digitize them, and provide them to media servers for distribution. For more information on creating a broadcast, please see [WM9CSEB], section 5.

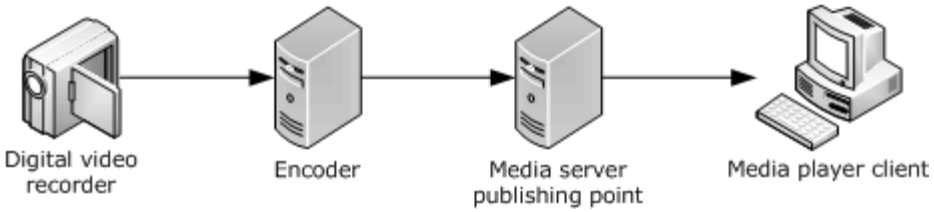

Figure 1: Live broadcast configuration

Encoders typically capture the video and audio **streams** from capture cards and devices. In order to capture from an analog Video Tape Player or Digital Video Recorder that records in analog, the PC will need a capture card that imports the analog stream. The encoder will then convert the analog stream to digital.

3.1.3 Live Broadcast

A live broadcast is often used when viewers want to see and hear an important event as it is occurring. For information on on-demand versus live broadcasts, please see [MSFT-WMSDG], Distributing content.

3.1.4 Media Player Clients

A media player client is usually the destination point in the Media Streaming Server System, and is typically designed for rendering the media streams.

3.1.5 On Demand Broadcast

On-demand broadcast is a re-broadcast of a live event or file that is not time critical. In this case, users can request the stream when they want to watch it, and can control the playback to meet their needs. For information on on-demand versus live broadcasts, please see [MSFT-WMSDG], Distributing content.

3.1.6 Playlists

A **playlist** is a file or collection of content files designed to play in specific order, or a query that results in a list of content files designed to play in a specific order.

The following table describes various playlists used by the MSS:

Playlist formats	Details
Media playlist files	Designed for audio only files and often referred to as the MP3 playlist. It can provide URLs to HTTP servers or media servers.
Advanced Stream Redirector files	Advanced Stream Redirector files are based on the Extensible Markup Language (XML) syntax and are designed specifically to provide URLs to content from media servers to the client. For more information on Advanced Stream Redirector files, see [MSDN-ASX].
Windows Media Player playlist	A SMIL-based playlist query that only works on local content and is not used by the MSS. Windows Media Player Playlist syntax is based on the Synchronized Multimedia Integration Language (SMIL 2.0). Clients load the playlist locally and process them. The playlist provides URLs or paths to the files. The client then streams the individual

Playlist formats	Details
	content using the MSS protocols. For more information about the SMIL 2.0 Specification, see the [W3C] Web site. For more information on the Windows Media Player Playlist syntax, see [MSDN-MediaPlaylists].
Server-side playlists	Server-side playlists are a query method used to generate a list of content to be streamed to the client. The server processes the playlist locally and then streams the content to the client using the MSS protocols. The client is sent a new Advanced Systems Format (ASF) file header each time the server transitions from one entry to the next in the server-side playlist. Server-side playlists are beneficial as they allow the server or encoder operator to inject new entries, such as ads, into a live program.

3.1.7 Origin Servers

An origin server is a media server that publishes on-demand or live content.

3.1.8 Distribution Server

Distribution servers improve the scalability of the Media Streaming Server System. A distribution server publishes content received from another media server. The distribution server has to be networked to the origin server and have permission to stream from the origin server.

A distribution server publishes content received from another streaming source, such as another media server. The origin server is the source of the content being streamed by the distribution server. Clients then connect to the distribution server as if it were the origin server. Distribution servers are located between the origin server and the client in the content stream and are therefore able to perform load balancing. Distribution servers are an easy way to reduce the client load on your media server because you distribute the client content requests to several servers on the network.

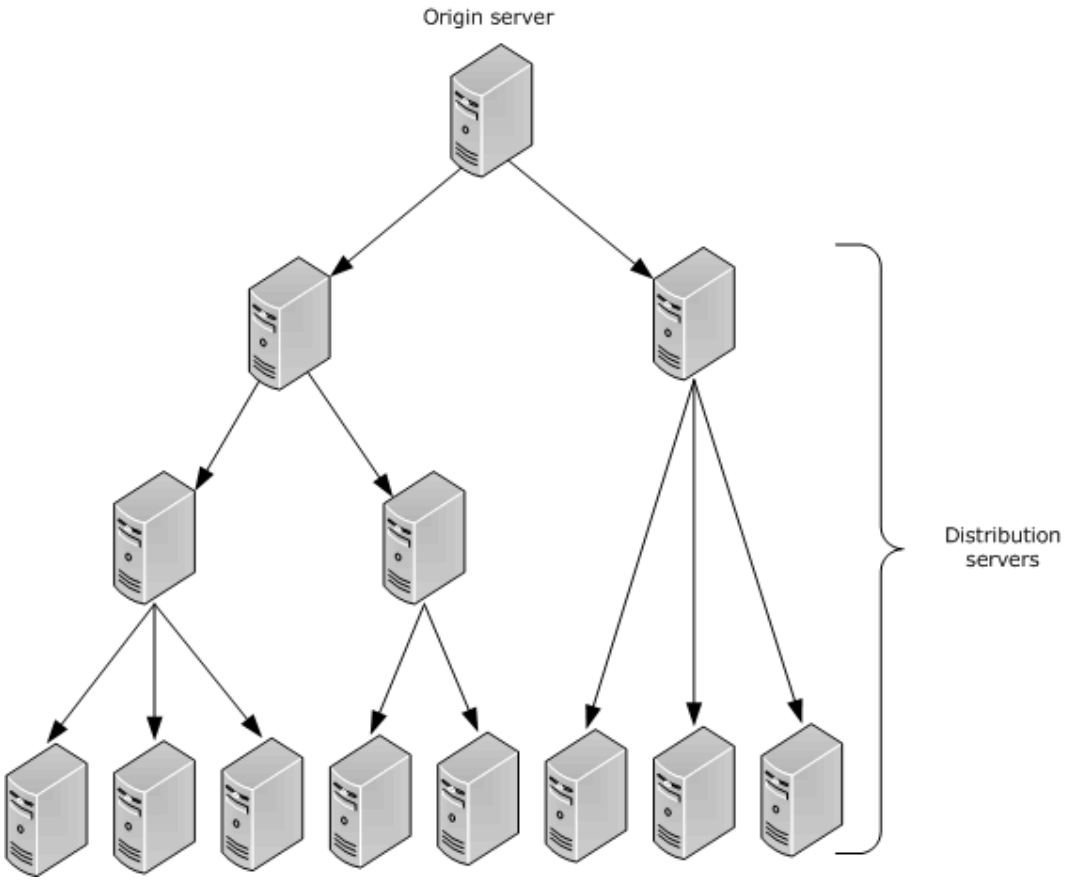

Figure 2: Publishing via Distribution Servers

3.1.9 Proxy Servers

A proxy server is a dedicated computer that proxies data between the media player client and the server. If the server is acting as a caching server the proxy server will request a stream from the origin server and allow multiple clients to stream the content. Therefore the origin server is limited to one network request. If the content is broadcast content the content cannot be cached. In this case the proxy server may **split stream** the content. The proxy server, receiving the stream from the origin server, splits the stream to distribute to multiple clients simultaneously without increasing the requests to the origin server. Proxy servers fall into 3 categories:

Forward Proxy Server

The forward proxy server can retrieve information from another server on behalf of a client. Typically, a client is explicitly configured to use a specific proxy server, and when the client requests content, the proxy server connects to an origin server to retrieve the content.

Reverse Proxy Server

A reverse proxy server is a proxy server configured to be responsible for servicing all client requests. For unicast broadcasts, a reverse proxy server can reduce the load on the origin server by streaming multiple unicast streams while receiving only one stream from the origin server. For on-

demand content, a reverse proxy server can reduce the load on the origin server by caching the content from the origin server and streaming it to clients from its cache.

To the client, the reverse proxy server appears to be the origin server. This enables you to isolate your origin server from your clients. A reverse proxy server can increase the security of your streaming media system because the client never connects to the origin server directly.

Transparent Proxy Server

A transparent proxy server is a server that transmits data between the server and the client without any modification of the data. It is a forwarding service that the client is unaware of.

3.1.10 Packet-Pair Bandwidth Estimation

Packet-pair bandwidth estimation is a technique used to estimate the bandwidth of a streaming media connection over the Internet.

To estimate bandwidth, the server sends two or more consecutive packets of highly entropic data, and the client estimates the bandwidth by measuring the difference between the times that it receives the packets. This method is usually reliable; however, if the client traverses a Network Address Translation (NAT), firewall, or proxy server, the packet-pair bandwidth measurement might be inaccurate. Packet-pair bandwidth estimation is supported by the following protocols: RTSP-WME, MMSP, and WMSP.

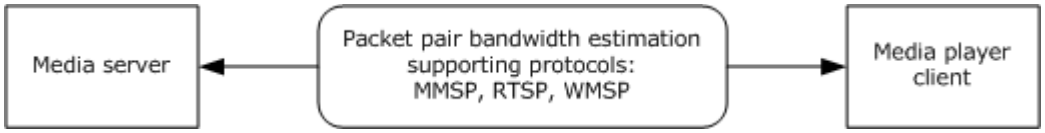

Figure 3: Packet-pair bandwidth estimation

3.1.11 Fast Start

Fast Start allows the player to buffer at speeds higher than the bit rate of the content requested. This enables users to start receiving content more quickly. After the initial buffer requirement is fulfilled, on-demand and broadcast content streams at the bit rate defined by the content stream.

Fast Start also allows a distribution server to request the data from the origin server at a faster bit rate. The bit rate specified in the Fast Start protocol headers ensures that the distribution server has enough data buffered to meet its needs and the needs of the media player client. In order to enable Fast Start, the protocols use two headers or tokens to request Fast Start:

- **Accelerate headers:** The tokens that the client uses to request a higher transmission rate and duration from the server.
- **Burst headers:** The tokens that the client uses to request a higher transmission rate and duration from the server. The client that sends the request is usually an intermediate device that is relaying the request for another client.

The Fast Start headers are initially set by the media player client. When going through a distribution server, additional burst headers are added.

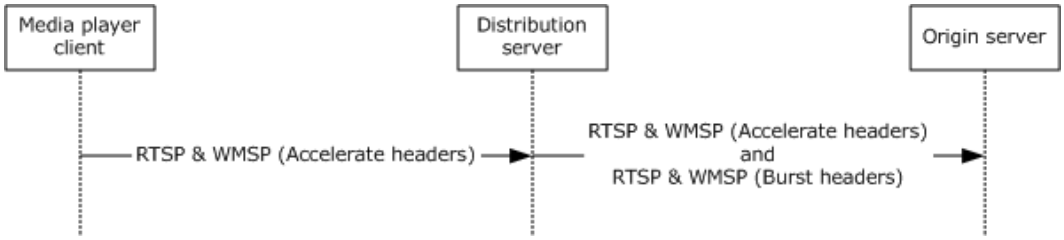

Figure 4: Fast Start

The figure shows the media player client requesting a preferred bit rate and duration to receive packets from the Distribution Server. The distribution server forwards these headers on to the origin server along with the burst headers, which are targeted at requesting an increased bit rate and duration from the Origin Server.

The distribution server will set the burst headers to be greater than the accelerate headers, therefore allowing the distribution server (acting as the client) to fill its buffer faster than the media player client is using it. The amount of additional data that is recommended for the distribution server will depend on the specific network configuration.

Fast Start is only supported by WMSP and RTSP-WME protocols.

3.1.12 Advanced Fast Start

Advanced Fast Start is designed to minimize startup latency in the media player client. Startup latency is the period of time starting when a viewer requests a stream by using the player and ending when the content begins playing. The primary reason for startup latency is the delay caused by buffering on the client.

Advanced Fast Start enables the media player client to begin playing a stream before its buffer is full. As soon as the media player client receives a minimum amount of data, it can begin playing a stream while its buffer continues to fill at an accelerated rate—a rate that is faster than the encoded bit rate of the content. When the buffer is full, acceleration stops and the media player client begins receiving data at the encoded bit rate.

For Advanced Fast Start to work effectively, adequate bandwidth must be available above the encoded bit rate of a stream. For example, if 1,200 kilobits per second (Kbps) of bandwidth is available for an 800 Kbps stream, the media player client can use an acceleration rate of 1.5 times the encoded bit rate. If no additional bandwidth is available, the player must fill its buffer before it begins playing a stream and no benefit can be gained from either Advanced Fast Start or Fast Start.

The following process describes how Advanced Fast Start works after the player sends a request and the server has successfully located the requested content and authorized the player to receive it:

1. The client indicates support for the Advanced Fast Start feature.

2. The server analyzes the content and calculates the following data for a number of acceleration rates:

   Largest underflow amount. Underflow occurs when the player renders the data faster than the player is receiving data. When underflow occurs, the player must stop and buffer more data. Therefore, underflow must be avoided to maintain a quality user experience. The largest underflow amount is the largest amount of extra data that the client will require during a given acceleration period to avoid underflow. The higher the acceleration rate and longer the

acceleration period, the less likely it is that underflow will occur. The server initially determines a fixed acceleration period.

Time of underflow. The time that the largest underflow amount occurs.

The X-StartupProfile header contains buffering information for different acceleration rates 1.0, 1.2, 1.5, 2.0, and 3.0 as specified in [MS-WMSP] section 2.2.1.12 and [MS-RTSP] section 2.2.6.28. For example, the data might show that with an acceleration rate of 1.2, the client will need 150 KB of additional data in its buffer after five seconds to avoid underflow.

3. The player calculates the current bandwidth between the client and server, and then requests the most appropriate acceleration rate from the server.

4. The server streams the content at the requested acceleration rate.

5. The player buffers the minimum amount of data, checks to make sure that adequate data has been received, and then begins playing the content. Meanwhile, its buffer continues to fill at the requested acceleration rate.

6. When its buffer is full, the player sends a message to the server to end acceleration, and begins streaming at the encoded bit rate.

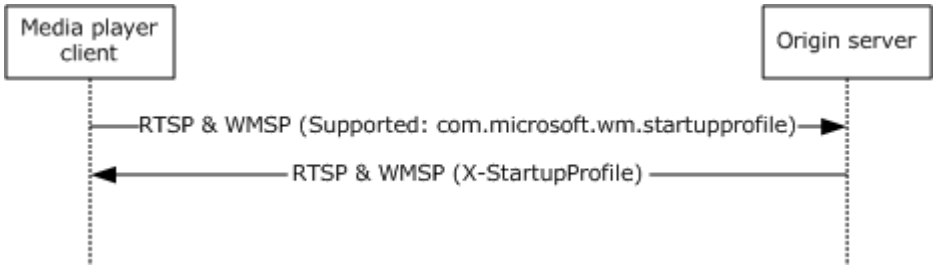

Figure 5: Advanced Fast Start header

The server responds to the client-supplied token on the Supported header, com.microsoft.wm.startupprofile, with the Advanced Fast Start header as specified in [MS-WMSP] section 2.2.1.12 and [MS-RTSP] section 2.2.6.28.

Advanced Fast Start is used only by clients that connect to a unicast stream and is supported only by the WMSP and RTSP-WME protocols.

3.1.13 Unicast Streaming

Unicast streaming is a one-to-one connection between the server and a client, which means that each client receives a distinct stream and only those clients that request the stream receive it. The server can deliver content as a unicast stream from either an on-demand or a broadcast **publishing point**. Unicast streaming offers the benefits of interactivity between the player and server. However, the number of users that are able to receive unicast streams is limited by the bit rate of the content, the speed of the server network, and the available server resources. The number of users served is directly proportional to the amount of available server resources and instances.

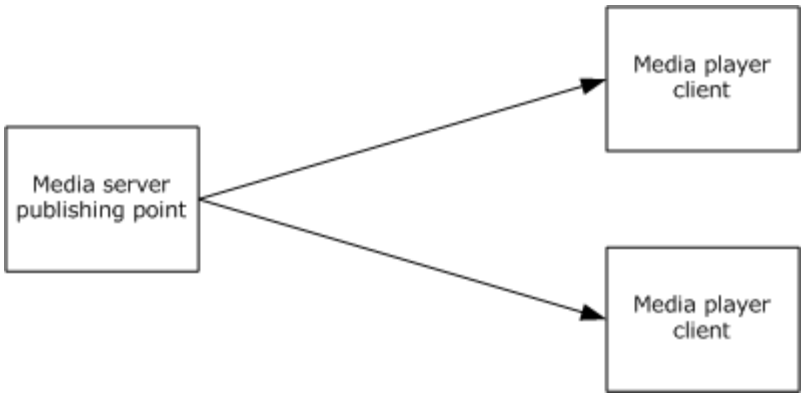

Figure 6: Unicast streaming

3.1.14 Multicast Streaming

Multicast streaming is a one-to-many relationship between a media server and the media player clients receiving the stream. With a multicast stream, the server streams to a multicast IP address on the network, and clients receive the stream by subscribing to the IP address. All clients receive the same stream and do not have control of content playback. Because there is only one stream from the server regardless of the number of clients receiving the stream, a multicast stream requires the same amount of bandwidth as a single unicast stream containing the same content. Therefore, multicast streaming improves the scalability of the server-side resources. More clients can be serviced with fewer server resources.

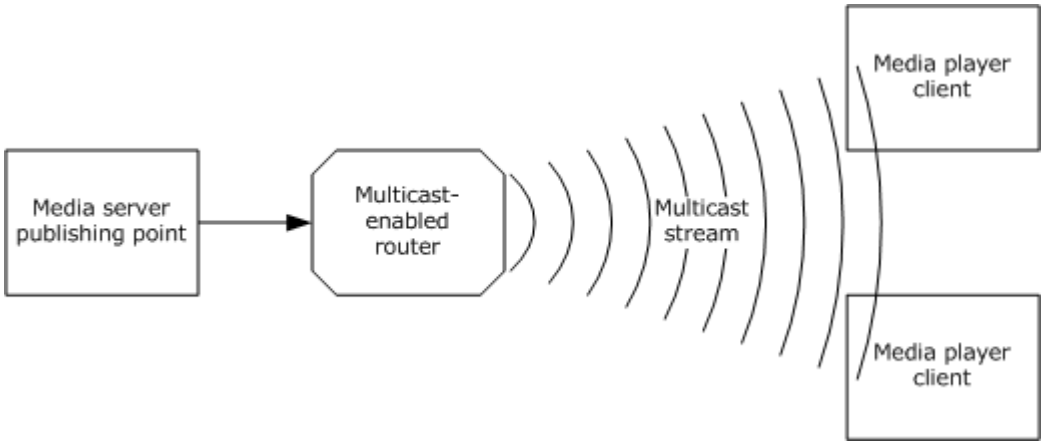

Figure 7: Multicast streaming

3.1.15 UDP Packets

The User Datagram Protocol (UDP) is a core level network protocol used by a number of individual protocols in the Media Streaming Server (MSS) System protocol family as stated in section 2.2. UDP is an unreliable protocol, which means there is no guarantee of the successful receipt of the packets. The packets can be dropped, arrive out of order, or be replicated.

3.1.16 TCP Packets

The Transmission Control Protocol (TCP) is one of the core level network protocols. TCP is a reliable protocol which means that the packets are guaranteed to arrive in the order specified.

3.2 System Purposes

The Media Streaming Server System includes protocols for communicating data packets that originate from downloadable and streaming audio, visual, and other multimedia data files.

The main purpose of the system is to deliver real-time or downloadable audio-visual content via the transfer of streams from a server to a single client or multiple clients. As specified in section 2.2, there are various protocols available to use to perform this task.

3.3 System Use Cases

3.3.1 Stakeholders and Interests Summary

The Stakeholders in the Media Streaming Server System are described as follows.

Administrators: Administrators in the corporate environment are responsible for setting up the network and broadcasts. They configure clients and servers to guarantee a certain level of quality and security of the content. For example, the administrator can limit the client access to the server, or direct user access to the server though a proxy server. In addition, to avoid bandwidth concerns, the administrator can select a protocol that multicasts rather than streams unicast in order to eliminate some network overhead.

Internet Content Provider (ICP): An ICP is the primary user of the streaming media family. Their role is to provide high quality media to the consumer. Which protocol is used is dependent on the scenario that they are trying to achieve and the reach they are trying to have.

The ICP initially needs to decide if the content is to be a live stream, download, or on demand stream. Depending on the selection, the ICP will choose the protocol to broadcast in most appropriate to his needs.

ICPs often find themselves needing to go beyond streaming and incorporate Digital Rights Management around the files or streams, or creating complex playlists in order to achieve their scenario goals.

IT Professionals: IT Professionals (IT Pro) are those administrators in the corporate environment responsible for setting up the network and broadcasts. They configure clients and servers to guarantee a certain level of quality and security of the content. They configure clients to access certain proxy settings, or they can disable access as appropriate. In addition, to avoid bandwidth concerns, the IT Pro can choose a protocol that multicasts rather than streams unicast in order to eliminate some network overhead.

End Users: Generally streaming media is targeted towards creating the experience demanded by consumers or End Users. In the corporate environment the End User is the individual viewing the live broadcast of the company meeting or the individual watching the corporate compliance video at a convenient time.

Outside of the corporate environment, the End User is the consumer who purchases a subscription to a Web site to access their live streams, or purchase media and download it for use later.

Media Server: The media server is the server that receives media from an encoder and streams it to the media player clients. The media server can act as an origin server or a distribution server.

Encoder: The Encoder is an application for converting both live and prerecorded audio and video content to digitized media format.

DRM Packager: The DRM Packager is a tool that is used to package media files that conform to the Advanced Systems Format (ASF) specification in an encrypted file format. When an ASF file is packaged, a DRM-specific section is added to the ASF file header containing business usage and distribution rules.

Media Player Clients: The media player client is the application that renders the media stream provided by the media server. This is the primary interface to the MSS System for the End User.

3.3.2 Supporting Actors and System Interests Summary

There are no systems in which the MSS System is an actor.

3.3.3 Use Case Diagrams

The following table provides an overview for the groups of use cases that span the functionality of the Media Streaming Server System. The sections that follow provide detailed descriptions of the use cases in each group.

Use case group	Use cases
Publish Content	Publish Content to Media Server -- Encoder Publish Secure Content to Media Server -- DRM Packager
Stream Content	Stream Content from Media Server -- Media Player Client Request License from License Server -- Media Player Client Log Statistics to Servers -- Media Player Client

The following diagrams illustrate the use cases in each use case group.

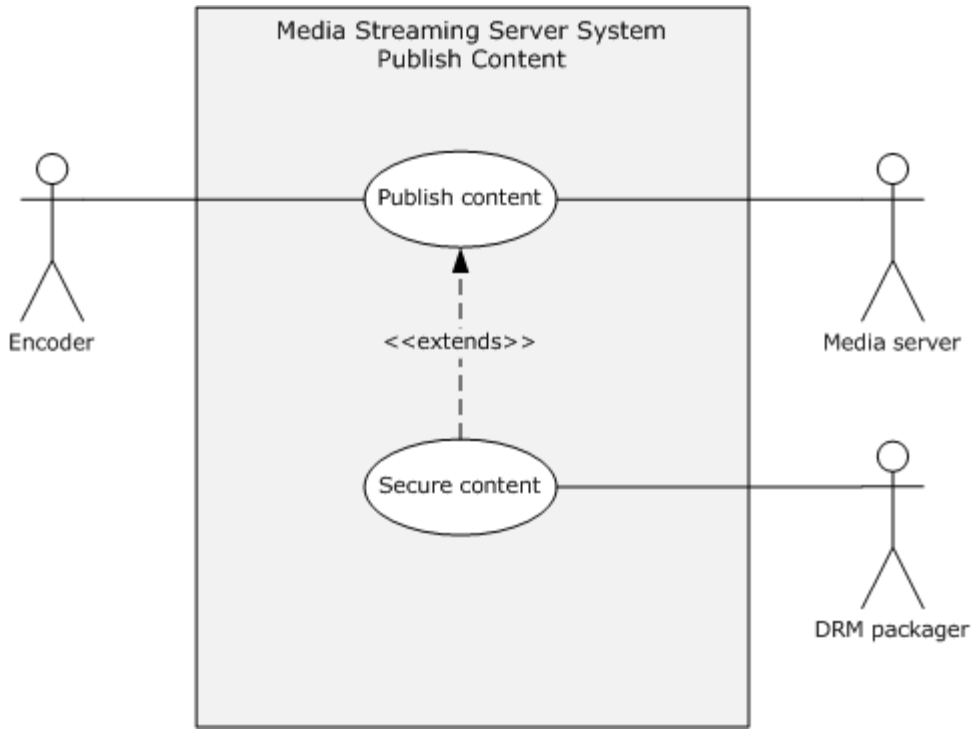

Figure 8: Publish Content use case group

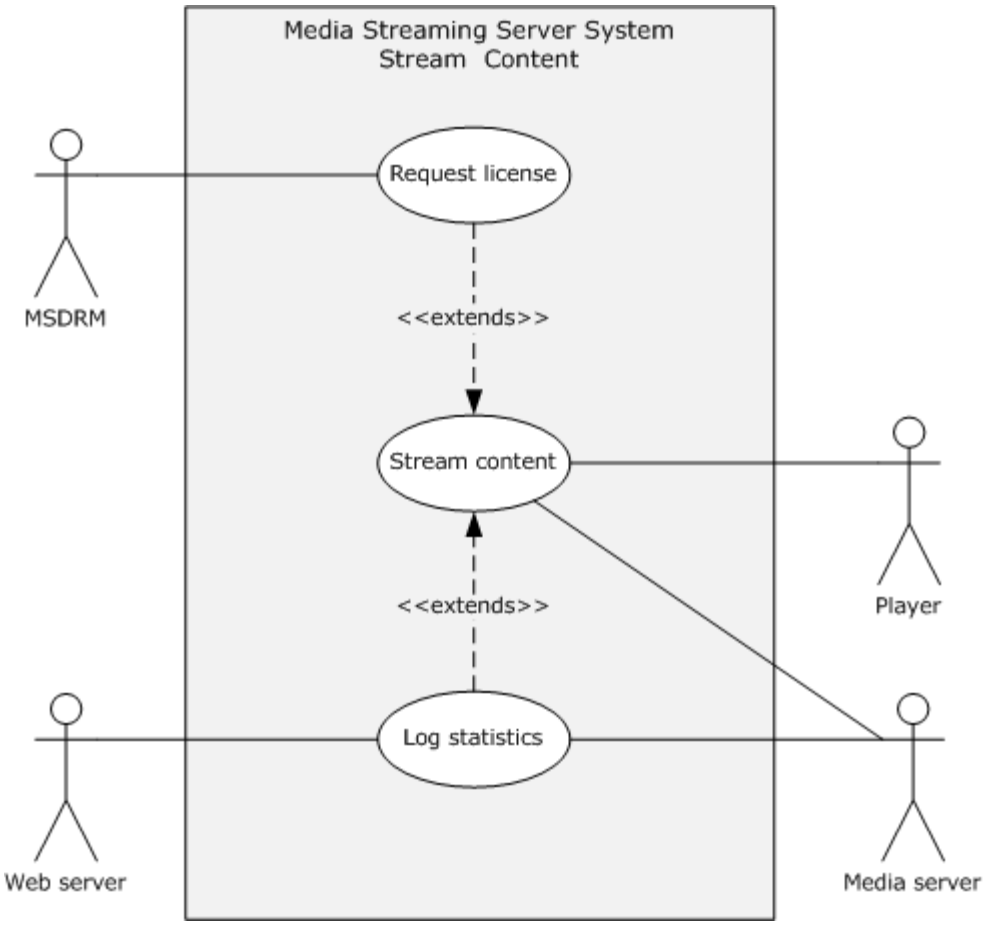

Figure 9: Stream Content use case group

3.3.4 Use Case Descriptions

3.3.4.1 Publish Content to Media Server—Encoder

This use case has two variations:

(a) Encoder pushing content to the Media Server.

(b) Media Server pulling content from the Encoder.

Goal: To create content for broadcast through the Media Streaming Server System.

Context of Use: The captured media is pushed to a Media Server for distribution on the network.

Direct Actor: The direct actor of this use case is the Encoder.

Primary Actor: The primary actor is the Internet Content Provider.

Supporting Actors: The Media Server in this case is the supporting actor. It is receiving the content from the Encoder.

Stakeholders and Interests:

The stakeholders for this scenario would be the Internet Content Providers and Administrators for creating the content and the Media Server for receiving the content.

Preconditions:

- The encoder is available on the MSS System network to the Media Server.
- The encoder has to be configured to capture live streams or recordings.
- When pushing content, the encoder has to have access to the Media Server publishing point, or be able to create a publishing point on the server.
- If the encoder is pushing to the Media Server, then the ports are required to be opened on the server to receive the media stream.
- If the server is pulling from the Encoder, then the ports need to be opened on the encoder. If using the WMSP protocol, any port is possible. Generally, the Server initiates an HTTP connection with the encoder through port 8080.
- The network supports HTTP and TCP, or UDP.<1>

Minimal Guarantees: Failure in the network will prevent the multimedia content from reaching the media server.

Success Guarantee: Multimedia content will reach the Media Server and be available for distribution.

Trigger: If the Encoder is pushing content to the server, the trigger is the Encoder sending the stream to the server. If the server is pulling content from the Encoder, the trigger is the server initiating the connection to the Encoder.

Main Success Scenario:

Variation (a) Push mode

1. The encoder establishes a connection to the server.
2. The encoder sends a push request to the media server.
3. The encoder begins to capture content.
4. The encoder pushes the multimedia stream to the media server.

Variation (b) Pull mode

1. The encoder begins to capture content.
2. The server connects to the encoder.
3. The encoder and server then exchange messages allowing the server to pull the multimedia stream to the media server.

Extensions:

Extension (a) Push Mode

3a. DRM can be used to extend this scenario by packing the contents within a protected package as described in section 3.3.4.2.

Extension (b) Pull Mode

1a. DRM can be used to extend this scenario by packing the contents within a protected package as described in section 3.3.4.2.

3.3.4.2 Publish Secure Content to Media Server—DRM Packager

Goal: To provide protected content to the Media server for streaming.

Context of Use: The Media server is configured correctly to broadcast content. The DRM Packager encapsulates the content so that it can be sent securely to the client.

Direct Actor: The direct actor of this case is the Encoder that encapsulates the media content.

Primary Actor: The primary actor is the Internet Content Provider.

Supporting Actors: The DRM Packager is the supporting actor. It is packaging the streams that are then made available on the media server.

Stakeholders and Interests: The stakeholders for this scenario are the Administrators, Media server, Encoder, and DRM Packager.

Precondition: The encoder already has the license key seed, certificate values, and signing keys as described in [MSDN-LPS-WME].

Minimal Guarantees: Media files that are encrypted cannot be decrypted unless the client has a certificate (as specified in [MS-DRM] section 2.2.3.2.7), that allows it to decrypt the content.

Success Guarantee: Media files that are encrypted cannot be decrypted unless the client has a certificate (as specified in [MS-DRM] section 2.2.3.2.7), that allows it to decrypt the content. Encrypted files exposed on the Media server are available for access by authorized media player clients.

Trigger: The encoder is instructed to protect the stream.

Main Success Scenario:

1. The Encoder requests encryption for the stream it is going to capture.

2. The DRM Packager encrypts the stream.

Extensions: None.

3.3.4.3 Stream Content from Media Server - Media Player Client

Goal: To stream content from Media servers.

Context of Use: The Media server is configured to provide media streams and files to the network. The Media Player client then plays back those files and streams.

Direct Actor: The direct actor of this use case is the Media Player client.

Primary Actor: The primary actor is the End User.

Supporting Actors: The Media server in this case is the supporting actor. It is providing the media stream to the player.

Stakeholders and Interests: The stakeholders for this scenario are the Administrators, End Users and the Media Player client.

Preconditions: In order to access content from the Media server, the Media Player client must be preconfigured to access the server. In particular, the Media Player client has to have read permission to access the content on the server, and the media player client has to support the protocol that is providing the playback experience. In addition, the network used has to support HTTP or UDP or multicast depending on the protocol used.

Minimal Guarantees: The Media Player client will be notified with an error, if the Media Player client cannot connect successfully to the server.

Success Guarantee: The Media Player client will be able to stream media stream from the Media server.

Trigger: The Media Player client requests a stream from the Media server.

Main Success Scenario:

1. The Media Player client connects to the stream from the Media server.
2. In the event the media is not multicast, the Media server evaluates the request and then acts upon the request.
3. Depending on the configuration and settings, the Media server might stream the file via unicast or multicast to the Media Player client.

Extensions:

1a. WEB servers can host files that Web pages and **redirector files**, like **.asx files** and **.nsc files**, use for discovering the media server content.

3a. Logging is an extension to streaming as specified in section 3.3.4.5.

3b. Enabling the DRM server to issue licenses to the Media Player client to enable playback is an extension to the streaming content from the Media server as specified in section 3.3.4.4.

3.3.4.4 Request License from License Server - Media Player Client

Goal: To provide and enable playback for DRM protected media streams.

Context of Use: When attempting to stream the file, the media player client will need to obtain a license to be successful.

Direct Actor: The direct actor of this use case is the Media Player client.

Primary Actor: The primary actor is the End User.

Supporting Actors: The DRM license server is the supporting actor, issuing the licenses, as well as the DRM packager, which packages the content.

Stakeholders and Interests: The stakeholders for this scenario are the Administrators, Internet Content Providers, End Users, and the Media Player client.

Precondition: The media server is configured correctly to broadcast content. The DRM packager has packed the content such that it requires a license.

Minimal Guarantees: Media files that are encrypted cannot be decrypted unless the media player client has a certificate or license that allows it to decrypt the content.

Success Guarantee: Media Player clients that successfully obtain a license for the content will be able to render the stream.

Trigger: The Media Player client determines that it cannot play the file without a license.

Main Success Scenario:

1. The Windows Media Player opens protected digital media and determines it cannot play it without a license.
2. The Windows Media Player contacts the license server and presents the user's license request information.
3. The license server issues the license that allows the Windows Media Player to play the media.

Extensions: None.

3.3.4.5 Log Statistics to Servers - Media Player Client

This use case has two variations:

a) Logging to a Media server

b) Logging to a Web server

Goal: To obtain statistics for the Media Streaming Server System experience.

Context of Use: The Media Player client can provide statistics during the playback experience allowing the Media server to optimize future playback scenarios.

Direct Actor: The direct actor of this use case is the Media Player client.

Primary Actor: The primary actor is the same as direct actor.

Supporting Actors: The supporting actors for this scenario can be both the Media server and a Web server that can receive the Media Player client statistics.

Stakeholders and Interests: The stakeholder for this scenario is the Media server or Web server.

Preconditions:

- The Media server has to be configured to process the statistics, including installing necessary plug-ins.
- If the Web server is the recipient of the log file, the Media Player client has to be notified of the alternative destination for the WMLog.
- The receiving servers have to be configured to receive logging messages.

Minimal Guarantees: Failure to generate logs will not prevent the system from streaming.

Success Guarantee: The Media server will receive logging information that will communicate usage and performance for its streamed media.

Trigger: The trigger for sending logging messages is the Media Player client.

Main Success Scenario:

One of the primary functions of a Media Player client is to play back content that is streamed over a network. To provide this service, it is necessary for the Media Player client to communicate with a streaming media server.

During playback of content, the Media Player will send logging messages to the streaming media server. The Administrator can also configure the Media Player to send the logging messages to a Web server as defined in the protocol family.

The Media Player client can provide different types of logs:

- Streaming Logs: The Streaming Log specifies how the client received streaming data but not how the client rendered the data. Streaming logs can be sent to either a Windows Media server or a Web server.
- Rendering Logs: The Rendering Log describes playback of content by a client and is submitted to the media server or a Web server when the client ends playback.
- Legacy Logs: The Legacy Log contains both rendering and streaming information and can be sent to either a Windows Media server or a Web server.
- Connect-Time Logs: The purpose of the Connect-Time Log is to specify some minimal amount of logging information about the client. Connect-Time Logs are only defined for Windows Media servers.

Variation (a) Logging to a Media server:

1. The Media Player client successfully plays the stream
2. The Media Player client submits the log to the media server

Variation (b) Logging to a Web server:

1. The Media Player client successfully plays the stream
2. The Media Player client submits the log to the Web server

Extensions: None.

3.3.4.6 Discover Media Server URLs – Media Player Client

Goal: To use a Web server to provide redirection to the media server content URLs.

Context of Use: The Administrator uses a Web server to host pages and redirector flies like .asx files, and .nsc files, for providing links to the media server content.

Direct Actor: The direct actor of this use case is the Media Player client.

Primary Actor: The primary actor is the Administrator.

Supporting Actors: The supporting actors for this scenario can be both the Media server and a Web server.

Stakeholders and Interests: The stakeholder for this scenario is the Administrator, Media server or Web server.

Preconditions:

- The Media server has content available to be streamed and the URL is known.

- The Web server and Media server are on networks available to the Media Player client.

Minimal Guarantees: Failure to create valid redirection will not prevent the media server from streaming.

Success Guarantee: The Media Player client will be able to access the Media server without the user knowing the specific Media server URL.

Trigger: The trigger is content is made available on the Media server.

Main Success Scenario:

1. The Administrator enables content for streaming as suggested in section 3.3.4.3.
2. The Administrator configures the Web server to host redirector files or Web pages with links to the media server content.
3. The End User discovers the content on the Web server and clicks the link.

Extensions: None.

4 System Context

This section describes the relationship between this system and its environment.

4.1 System Environment

The Media Streaming Server (MSS) System typically consists of a computer running and encoder, a media server and a number of client computers running a media player. The encoder converts both live and prerecorded audio to a digital format and the media server distributes the content over a network or the Internet. Players then receive the content. The MSS System also includes a Web server as an optional component. The Web server is used for serving Web pages (which have URLs to the media server), .asx files, and .nsc files. The Web server can also be used for receiving WMLOG messages.

Other optional components include the DRM license server and the DRM packager. These optional components allow the MSS system to stream encrypted content to the media player client.

For its success, the MSS System depends on a number of prerequisite factors for it to be configured and used by server and client computers. There are core networking protocols and services that are required to be open, running, and configured in order to correctly communicate with each other.

The network has to be capable of supporting TCP/IP traffic such as TCP and UDP. Firewall ports are required to be opened in order to all network traffic to flow between clients, servers, and encoders. For authenticated streaming, the server, encoders, and clients are required to support NTLM ([MS-NLMP]) or digest.

Finally, in some cases the protocol does not provide a mechanism for a client to discover the URL to the server. Therefore, the client will need to discover this data another way. This is often done by putting a URL to the server as a hyperlink in a Web page. It also can be done by way of a redirector file such as an .nsc file or .asx file.

In general, environment assumptions and preconditions are dependent on features and functional modes expected for the media streaming solution.

Specific individual protocols as listed in section 2.2 are used depending on the scenario, and the media server and client available on the network.

If the system requires interactivity by the media player client, then the system will provide a unicast transmission, and subsequently a protocol that supports unicast delivery.

Systems that require broadcasting media to a large audience, and where network bandwidth and server capacity are limited, will choose multicast delivery. A multicast broadcast may add additional requirements on the network. For example, networks must use multicast-enabled routers if they use routers with the multicast-enabled MSS System.

Port settings for the various protocols are generally different depending on which MSS protocol is used, which network protocol is used, and whether or not you are multicasting or unicast streaming.

If the system requires clients to authenticate before streaming content, the media server will need to employ the authorization function of the protocols. If the media server or encoder allows anonymous connection, then these authorization functions are not needed.

The following table summarizes the unicast, multicast, UDP, and TCP support provided by each of the MSS System Member Protocols.

Protocol features

Protocol	Multicast or unicast	UDP support	TCP support	Client redirection file required
[MS-RTSP]	Unicast	Used for transmitting RTP and RTCP packets.	Used for transmitting RTP and RTCP packets and normally used for controlling the streaming media session.	No
[MS-MMSP]	Unicast	Used for transmitting data packets and can be used to request a packet resend.	Used for transmitting data packets and for controlling the streaming media session.	No
[MS-WMSP]	Unicast	None	TCP Only	No
[MS-WMHTTP]	Unicast	None	TCP Only	N/A
[MS-MSBD]	Multicast Unicast	UDP encapsulation mode is only used for transmitting packets to an IPv4 multicast group	MSBD packets are encapsulated in TCP	No
[MS-MSB]	Multicast	MSB packets are encapsulated in UDP	None	Yes

4.1.1 Authentication

The MSS System can optionally support authentication; however, there are limitations and dependencies that need to be considered, as follows:

- If using the WMHTTP or RTSP-WME, then the system has to support HTTP access authentication as specified in [RFC2616] section 11.
- For WMSP, the authentication system is based on HTTP 1.0, and therefore, in order to support authentication, the client and servers are required to support access authentication, as specified in HTTP 1.0 [RFC1945] section 11.
- MMSP supports Basic authentication (as specified in [RFC2617]) and NT LAN Manager (NTLM) authentication (as specified in [MS-NLMP]).
- MSB and MSBD do not support authentication natively.

4.1.2 Media Player Client

In order for media player clients to stream content from the Media Streaming Server System, they are required to be able to stream content using the protocol provided by the server. Typically the players will support all protocols. The following protocols can be supported by the client software:

- Microsoft Media Server (MMS) Protocol: [MS-MMSP]
- Media Stream Broadcast (MSB) Protocol: [MS-MSB]
- Real-Time Streaming Protocol (RTSP) Windows Media Extensions: [MS-RTSP]

- Windows Media HTTP Streaming Protocol: [MS-WMSP]
- Media Stream Broadcast Distribution (MSBD) Protocol: [MS-MSBD]

4.1.3 Encoder

The streaming media system depends on a source for content, as described in section 3.3.4.1. This can be obtained using an encoder. The encoding computer is typically networked to the Media Streaming Server. The encoder will typically enable both "push" and "pull" distribution. With push distribution, the encoder initiates the connection with a Media server and passes the content to the server. With pull distribution, a player or a Media server connects to an HTTP port on your computer to receive the content. Encoder to server protocols are as follows:

- Windows Media HTTP Push Distribution Protocol (WMHTTP): [MS-WMHTTP]
- Media Stream Broadcast Distribution Protocol (MSBD): [MS-MSBD]
- Windows Media HTTP Streaming Protocol (WMSP): [MS-WMSP]

4.2 System Assumptions and Preconditions

Given the environment described in section 4.1, the system has the following assumptions and preconditions:

- Encoders pushing media to media servers have the right to push to a publishing point or create a new publishing point from a template publishing point.
- A publishing point has been configured on the media server in order to accept a push from the encoder.
- The media server receiving a push has to be preconfigured to receive the stream. This includes opening ports as specified in section 4.1. For individual protocol port settings, see the individual protocol documents.
  - For RTSP-WME port settings, see [MS-RTSP] section 2.1.
  - For MMSP port settings, see [MS-MMSP] section 2.1.
  - For WMSP port settings, see [MS-WMSP] section 2.1.
  - For MSB port settings, see [MS-MSB] section 2.1.
  - For MSBD port settings, see [MS-MSBD] section 2.1.
  - For WMHTTP port settings, see [MS-WMHTTP] section 2.1.
- All media servers are required to be configured to stream across a computer network.
- All players are required to be configured to support the server's specified protocol. Failure to confirm that all players support the protocol might result in some players' failure to play as specified in section 4.1.
- Multicast streaming requires routers to be multicast enabled.
- Web servers that support logging are required to be configured to support POSTs ([RFC2616]) from the client.

- Players are required to be able to access the license server at the same time as they are accessing the media, unless the player obtained the license prior to playback. This might require players to have multiple connections to the Internet or intranet to play a stream.

Member protocols supported by the system, as listed in section 2.2, might have additional assumptions and preconditions when that protocol is being used. Please see the relevant member protocol specification for details.

Unicast streaming is supported with the following protocols:

- Microsoft Media Server Protocol (MMSP): [MS-MMSP]
- Windows Media HTTP Streaming Protocol (WMSP): [MS-WMSP]
- Real-Time Streaming Protocol Windows Media Extensions (RTSP-WME): [MS-RTSP]
- Windows Media HTTP Push Distribution Protocol (WMHTTP): [MS-WMHTTP]
- Media Stream Broadcast Distribution Protocol (MSBD): [MS-MSBD]

Multicast streaming is a one-to-many relationship between a server and the clients receiving the stream. With a multicast stream, the server streams to a multicast IP address on the network and clients receive the stream by subscribing to the IP address. All clients receive the same stream and do not have control of content playback.

Multicast streaming does not work reliably on the Internet due to the fact that many routers are not multicast-enabled. In order to provide a multicast streamed event, the routers on the network are required to be multicast-enabled.

Multicast streaming is supported with the following protocols:

- Media Stream Broadcast protocol (MSB): [MS-MSB]
- Media Stream Broadcast Distribution protocol (MSBD): [MS-MSBD]

4.3 System Relationships

This section describes the relationships between the system and external components, system dependencies, and other systems influenced by this system.

The MSS System protocols are not technically dependent on each other; they are implemented independently of each other. Unless otherwise specified, their only interdependencies are those of the individual protocols' underlying transport mechanisms, such as HTTP, TCP, and UDP.

4.3.1 Black Box Relationship Diagram

At a high level, the MSS System provides a way for an encoder application to provide content to a media server and for a media player client application to stream from a media server. The following diagram illustrates this concept.

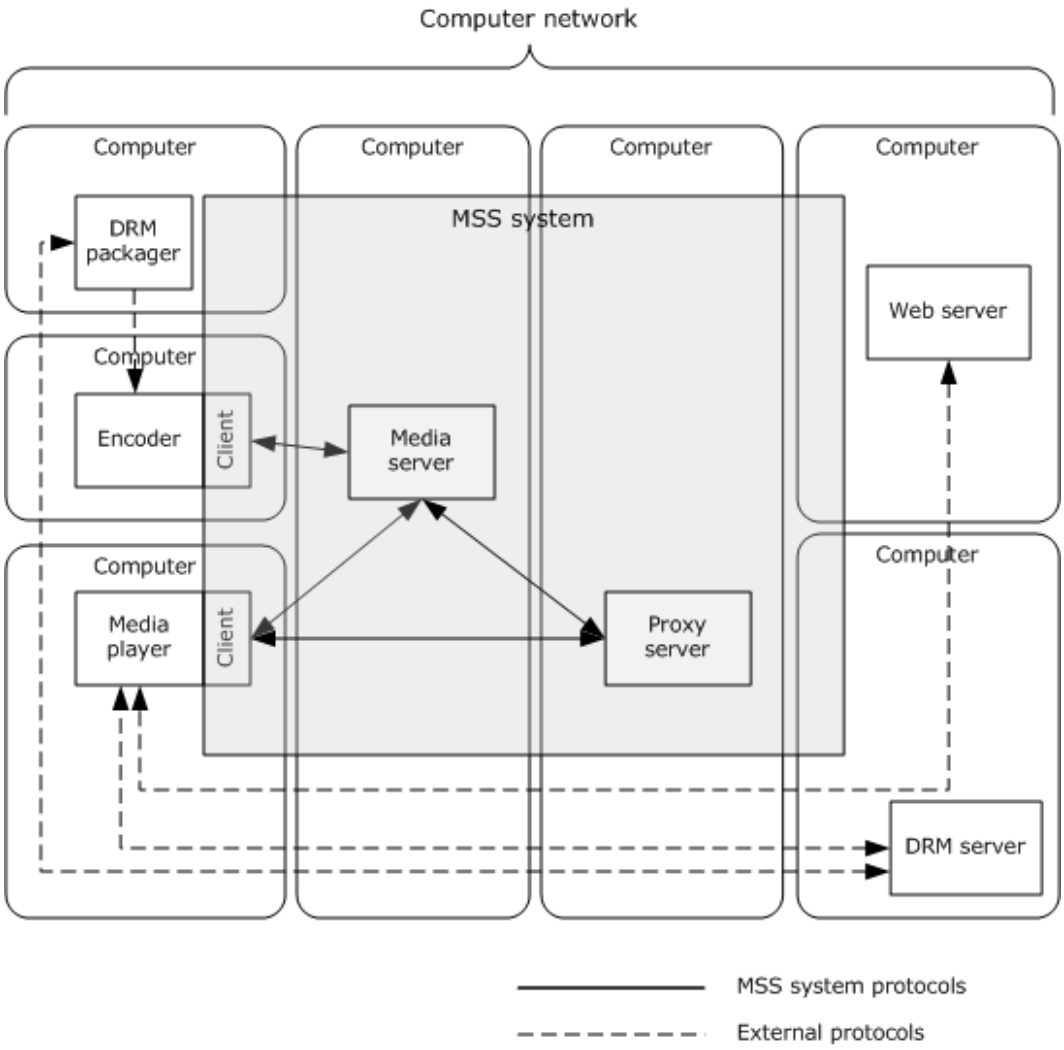

Figure 10: MSS System black box diagram

Roles that use MSS System protocols are as follows:

- Encoder: An encoder application that wants to push or transmit digital media to a media server.
- Player: A playback application that wants to stream digital media from a media server.
- Proxy Server: A media server acting as a proxy server.

Roles that use system external protocols are as follows:

- Player: A playback application that wants to decrypt and playback DRM encrypted digital media from a media server.
- Web Server: A server that receives logging data from a Player.
- DRM Packager: An application used to encrypt media for secure playback.

4.3.2 System Dependencies

The MSS System has dependencies on physical devices, applications and network configurations. The dependencies may ship depending on the requirements of the scenario.

Physical Dependencies

The MSS System requires physical network connectivity and correctly configured network configuration on both the MSS server and the MSS client. There is no specific requirement for the type of physical networking topology.

Network Dependencies

If the MSS System is streaming a multicast stream, the MSS client has to determine the URL to the server. Since the media server protocol does not provide a mechanism for a client to discover the URL, the MSS client is dependent on a redirector file to provide information. The MSS System is dependent on the redirector file to provide that information.

In addition, the MSS System during multicast streaming is dependent on the UDP networking protocol to be available on the MSS network.

For unicast streaming the MSS client depends on an IP address of a correctly configured DNS server and the ability to connect and to be able to discover and resolve host names of MSS servers. The MSS client requires access to all the prerequisite MSS servers via TCP/IP and expects to access MSS server services via the TCP ports exposed by those services. See section 4.2 for port specific details.

The network might also support the UDP networking protocol. For UDP streaming, it requires access to all the prerequisite MSS servers via UDP and expects to access MSS server services via the UDP ports exposed by those services. See section 4.2 for port specific details.

Authenticated Streaming

For authenticated streaming, the MSS servers and clients pick up a dependency on an authentication scheme. The supported authentication scheme is based entirely on which protocol is being used. See section 4.1 for specific protocol details.

Application Dependecies:

- **Encoder Application** - The success of the MSS System is dependent upon digital media entering into the system. Digital media is created by the encoder and pushed to the server or pulled to the server. Therefore the MSS System has a dependency on the encoder as a content creator.
- **Player Application** - The success of the MSS System is dependent upon digital media entering into the system. Digital media is created by the encoder and pushed to the server or pulled to the server. Therefore the MSS System has a dependency on the encoder as a content creator.

MSS System protocols supported by the system, as listed in section 2.2, have additional dependencies when that protocol is being used. Please see the relevant member protocol specification for details.

4.3.2.1 Dependencies on MSS

MSS System dependencies are as follows:

- Player Applications: Player applications designed for streaming content from a media server might have a dependency on the MSS protocols.

- Encoder Applications: Encoder applications designed to push content to a media server might have a dependency on the MSS protocols.

4.3.3 System Influences

The MSS System is influenced by the following systems:

Windows Media Digital Rights Management (WMDRM), as described in [MS-DRM]: This system influences the playback experience for media in the MSS System. Rights imposed on the media file will prevent playback and generate error cases not normally seen in the MSS experience.

IIS Server: An IIS Server, as mentioned in section 4.1, can provide multiple options to the MSS System. It can be used by the administrator to obtain a log, as well as provide the player with a URL to the server in the Web page, or a redirector file (.nsc or .asx) that points the media player client to the stream.

Firewall: A network firewall can influence the system by altering the ability for the client to discover and communicate with the media server. When a server and clients are behind a firewall, it is important to open up the correct port to enable streaming. Please see section 4.2 for more specific information on port settings.

4.4 System Applicability

The MSS System used is suitable for streaming delivery of real-time multimedia data. The term streaming means that the data is transmitted at some fixed rate or at some rate that is related to the rate at which the data will be consumed (for example, displayed) by the receiver.

The applicability of each member protocol supported by the system, as listed in section 2.2 is dependent on the use cases in section 3.3 and protocol relevance described in section 4.1.

For individual protocol applicability, please see the specifications of the protocols supported by the system, as listed in section 2.2.

4.5 System Versioning and Capability Negotiation

There is no capability negotiation that is associated with this system. Any deviations from a specific version's implementation of these protocol specifications are documented in the respective protocol documents. Capability negotiations between client and server implementations of these protocols are specified in the Versioning and Capability Negotiation sections in their respective technical documents (TDs).

The following table describes the availability of a protocol on a given server system.

Operating system versions

Protocols implemented	Operating system versions
Microsoft Media Server (MMS) Protocol [MS-MMSP]	Microsoft Windows NT® Server 4.0 operating system Microsoft Windows® 2000 Server operating system Windows Server® 2003 operating system
Real-Time Streaming Protocol (RTSP) Windows Media Extensions	Windows Server 2003 Windows Server® 2008 operating system

Protocols implemented	Operating system versions
[MS-RTSP]	Windows Server® 2008 R2 operating system
Windows Media HTTP Streaming Protocol [MS-WMSP]	Windows NT Server 4.0 Windows 2000 Server Windows Server 2003 Windows Server 2008 Windows Server 2008 R2
Windows Media HTTP Push Distribution Protocol [MS-WMHTTP]	Windows Server 2003 Windows Server 2008 Windows Server 2008 R2
Media Stream Broadcast (MSB) Protocol [MS-MSB]	Windows 2000 Server Windows Server 2003 Windows Server 2008 Windows Server 2008 R2
Media Stream Broadcast Distribution (MSBD) Protocol [MS-MSBD]	Windows NT Server 4.0 Windows 2000 Server Windows Server 2003

The following table describes the availability of a protocol on a given client system.

Protocols implemented	Operating system versions
Microsoft Media Server (MMS) Protocol [MS-MMSP]	Microsoft Windows® 2000 operating system Windows® XP operating system
Real-Time Streaming Protocol (RTSP) Windows Media Extensions [MS-RTSP]	Windows XP Windows Vista® operating system Windows Server 2008 R2
Windows Media HTTP Streaming Protocol [MS-WMSP]	Windows 2000 Windows XP Windows Vista Windows Server 2008 R2
Media Stream Broadcast (MSB) Protocol [MS-MSB]	Windows 2000 Windows XP Windows Vista Windows Server 2008 R2
Media Stream Broadcast Distribution (MSBD) Protocol [MS-MSBD]	Windows 2000 Windows XP

4.6 System Vendor-Extensible Fields

The system does not define any vendor-extensible fields beyond those described in the specifications of the protocols supported by the system, as listed in section 2.2.

5 System Architecture

This section describes the basic structure of the system and the interrelationships among its parts, consumers, and dependencies.

5.1 Abstract Data Model

The MSS System protocols are not technically dependent on each other. They are implemented independently of each other, and do not share any data or state between the protocols supported by the system, as listed in section 2.2.

5.2 White Box Relationships

This section describes the logical dependencies among the media streaming protocols and protocol stack views, as appropriate.

5.2.1 Encoder Push Diagram

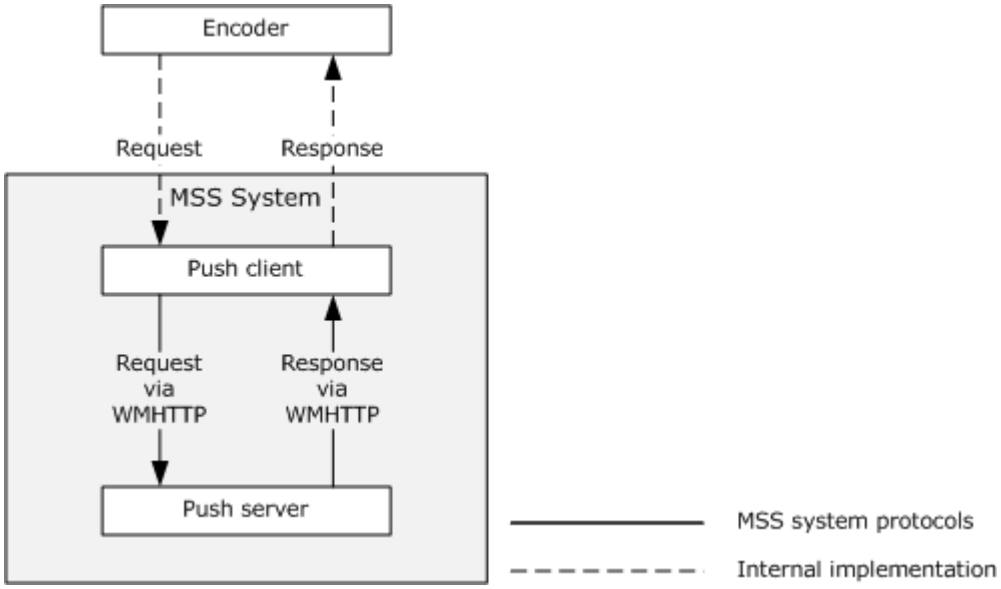

Figure 11: Encoder push model

The preceding figure represents the abstract model of the MSS System and the Encoder application. The encoder applications use the WMHTTP protocol to push the digitized video streams to the media server.

5.2.2 Encoder Pull Diagram

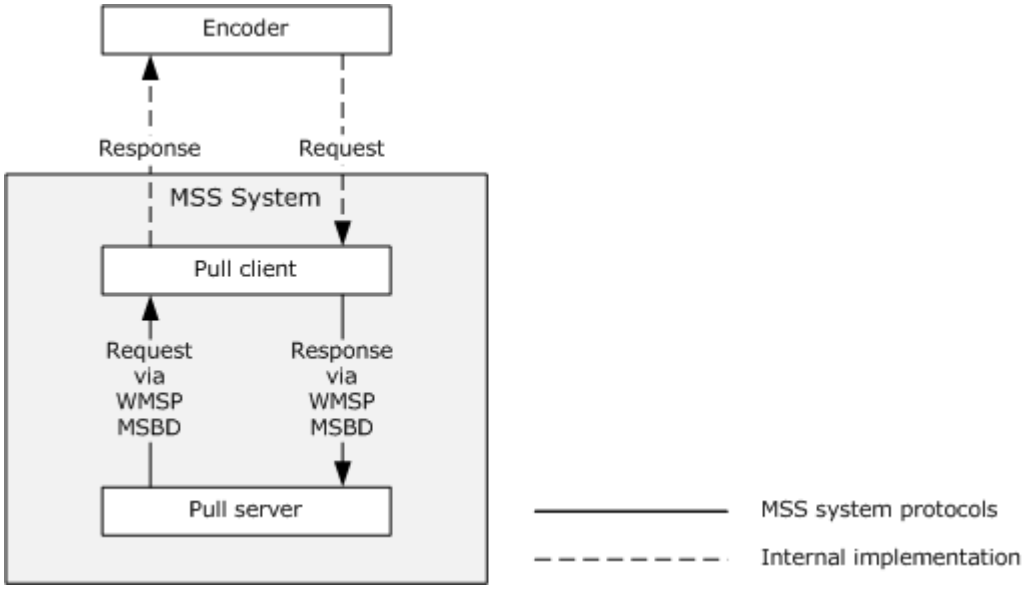

Figure 12: Encoder pull model

The preceding figure represents the abstract model of the MSS System and the Encoder application. The media server can use two MSS protocols to pull the digital media files and streams from the server. The MSS System would only use one of the protocols specified to accomplish the pull. Which protocol is used is based on the system platform. Please section 4.5 for more details.

5.2.3 Unicast Playback Diagram

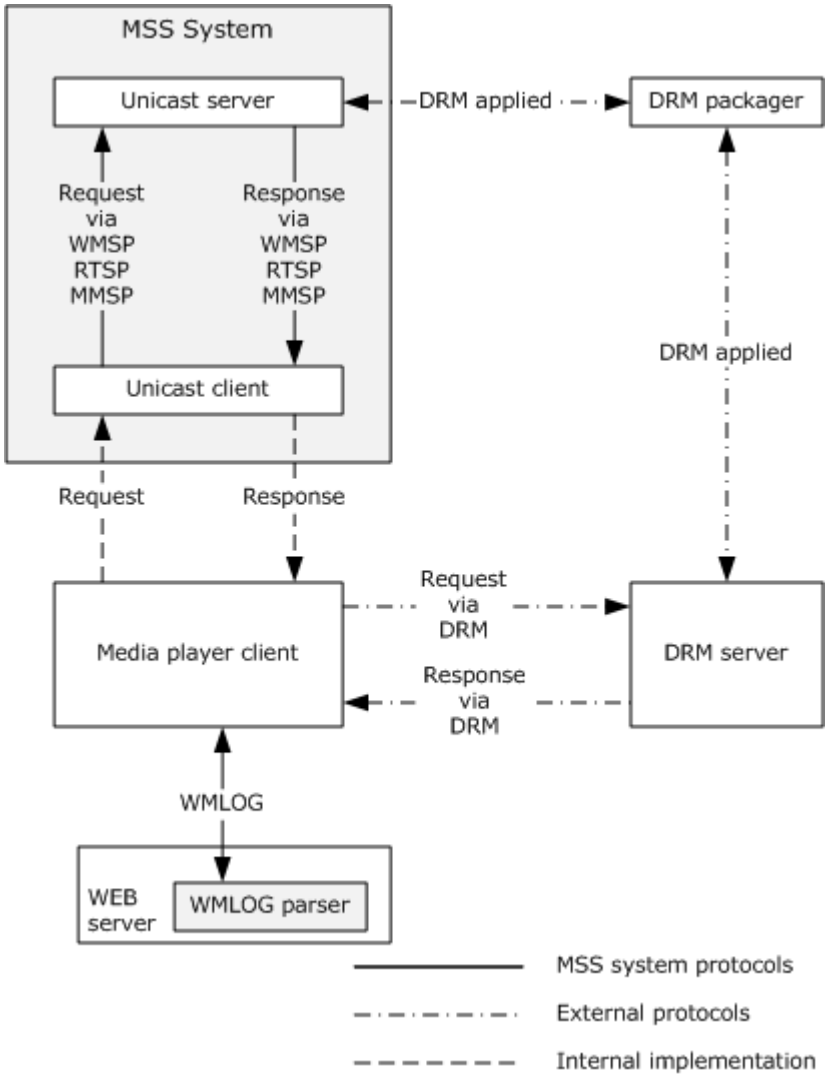

Figure 13: Unicast Playback protocol diagram

The preceding figure represents the MSS System during a unicast playback with the player application. The player application uses the system to pull media streams from the server. The communication between the unicast client and the unicast server will also include logging messages, as specified in the individual protocols.

As can be seen, the client server communication is supported with multiple protocols. Which protocol is used is determined based on the MSS System platform as specified in section 4.5.

5.2.4 Multicast Playback Diagram

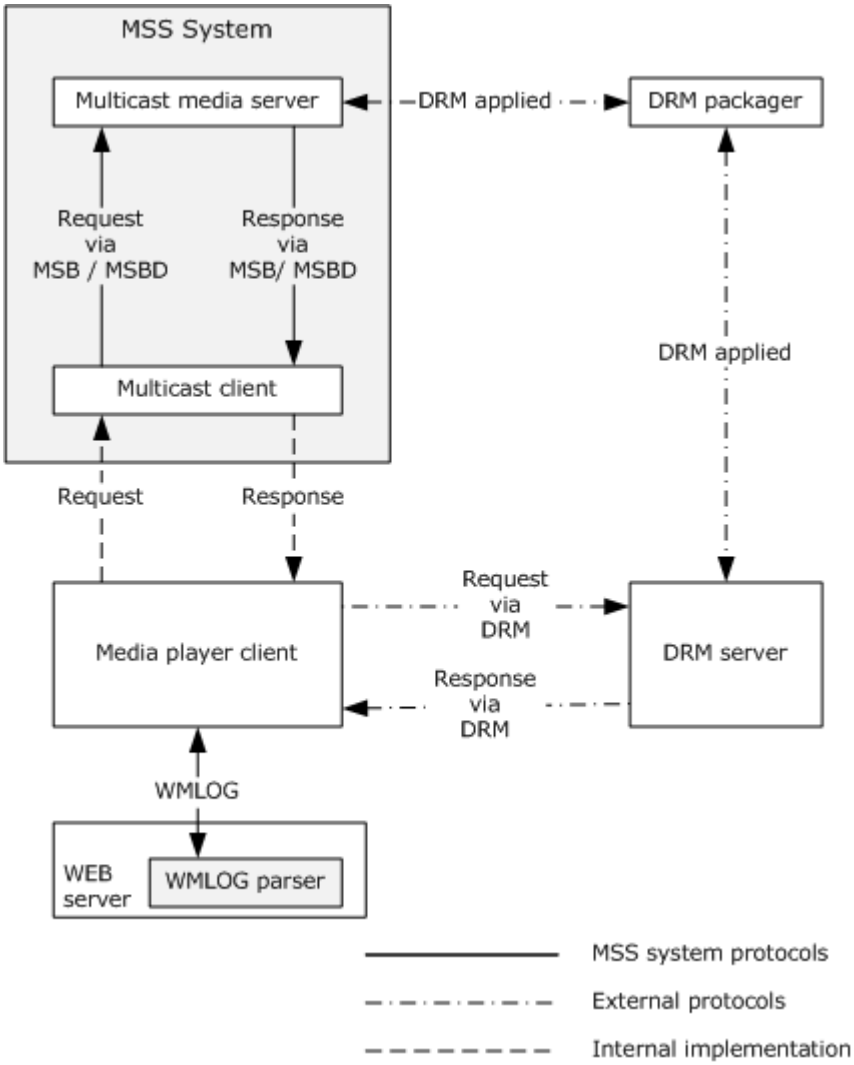

Figure 14: Multicast playback protocol diagram

The preceding figure represents the MSS System during a multicast playback between the server and the player application. The player application uses the system to pull media streams from the server.

5.3 Member Protocol Functional Relationships

5.3.1 Member Protocol Roles

The following table describes in more detail the individual protocol roles.

MSS System protocol roles

Protocol name	Protocol description	Document short name
Media Stream Broadcast (MSB) Protocol	Allows the multicast distribution of ASF packets over a network for which Internet Protocol (IP) multicasting is enabled. MSB allows clients to tune in to a broadcast on a network, much like television and radio users can tune to a particular television or radio station.	[MS-MSB]
Media Stream Broadcast Distribution (MSBD) Protocol	Used to transfer a live stream of audiovisual content from a server to a single client or multiple clients. The Media Stream Broadcast Distribution (MSBD) Protocol can be used to transmit the digitized audio and video of a live event to another computer that is running appropriate streaming media server software or to distribute the stream to multiple clients.	[MS-MSBD]
Microsoft Media Server (MMSP) Protocol	Used by the MSS System to stream data between the Windows Media Player (WMP) and Windows Media Server (WMS) over Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).	[MS-MMSP]
Windows Media HTTP Streaming Protocol (WMSP)	Used to transfer real-time multimedia data (for example, audio and video). The protocol depends on Hypertext Transfer Protocol (HTTP) for the transfer of all protocol messages, including the transfer of multimedia data.	[MS-WMSP]
Real-Time Streaming Protocol Windows Media Extensions (RTSP-WME)	Defines extensions to the Real-Time Streaming Protocol (RTSP), Real-Time Transport Protocol (RTP), Session Description Protocol (SDP), and Real-Time Transport Control Protocol (RTCP) to enable the delivery of multimedia data that is encapsulated in Advanced Systems Format (ASF) packets.	[MS-RTSP]
Windows Media HTTP Push Distribution Protocol (WMHTTP)	Used to transfer real-time multimedia data (for example, audio and video) from a client to a server. Push distribution can be used for broadcasting company meetings or live presentations. In such scenarios, the client is likely to be an encoder software application, perhaps implemented using the Windows Media Encoder software development kit (SDK).	[MS-WMHTTP]
Windows Media Log Data Structure (WMLOG)	The Windows Media Log Data Structure is a syntax for logging messages. The logging messages specify information about how a client received multimedia content from a streaming server. In addition to focusing on the streaming media protocols, it is important to identify some of the external system protocols that are needed to capture the scenarios.	[MS-WMLOG]

Microsoft Media Server Protocol (MMSP) as defined in [MS-MMSP], Windows Media HTTP Streaming Protocol (WMSP) as defined in [MS-WMSP], and Real-Time Streaming Protocol Windows Media Extensions (RTSP-WME) as defined in [MS-RTSP], serve similar roles as protocols. They enable unicast streaming of content to media player clients.<2>

There are differences, however. Windows Media HTTP Streaming Protocol and Real-Time Streaming Protocol (RTSP) Windows Media Extensions provide additional features beyond the deprecated MMSP Protocol.

This table shows some key differences in these protocols:

Protocol	Packet-Pair Bandwidth Estimation	Server-Side Playlist Seeking	Advanced Fast Start	Fast Start	Packet Transport
[MS-RTSP]	Yes	Yes	Yes	Yes	TCP and UDP
[MS-MMSP]	Yes	No	No	No	TCP and UDP
[MS-WMSP]	Yes	Yes	Yes	Yes	TCP only

5.3.1.1 RTSP-WME Overview

The RTSP Windows Media Extensions (RTSP-WME) are used to stream multimedia from a media server to a media player client or to another instance of a media server. To understand the extensions, it is important to understand the underlying protocol upon which the extensions are based.

RTSP, as specified in [RFC2326], is used to transfer real-time multimedia data (for example, audio and video) between a server and a client. RTSP is a streaming protocol; this means it attempts to facilitate scenarios in which the multimedia data is being transferred and rendered (that is, video displayed and audio played) simultaneously. RTSP establishes and controls either a single or several time-synchronized streams of continuous media. This protocol does not typically deliver the continuous streams itself, although interleaving of the continuous media stream with the control stream is possible. In other words, RTSP acts as a network remote control for multimedia servers.

RTSP typically uses a Transmission Control Protocol (TCP) connection for control of the streaming media session, although User Datagram Protocol (UDP) also can be used for this purpose.

The entity that sends the RTSP request that initiates the session is referred to as the client, and the entity that responds to that request is referred to as the server. Typically, the multimedia data flows from the server to the client. RTSP also allows multimedia data to flow in the opposite direction; however, the extensions defined in the RTSP Windows Media Extensions specification were not designed for such scenarios.

RTSP-WME does not use some features in the RTSP standard [RFC2326]. The following is a list of some of the features allowed by the RTSP standard that are not used by RTSP-WME:

- The use of UDP for controlling the streaming media session
- Streaming of multimedia data from a client to a server
- The RTSP RECORD command
- Queued RTSP PAUSE commands
- Time-delayed RTSP PAUSE commands
- The use of the multicast transport
- Multiple UDP connections to the same client for the same content
- Real-Time Transport Control Protocol (RTCP) Sender Reports and Receiver Reports
- The use of non-aggregate control

- The use of the REDIRECT method. Windows Media Services support the redirection response codes.

Clients use RTSP requests to control the session and to request the server to perform actions such as starting or stopping the flow of multimedia data. For each request, a corresponding RTSP response is sent in the opposite direction. Servers can also send RTSP requests to clients; for example, to inform them that session state has changed.

5.3.1.1.1 RTSP-WME: Logical Dependencies and Relationship to Other Protocols

The RTSP Windows Media Extensions (RTSP-WME) rely on TCP to control the streaming media session. RTSP uses the SDP syntax to describe the properties of content.

RTSP-WME is similar in functionality to the MMSP protocol (for more information, see [MS-MMSP]). However, RTSP-WME provides additional functionality that is not available in MMSP.

RTSP-WME is similar in functionality to the Windows Media HTTP Streaming Protocol (WMSP), as specified in [MS-WMSP]. However, in that protocol, the delivery of ASF packets is limited to TCP only.

RTSP-WME defines multiple extensions to the RTSP standard [RFC2326]. The extensions include the following:

- Extensions to the RTP payload format definitions to allow for ASF data packets, Forward Error Correction data, retransmitted packets, and Packet-Pair Data to be carried over RTP. For more information, see [MS-RTSP], sections 2.2.1, 2.2.2, and 2.2.3.
- Extensions to the RTCP protocol to allow the request for retransmission of lost RTP packets. For more information, see [MS-RTSP], section 2.2.4.
- Extensions to the Session Description Protocol to allow the inclusion of information contained in ASF file headers. For more information, see [MS-RTSP], section 2.2.5.
- New RTSP protocol headers to allow feature negotiation, caching, and faster than real-time streaming, among other things. For more information, see [MS-RTSP], section 2.2.6.
- Extensions to existing RTSP commands to enable the end of the stream to be indicated, to enable changing streams without interrupting streaming, and for initiating packet-pair measurements, among other things. For more information, see [MS-RTSP], section 2.2.7.

RTSP-WME is mutually exclusive from WMSP, MSB, MSBD, WMHTTP, and MMSP. When using RTSP-WME, you are not using the aforementioned protocols. You can, however, use WMLOG at the same time as RTSP-WME.

5.3.1.2 Microsoft Media Server Protocol (MMSP)

The Microsoft Media Server (MMSP) Protocol is used to transfer real-time multimedia data (for example, audio and video). Because it is a streaming protocol, the MMSP protocol attempts to facilitate scenarios in which the multimedia data is being transferred and rendered (such as video displayed and audio played) simultaneously.

The MMSP protocol is suitable for streaming delivery of real-time multimedia data. It is appropriate to use this protocol when a client is streaming from a server that does not support the RTSP Windows Media Extensions or the Windows Media HTTP Streaming Protocol. Otherwise, use of the MMSP protocol is generally not appropriate.

The MMSP protocol does not support seeking in a server-side playlist. That functionality is available in the RTSP Windows Media Extensions and in the Windows Media HTTP Streaming Protocol.

If the MMSP protocol is used in a scenario in which the multimedia data is always transferred over TCP, the Windows Media HTTP Streaming Protocol might be more appropriate to use instead because it does not include the UDP functionality.

The MMSP protocol uses a TCP connection for control of the streaming media session. The multimedia data flows from the server to the client.

The client can send MMSP protocol request messages to the server over the TCP connection to request the server to perform actions such as starting and stopping the flow of multimedia data. The multimedia data is transferred either over the same TCP connection or as a flow of UDP packets.

While the server is transmitting multimedia data to the client, the client can send MMSP protocol messages to the server, requesting that it change the stream being transmitted. For example, the client can request that the server replace the currently transmitted video stream with a lower bit-rate version of the same video stream.

If UDP is used to transmit the multimedia data to the client, the client can send an MMSP protocol message to the server requesting that it resend a UDP packet. This approach is useful if the client does not receive a UDP packet the server transmitted. Unlike other MMSP protocol messages sent by the client, the request to resend a UDP packet is sent using UDP.

Streaming media content can be made accessible only to authorized clients. The authentication sequences that the MMSP protocol supports are based on a basic and new technology local area network (NT LAN) Manager (NTLM) sequence. Authentication is not covered in detail as part of this task overview. See the Authentication Services overview for more information on authentication protocols.

5.3.1.2.1 MMSP Logical Dependencies and Relationship to Other Protocols

The MMSP protocol relies on TCP for the connection that controls the streaming media session. Both the client and the server send MMSP protocol messages over the TCP connection. The multimedia data that is being transferred by the server is sent over either TCP or UDP. The client also relies on UDP to send requests that ask the server to resend lost UDP packets.

The MMSP protocol is similar in functionality to the RTSP Windows Media Extensions (for more information, see [MS-RTSP]). However, the RTSP Windows Media Extensions support functionality that is not available in MMSP. See section 5.3.1.2 for a summary of functionality supported by MMSP.

The MMSP protocols are mutually exclusive from WMSP, MSB, MSBD, WMHTTP and RTSP-WME. When using MMSP, you are not using the aforementioned protocols. You can however use WMLOG at the same time as MMSP.

5.3.1.3 Media Stream Broadcast Protocol (MSB)

The Media Stream Broadcast Protocol (MSB) defined in [MS-MSB] allows the multicast distribution of ASF packets over a network for which IP multicasting is enabled. The MSB Protocol allows clients to tune in to a broadcast on a network, much like television and radio users can tune to a particular television or radio station.

To access a network broadcast, clients listen for Media Stream Broadcast packets on a particular IP address and UDP port. The specific IP multicast address and UDP port are delivered to clients by a Windows Media Station (.nsc) file. The .nsc file is delivered to the clients by some other means, such

as hosting the file at a URL for retrieval by means of HTTP, or sending the file as an email attachment.

5.3.1.3.1 MSB Relationship to Other Protocols

MSB Protocol packets are encapsulated in UDP. The UDP packets can be transmitted over either IP version 4 (IPv4) or IP version 6 (IPv6). The MSB Protocol packets are used to transport ASF packets. In addition, the MSB Protocol uses the forward error correction (FEC) algorithm, as specified in [RFC3452], for error detection.

The MSB protocol is mutually exclusive from WMSP, MMSP, MSBD, WMHTTP and RTSP-WME. When using MSB, you are not using the aforementioned protocols. You can, however, use WMLOG at the same time as MSB.

5.3.1.4 Media Stream Broadcast Distribution Protocol (MSBD)

The Media Stream Broadcast Distribution (MSBD) Protocol is used to transfer a stream of audiovisual content from a server to a single client or to multiple clients. For example, the MSBD Protocol might be used for 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 the Media Streaming Server System (MSS).

The MSBD Protocol also can be used for transmitting a stream from one MSS System server installation to another. Certain versions of WMP also support the MSBD Protocol, and can use it to monitor a live stream; they do this by connecting to a server running MSS, or directly to a real-time encoder.

5.3.1.4.1 MSBD Logical Dependencies and Relationship to Other Protocols

MSBD protocol packets are encapsulated in TCP. However, one MSBD protocol packet type can also be encapsulated in UDP. The UDP encapsulation mode is used only to transmit packets to an IPv4 multicast group.

The UDP encapsulation mode of this protocol might not be suitable for content that uses large Advanced Systems Format (ASF) data packets. Large ASF data packets might cause the UDP packets to be fragmented into multiple IP datagrams, and fragmentation of IP datagrams might be undesirable. In such cases, it is recommended to use the TCP encapsulation mode instead.

The MSBD Protocol is mutually exclusive from WMSP, MMSP, MSB, WMHTTP and RTSP-WME. When using MSBD, you are not using the aforementioned protocols. You can however use WMLOG at the same time as MSBD.

MSBD allows the server to specify that the ASF header of the content was obtained from an .nsc file. However, MSBD does not use .nsc files since the ASF header is transmitted using the MSBD protocol itself.

5.3.1.5 Windows Media HTTP Streaming Protocol (WMSP)

The Windows Media HTTP Streaming Protocol (WMSP) is used to transfer real-time multimedia data (that is, audio and video). This protocol is a streaming protocol, which means that it attempts to facilitate scenarios in which the multimedia data is being transferred and rendered (that is, video displayed and audio played) simultaneously.

This protocol depends on HTTP for the transfer of all protocol messages, including the transfer of the multimedia data. The multimedia data flows from the server to the client.

The client can send feedback to the server in the form of HTTP-based request messages. Based on the feedback, the server can, for example, replace the currently transmitted video stream with a lower bit rate version of the same video stream.

The Windows Media HTTP Streaming Protocol also supports pull distribution as specified in [MS-WMSP] section 4.11. Pull distribution is a method by which streaming content is transmitted from a source, such as Windows Media Encoder or Windows Media Services server, to a requesting server using the Windows Media HTTP Streaming Protocol. The connection to the encoder and transmission of the content is initiated and managed by the requesting server. When using pull distribution, the server acts as a logical client. Similarly, an encoder behaves as a logical server.

5.3.1.5.1 WMSP Logical Dependencies and Relationship to Other Protocols

The Windows Media HTTP Streaming Protocol (WMSP) depends on HTTP 1.0, as specified in [RFC1945]. The pipelined mode of the protocol can be used only if the client, the server, and any intermediate HTTP proxy servers support the pipelining feature of HTTP 1.1, as specified in [RFC2616].

This protocol can be used instead of the MMSP protocol, as specified in [MS-MMSP]. This protocol can also be used instead of RTSP-WME specified in [MS-RTSP]. However, it is important to note that although these two other protocols allow the multimedia data to be transmitted over either UDP or TCP, WMSP allows multimedia data to be transmitted only over TCP (because HTTP always uses TCP). WMSP is a good choice, where the multimedia data needs to pass through a proxy or a firewall as it uses HTTP, which is typically configured to pass through the proxy or firewall.

WMSP is mutually exclusive from WMHTTP, MMSP, MSB, MSBD and RTSP-WME. When using WMSP, you are not using the aforementioned protocols. You can however use WMLOG at the same time as WMSP.

5.3.1.6 Windows Media HTTP Push Distribution Protocol (WMHTTP)

The Windows Media HTTP Push Distribution Protocol (WMHTTP) is used to transfer real-time multimedia data (for example, audio and video) from a client to a server. Push distribution is ideal for broadcasting company meetings or live presentations. In such scenarios, the client is likely to be an encoder software application, perhaps implemented using the Windows Media Encoder Software Development Kit (SDK). For more information, see [WMESDK].

The Windows Media HTTP Push Distribution Protocol depends on HTTP for the transfer of all protocol messages, including the transfer of the multimedia data. With the Windows Media HTTP Push Distribution Protocol, multimedia data flows from the entity that initiates the HTTP connection to the entity that responds to the HTTP connection; the opposite of other streaming protocols such as the Windows Media HTTP Streaming Protocol, as specified in [MS-WMSP].

This protocol could be appropriate if a firewall prevents the server from initiating a TCP connection to the client, or if the client administrator needs to maintain control of the broadcast. The Windows Media HTTP Push Distribution Protocol supports HTTP access authentication, as specified in [RFC2616].

5.3.1.6.1 WMHTTP Logical Dependencies and Relationship to Other Protocols

The Windows Media HTPP Push Distribution Protocol (WMHTTP) depends on the Hypertext Transfer Protocol (HTTP/1.1), as specified in [RFC2616]. Either HTTP version 1.1 or HTTP version 1.0 can be used with this protocol.

This protocol also uses headers, packet types, and other components from the Windows Media HTTP Streaming Protocol, as specified in [MS-WMSP].

The WMHTTP Protocol is mutually exclusive from WMSP, MMSP, MSB, MSBD, and RTSP-WME. When using WMHTTP, you are not using the aforementioned protocols. You can however use WMLOG at the same time as WMHTTP.

5.3.2 Member Protocol Groups

While each protocol is mutually exclusive, they can be grouped by functionality as follows:

- Server to Server: Protocols that enable origin server to distribution server communication.
- Encoder to Server: Protocols that enable encoder to media server communication.
- Server to Client: Protocols that enable server to client communication.

Member protocol groups

Group	Protocol
Server to Server	Windows Media HTTP Streaming Protocol (WMSP) Windows Media HTTP Push Distribution Protocol (WMHTTP) Media Stream Broadcast Distribution (MSBD) Protocol Real-Time Streaming Protocol Windows Media Extensions (RTSP-WME) Media Stream Broadcast (MSB) Protocol
Encoder to Server	Windows Media HTTP Push Distribution Protocol (WMHTTP) Media Stream Broadcast Distribution (MSBD) Protocol Windows Media HTTP Streaming (WMSP) Protocol
Server to Client	Media Stream Broadcast (MSB) Protocol Media Stream Broadcast Distribution (MSBD) Protocol Microsoft Media Server (MMSP) Protocol Real-Time Streaming Protocol (RTSP-WME) Windows Media Extensions Media Stream Broadcast (MSB) Protocol Windows Media HTTP Streaming (WMSP) Protocol

Each individual protocol can be grouped with the WMLOG to provide statistics for the protocol streaming experience.

5.4 System Internal Architecture

The conceptual framework for the MSS System is defined in terms of three internal roles: client, server, and encoder, as described in section 4.3.1.

Roles that use the MSS System protocols:

- Player Component: A client application component that uses the MSS System protocols to request and stream files from the media server.
- Encoder Component: The encoder application component digitizes media and provides those media streams and files to the media server.
- Media Server: A server component that uses the MSS System protocols to receive and request data from the encoder application and to stream and provide requested streams to the player application.

Roles that use system external protocols:

- Player Application: An application that uses protocols other than the MSS System protocols to communicate with the DRM server to request licenses.
- Encoder Application: An application that uses protocols other than the MSS System protocols to communicate with the DRM packager and encrypt media files.

The high-level view of the integrated media-streaming protocols is depicted in the following figure.

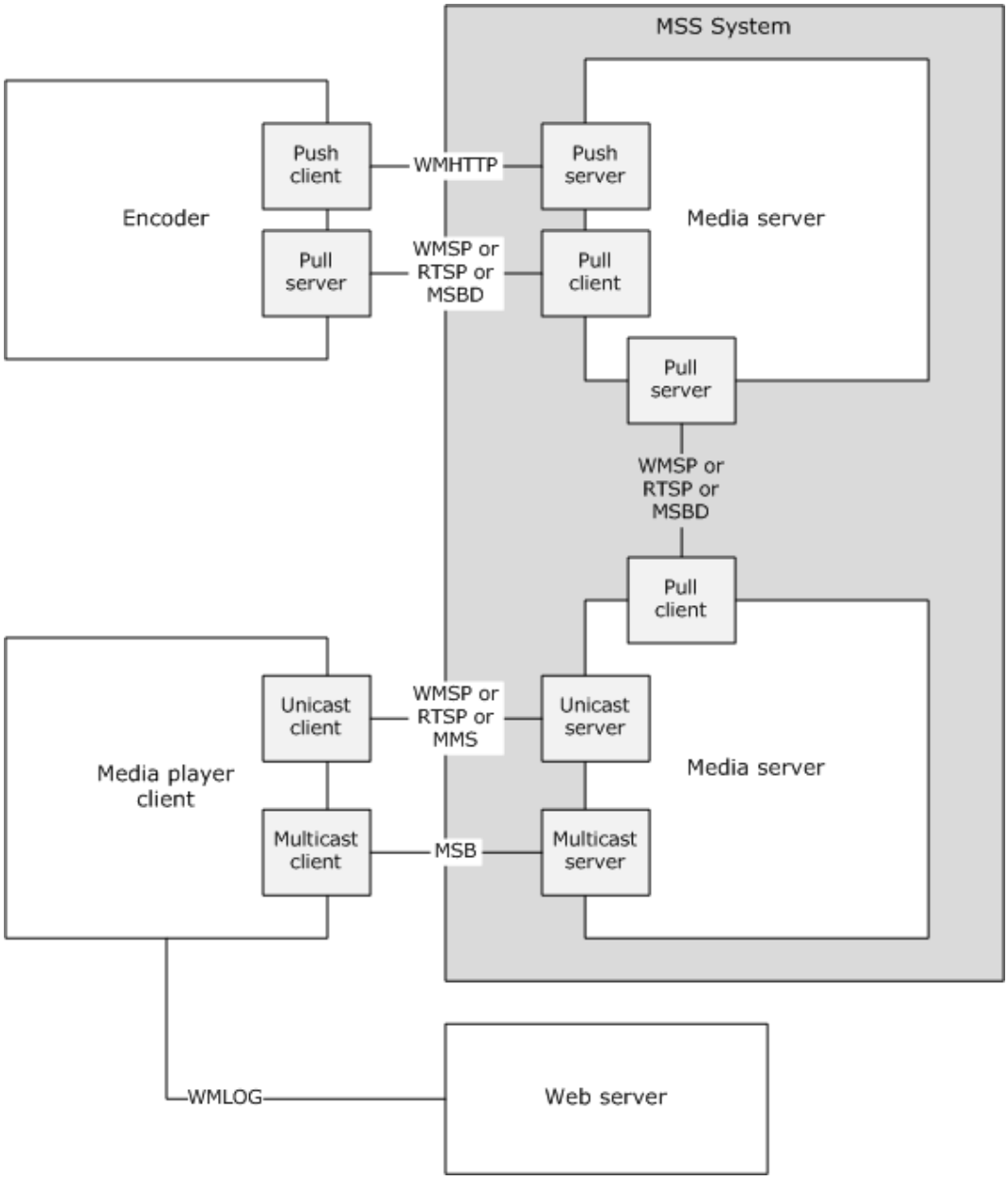

Figure 15: Integrated media streaming protocols

Encoder Push Client: The Push Encoder uses WMHTTP to push media streams to the media server.

Encoder Pull Server: The Pull Encoder uses WMSP or MSBD to stream digitized media streams to the media server. The decision on which protocol to use is based on the system platform in use.

Media Server as Origin Server: The Origin server plays two roles. It receives or pulls the content as provided by the encoders, and more importantly it streams the content to the client or distribution server. When streaming to a client, the Origin server uses MSB to stream multicast. When streaming unicast, the server will stream using one of three protocols: MMSP, WMSP, and RTSP-WME. The decision on which protocol to use is based on the system platform in use as specified in section 4.5. As part of the internal operation of these protocols, logging messages can be sent from the media player client to the media server.

Media Server as Distribution Server: The Distribution server plays two roles. It receives content from the origin server and therefore acts as a destination, but it also forwards the content on to the media player client. When streaming to the client, the Distribution server is limited to the same protocols that the origin server provides. The Origin server when distributing content to a distribution server, however, does so either through MSBD or WMSP. The decision on which protocol to use is based on the system platform in use as specified in section 4.5.

Media Server as Cache/Proxy Server: The media server can enable a built-in WMS Cache/Proxy capability.<3> As a Cache/Proxy server, it receives content from the origin server and caches the content for further distribution. It can also act as a reverse proxy server where the reverse proxy server appears to be the origin server to the client. If the content is broadcast content, the server may split stream the content.

Web Server: The web server can receive logging messages from the media player client.

5.5 Failure Scenarios

The system failures are defined in the specifications of the protocols supported by the system, as listed in section 2.2.

6 System Details

This section contains the details that complete the descriptions in earlier sections of the document. These details are needed to understand and implement this system.

6.1 Architectural Details

This section provides a series of examples illustrating the flow of communication between components of the MSS System. The examples are:

- Encoder Push to Media Server
- Media Server Pull from Encoder
- Multicast Playback
- Unicast Playback
- Packet-Pair Bandwidth Estimation
- Advanced Fast Start/Fast Start
- Logging
- Integrating DRM

6.1.1 Encoder Push to Media Server

As specified in section 3.3.4.1, Publish Content to Media Server - Encoder, getting content from the server and onto the media server is a key scenario. One way the MSS System enables the encoder to get content onto the media server is through a push. In the MSS System, the encoder push model is supported only by the WMHTTP protocol. The MSS System does not introduce additional functionality to the WMHTTP push scenarios. The WMHTTP sequence flow can be found in the WMHTTP specification. For general push sequences, see [MS-WMHTTP] section 4.1, General Push Distribution Sequence.

6.1.2 Media Server Pull from Encoder

In addition to the push model as specified in section 6.1.1, the MSS System also supports a pull model for getting content from an encoder. This is part of the use case in section 3.3.4.1.

While the push model only uses the WMHTTP protocol, pulling from an encoder is supported by the MSBD and WMSP protocols. The following diagram illustrates the communication flow between the encoder and the server during a pull mode.

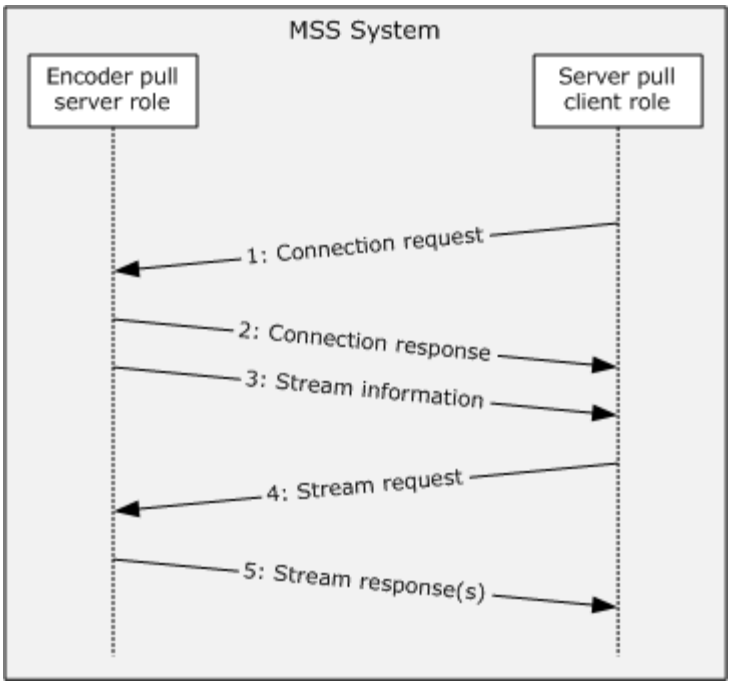

Figure 16: Media server pull from encoder communication flow

The encoder application publishes its URL: The WMSP and MSBD protocols do not provide a mechanism for a client to discover the URL to the server role. Thus, it is a prerequisite that the client role obtain a URL to the server role before this protocol can be used. For protocol prerequisites, see the individual member protocol [MS-MSBD] section 1.5.

Initialization: Prior to pulling content, the server acting as the client role has to initialize and establish a network connection to the encoder acting as the server role. For examples of the exact message content and format, see the individual member protocols [MS-MSBD] section 3.1.3 and [MS-WMSP] section 3.1.3.

1: Connection Request: The server acting as the client role MUST establish a TCP connection to the server by using the IP address and port number obtained from the URL provided by the encoder application. With the IP address known, the client can request to configure the encoder acting as the server role for streaming. For examples of the exact message content and format, see the individual member protocols [MS-MSBD] section 2.2.7 and [MS-WMSP] section 3.1.4.2.1.

2: Connection Response: The response to the request depends on the specific protocol. Although both protocols immediately send packets to the client, only WMSP expects a further request before it streams the rest of the file. For examples of the exact message content and format, see the individual member protocols [MS-MSBD] section 3.1.5.1 and [MS-WMSP] section 3.1.5.5.

3: Stream Information: Following the response, the server sends a packet that describes the media stream. For examples of the exact message content and format, see the individual member protocols [MS-MSBD] section 2.2.6 and [MS-WMSP] section 2.2.3.5.

4: Stream Request: Following a successful connection to the encoder, the next request the server, acting as the client role, sends to the encoder is to begin streaming. With MSBD, this was done as part of the initial MSB_MSG_REQ_CONNECT. For examples of the exact message content and format, see the individual member protocol [MS-WMSP] section 3.1.4.3.1.

5: Stream Response(s): After the encoder acting as the server role has sent its response, it immediately follows the response with a steady stream of data packets. (In order to simplify the diagram, individual data packets are not shown.) For examples of the exact message content and format, see the individual member protocols [MS-MSBD] section 3.2.5.2 and [MS-WMSP] section 3.1.5.11.

When the encoder has finished sending the packets and the capture is complete, the encoder notifies the server in the client role that the capture is complete. For examples of the exact message content and format, see the individual member protocols [MS-MSBD] section 3.1.5.3 and [MS-WMSP] section 3.1.5.13.

The server is now clear to close the connection.

6.1.3 Server to Client Streaming

As specified in section 3.3.4.3, a key scenario for the MSS System is playing back multimedia content. Playback can be done either through multicast or unicast. Which protocol to use depends on client and server negotiation, as well as the scenario being enabled by the administrator.

For multicast playback, see section 6.1.4.

For unicast playback, see section 6.1.5.

The Media server may act as a proxy server when providing a stream to the media player client. When acting like a proxy server, the server often caches the content so that addition client connections to the proxy server do not require an additional request to the origin server. For proxy server and distribution server communications see section 6.1.7.

If the content is broadcast content, then the content cannot be cached. In this case the proxy server may split stream the content. Split streaming allows the stream to be sent to additional clients. Before splitting the stream, the server must validate the stream type and that split stream is allowed.

For examples of the stream type directive, see [MS-RTSP] section 2.2.6.2.12 and [MS-WMSP] section 2.2.1.1.12.

For examples of the split stream permission directive, see [MS-RTSP] section 2.2.6.2.11 and [MS-WMSP] section 2.2.1.1.11.

6.1.4 Multicast Playback

In the MSS System, multicast is supported by the Media Stream Broadcast (MSB) and Media Stream Broadcast Distribution (MSBD) protocols. The MSB protocol is a server to client protocol. The architectural details of this protocol are not expanded by the MSS System. For complete information on the MSB protocol, please see [MS-MSB]. For the general sequence see [MS-MSB] section 4.1.

6.1.5 Unicast Playback

As mentioned in section 6.1.4, unicast streaming is a form of streaming for the MSS System. In the MSS System unicast playback is supported by the following protocols: WMSP, MMSP, and RTSP-WME. Unicast streaming is part of the use case in section 3.3.4.3.

The following diagram illustrates the communication flow between the media player application and the server during a play. In order to simplify the diagram, this sequence is limited to WMSP and RTSP over a TCP connection. In order to simplify the flow, the sequence diagram does not show

additional transport control functions, nor does it show all possible messages. For the sequence flows over a UDP connection, or the MMSP protocol, please see the individual member protocol TDs.

The process for streaming is broken down into three major areas:

- Discovery
- Stream selection
- Playback

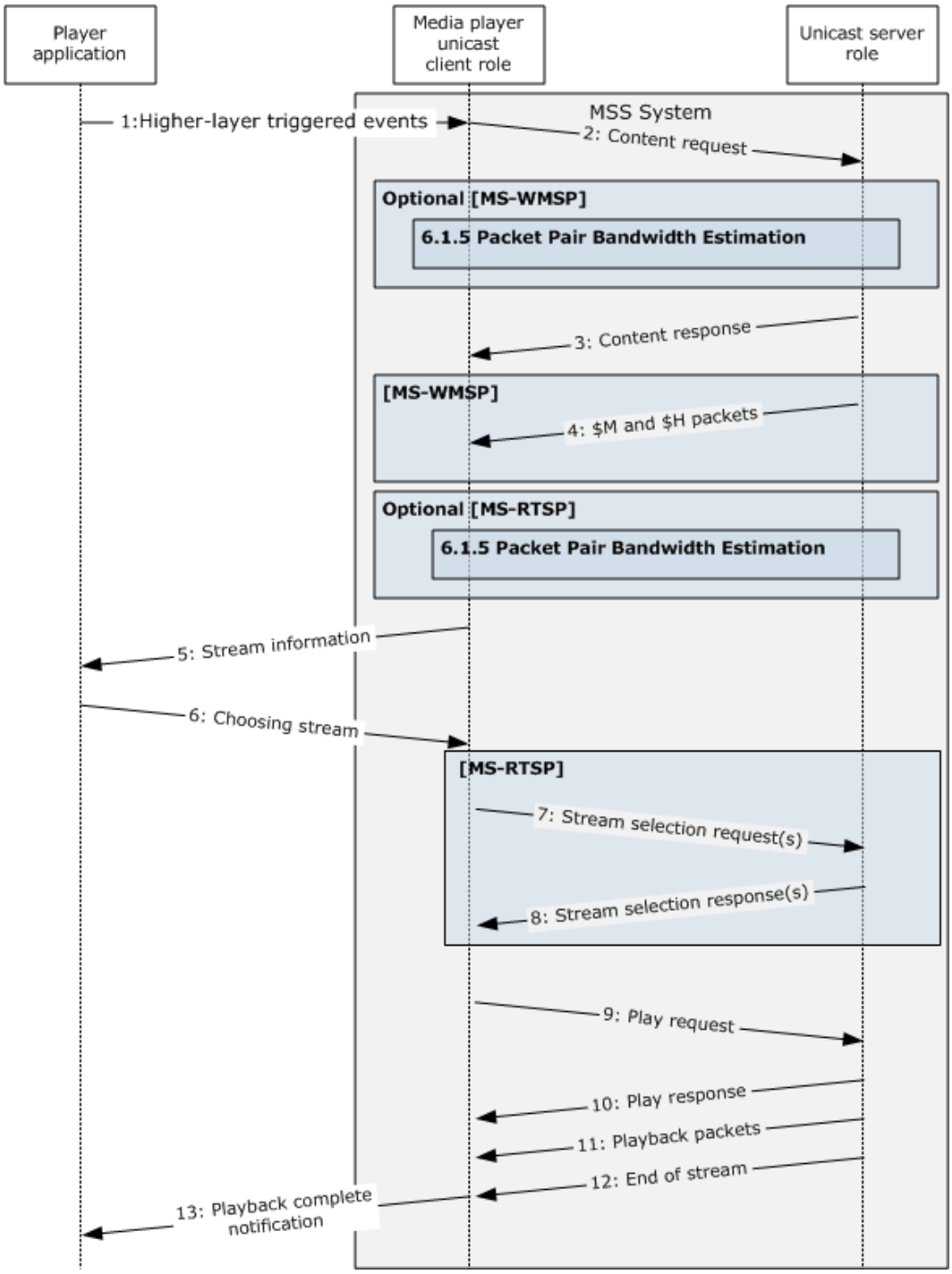

Figure 17: Unicast playback communication flow

1: Higher layer triggered events: The trigger for the unicast playback is the media player application providing the content URL to the media player unicast client role. The media player unicast client role uses the content URL to initiate the discovery.

For examples of the exact message content and format, see the individual member protocol TDs.

2: Content request: After receiving the request to stream from the media player application, the media player unicast client role will establish a TCP connection to the server by using the IP address and port number obtained from the URL provided by the media player application. With the IP address known, the client can request to configure the server for streaming. How this is accomplished is protocol-specific. For examples of the exact message content and format, see [MS-RTSP] section 3.1.4.2.1 and [MS-WMSP] section 3.1.4.2.1.

Optional WMSP: As part of the Describe request for [MS-WMSP] section 3.1.4.2.1, the client can request packet-pair bandwidth estimation. See section 6.1.6 for more details on packet-pair bandwidth estimation.

3: Content response: The media player unicast client role needs to obtain information on the stream itself. With RTSP-WME and WMSP, this data comes with the Describe response. For examples of the exact message content and format for the Describe response for RTSP-WME, see [MS-RTSP] section 3.1.5.4. The Describe response includes stream-specific information. For examples of the exact message content and format for WMSP, see [MS-WMSP] section 3.1.5.5. Note with WMSP this Describe response is the same Describe response described in section 6.1.6 step 1.

Optional RTSP-WME: Having received a response from the server, if the client is using the TCP transport, it may request packet-pair bandwidth estimation. See section 6.1.6 for more details

4: $M and $H packets: WMSP, following the packet-pair response or following the Describe request, will respond with $M and $H packets. These packets include stream-specific information. This information may be used by the higher layer to select a specific stream during the Play request in step 9. For examples of the exact message content and format, see [MS-WMSP] sections 3.1.5.7 and 3.1.5.8.

5: Stream information: RTSP-WME and WMSP receive the stream information with the original Describe response, as well as obtain the network performance data through the optional packet-pair bandwidth estimation (section 6.1.6). The file information provides details about the streams available. This data is provided back to the higher layer.

6: Choosing stream: After the file header information has been received in the request, the media player application can determine which stream to select for playback. The media player application will often choose the stream based on the packet-pair statistics it received during packet-pair testing. See section 6.1.6 for more details.

7: Stream selection request: To specify which stream to play, the media player unicast client role makes a request to the server for a specific stream. For examples of the exact message content and format on stream selection, see [MS-RTSP] section 3.1.4.3.1.

There may be more than one SelectStream request with RTSP-WME, since the setup method will be used for each selected stream. For more information see [MS-RTSP] section 2.2.7.10.

WMSP supports pipelined mode and non-pipelined mode. For non-pipelined mode, feedback from the client is sent to the server by using HTTP requests on TCP connections that are separate from the TCP connection that is used by the server to transfer the multimedia data to the client. This sequence diagram shows non-pipelined mode. For examples of the exact message content and format, of pipelined or non-pipelined mode, see [MS-WMSP].

8: Stream selection response: After receiving the stream request the server will send a response. For examples of the exact message content and format for RTSP-WME, see [MS-RTSP] section 3.2.5.6.

9: Play request: With RTSP-WME, once the stream or streams have been selected by the media player unicast client role, a request will be made to play the file. For examples of the exact message content and format forRTSP-WME, see [MS-RTSP] section 3.1.5.9.1.

With WMSP, as part of the play request, the media player unicast client role specifies which stream to play. For examples of the exact message content and format for WMSP, see [MS-WMSP] section 3.1.4.3.1.

10: Play response: The server responds to the Play request with a Play response. For examples of the exact message content and format for WMSP, see [MS-WMSP] section 3.1.5.11. For examples of the exact message content and format for RTSP-WME, see [MS-RTSP] section 3.1.5.10.

11: Playback packets: Following the Play requests, the server begins sending the packets to the client. For examples of the exact message content and format for WMSP, see [MS-WMSP] section 2.2.3. For examples of the exact message content and format for RTSP-WME, see [MS-RTSP] section 2.2.1.

12: End of stream: When the server has completed playing the file, it notifies the media player unicast client role that playback is complete. For examples of the exact message content and format for WMSP, see [MS-WMSP] section 3.2.4.1. For examples of the exact message content and format for RTSP-WME, see [MS-RTSP] section 3.2.4.1.

13: Playback complete notification: Following receiving the playback complete from the media server, the client-role sends a playback complete notification to the media player client application.

6.1.6 Packet-Pair Bandwidth Estimation

The following streaming media protocols support packet-pair bandwidth estimation in order to optimize playback: RTSP-WME, MMSP, and WMSP. Packet-pair is a technique for estimating the bandwidth of a streaming media connection over the Internet. To estimate bandwidth, the media server sends two or more consecutive messages, referred to as "$P packets", containing random data, and the client estimates the bandwidth by measuring the difference between the times that it receives the messages. Each MSS member protocol has different requirements for content and length of the packet-pair. For details on RTSP-WME packet-pair data, see [MS-RTSP] section 2.2.3.2. For details on WMSP packet-pair data, see [MS-WMSP] section 2.2.3.7. For details on MMSP packet-pair data, see [MS-MMSP] section 2.2.4.6.

The packet-pair operation consists of the following operations:

- Client connection to a server that supports packet-pair bandwidth estimation
- Client request for a packet pair
- A series of packets for the client to measure

Packet-pair bandwidth estimation is part of the use case in section 3.3.4.3.

A prerequisite requirement for packet-pair bandwidth estimation is that the client has already connected to the server and determined that the server supports packet-pair bandwidth estimation. For steps involved in connecting the client and server, see section 6.1.5.

The sequence of commands that is required to negotiate a packet-pair depends on whether the client and server have negotiated a UDP or a TCP connection for the transmission of streaming media packets and the specific protocol in use.

The following figure shows schematically the bandwidth operation and sequence for RTSP-WME and WMSP over a TCP connection. For the sequence over a UDP connection, or the MMSP protocol, see the individual member protocol TDs.

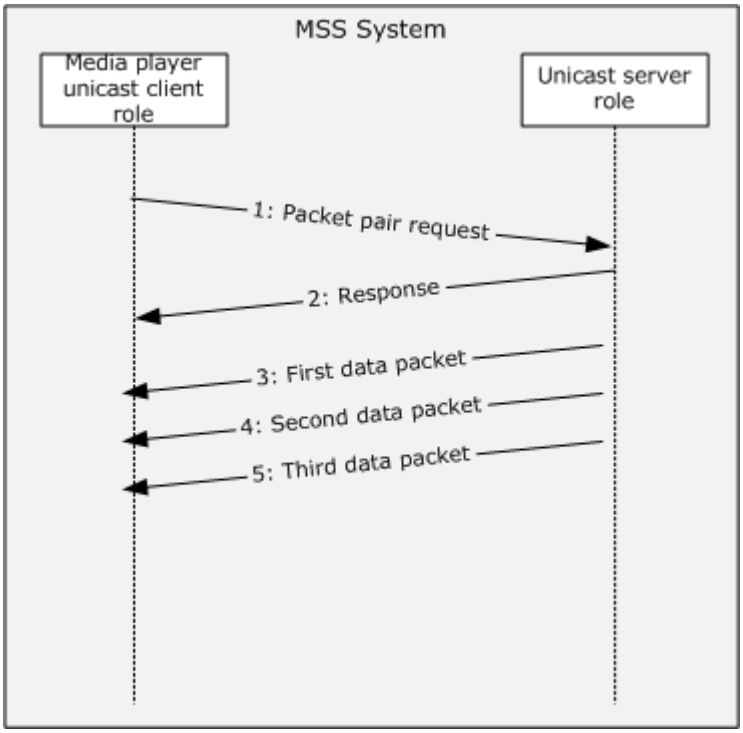

Figure 18: Packet-pair bandwidth estimation communication flow

1: Packet-pair request: Following the successful connection to a server that supports packet-pair bandwidth estimation, the media player client role can request a packet-pair bandwidth test from the media server. The requests are different depending on whether the transport negotiated was TCP or UDP. For examples of the exact message content and format for RTSP, see [MS-RTSP] sections 3.1.4.2.1 and 2.2.7.12. For examples of the exact message content and format for WMSP, see [MS-WMSP] sections 3.1.4.2.1, 2.2.1.4.14, and 2.2.1.7.3. Note that with WMSP this Describe Request is actually the same Describe Request described in section 6.1.5, step 2.

2: Packet-pair response: The server sends a "200 OK" response with three $P packets in the message body. For examples of the exact message content and format for RTSP-WME, see [MS-RTSP] section 3.1.5.5. For examples of the exact message content and format for WMSP, see [MS-WMSP] section 3.1.5.5.

3: First data packet: The server will send a packet containing packet-pair data to the client. For examples of the exact message content and format for RTSP-WME, see [MS-RTSP] section 2.2.3.2. For examples of the exact message content and format for WMSP, see [MS-WMSP] section 2.2.3.7.

4: Second data packet: The server will send a packet containing packet-pair data to the client. For examples of the exact message content and format for RTSP-WME, see [MS-RTSP] section 2.2.3.2. For examples of the exact message content and format for WMSP, see [MS-WMSP] section 2.2.3.7.

5: Third data packet: The server will send a packet containing packet-pair data to the client. For examples of the exact message content and format for RTSP-WME, see [MS-RTSP] section 2.2.3.2. For examples of the exact message content and format for WMSP, see [MS-WMSP] section 2.2.3.7.

6.1.7 Advanced Fast Start/Fast Start

Fast Start and Advanced Fast Start work together to optimize the playback experience for the End User. Fast Start and Advanced Fast Start are supported by WMSP and RTSP-WME protocols. Advanced Fast Start and Fast Start work together with the information provided by packet-pair bandwidth estimation as discussed in section 6.1.6.

Advanced Fast Start and Fast Start are part of the use case in section 3.3.4.3.

The following figure shows the communication between a media player client, an origin server, and a distribution server, taking into account the Advanced Fast Start and Fast Start headers.

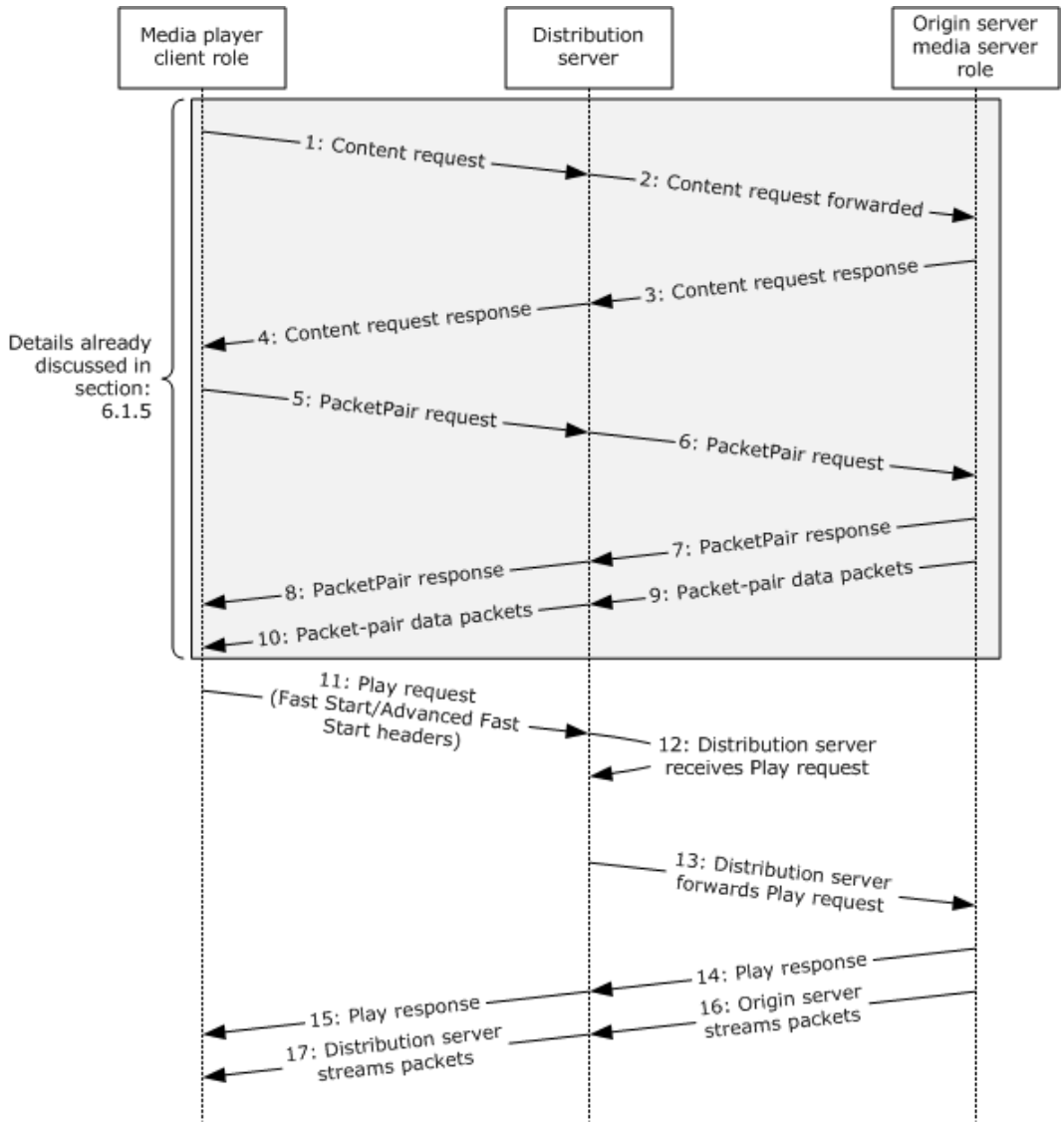

Figure 19: Advanced Fast Start/Fast Start communication flow

1: Content request: The trigger for Advanced Fast Start and Fast Start is the player application making a request to initiate playback, as seen in section 6.1.5. The media player client role receives

the URL for the content from the player application and makes a connection request to the distribution server.

2: Content request forwarded: The distribution server forwards the request to the origin server. The distribution server, when forwarding the request, is acting as the client role as seen in section 6.1.5. The distribution server is free to add and remove headers. For example, if WMSP is used, distribution servers add the Via header, as specified in [MS-WMSP] section 3.1.5.1. For specifics about additional headers, see the individual member protocol TDs.

3: Content request response: The origin server responds to the content request as seen in section 6.1.5.

4: Content request response forwarded: The distribution server acting as the server role forwards the response to the client role as seen in section 6.1.5.

5 through 10: Content request forwarded through packet-pair data packets: From the trigger moving forward, the sequence matches the sequence diagram already defined in section 6.1.6.

11: Play request (Fast Start and Advanced Fast Start headers): Following the successful processing of the packet-pair data as specified in section 6.1.6, the client role provides the data to the player application. The player application then decides which stream to play from the server and specifies the Fast Start and Advanced Fast Start headers in the media player client role play request. For examples of the exact Advanced Fast Start headers, see the individual member protocol TDs [MS-WMSP] sections 2.2.1.4.24 and 2.2.1.7.5, and [MS-RTSP] section 2.2.6.10.8. Also, see Speed header, as specified in [RFC2326] section 12.35.

For examples of the exact Fast Start headers, see the individual member protocol TDs [MS-WMSP] sections 2.2.1.4.1 and 2.2.1.4.2, and [MS-RTSP] section 2.2.6.13.

For examples of the exact play request, see the individual member protocol TDs [MS-WMSP] section 3.1.4.3.1 and [MS-RTSP] section 3.1.5.9.1.

12: Distribution server receives play request with Fast Start and Advanced Fast Start headers: When the distribution server receives the play request with the headers, it is acting as the server role to the media player client. The distribution server reads the Fast Start headers, and generates Fast Start headers (BurstBW and BurstDuration) from them. The value of the Fast Start headers set by the distribution server increases the preferred bandwidth and preferred duration from the values of Fast Start header of the play request. By doing this, the distribution server, when acting as the client role to the origin server, will request a higher bandwidth and duration than the media player client. This will ensure that the distribution server always has enough packets to support the media player client role. This is known as Fast Start, specified in section 3.1.11. For examples of the exact headers, see the individual member protocol TDs [MS-WMSP] sections 2.2.1.4.1, 2.2.1.4.2, 2.2.1.4.3, and 2.2.1.4.4, and [MS-RTSP] sections 2.2.6.13 and 2.2.6.17.

In addition the distribution server will forward the Advanced Fast Start headers as part of its request to the origin server.

For examples of the exact Advanced Fast Start headers, see the individual member protocol TDs [MS-WMSP] sections 2.2.1.4.24 and 2.2.1.7.5, and [MS-RTSP] section 2.2.6.10.8. Also, see Speed header, as specified in [RFC2326] section 12.35.

13: Distribution server forwards the play request with headers: The distribution server forwards the play request and client specified Fast Start headers as well as the newly calculated distribution server's Fast Start values to the origin server. The distribution server is not required to modify any headers received from the client, including the client's Fast Start and Advanced Fast Start headers, when it forwards the request to the origin server. The distribution server is playing the client role in this case. On receiving the play request the origin server begins to stream content at the requested

Fast Start speeds. For examples of the exact headers, see the individual member protocol TDs [MS-WMSP] section 3.2.5.6 and [MS-RTSP] section 3.2.5.8.

14: Origin server sends the play response: The origin server on receiving the request sends a play response. The origin server, when receiving the Advanced Fast Start headers, will respond in the play response with the X-StartupProfile. This header provides the information that the client needs for doing Advanced Fast Start. The information in this header is defined by the server based on the content to be streamed. For examples of the exact headers, see the individual member protocol TDs [MS-WMSP] section 2.2.1.12 and [MS-RTSP] section 2.2.6.28.

15: Distribution server sends the play response: The distribution server, acting as the client role, receives the response from the origin server. The distribution server, now acting as the server role to the media player client role, forwards the play response and headers. The distribution server is not required to modify any headers received from the origin server when it forwards the request to the client. For examples of the exact headers, see the individual member protocol TDs [MS-WMSP] section 2.2.1.12 and [MS-RTSP] section 2.2.6.28.

16: Origin server streams packets at Fast Start speeds: As specified in the TDs, after sending the play response, the origin server will begin to send packets. The packets will be sent according to the Fast Start headers that were provided by the distribution server client role. For examples of the exact headers, see the individual member protocol TDs [MS-WMSP] section 3.2.5.6 and [MS-RTSP] section 3.2.5.8.

17: Distribution server streams packets at Fast Start speeds and with minimal latency: As specified in the TDs, after sending the Play response, the distribution server will begin to send packets. The packets will be sent according to the Fast Start headers that were provided by the media player client role. In addition, the media player client role can begin playback with a minimal amount of buffering. The X-StartupProfile header, which the distribution server forwards unchanged from the origin server, communicates the minimum buffer the client needs before it can start streaming the file. For examples of the exact headers, see the individual member protocol TDs [MS-WMSP] section 3.2.5.6 and [MS-RTSP] section 3.2.5.8.

6.1.8 Logging

Logging is a process that allows the media player client to submit information and statistics to a media server or a Web server. The trigger for acquiring logging information depends on the specific log being submitted.

- **Connect-time log:** The trigger for the connect-time log is the start of the streaming **session**.
- **Legacy log:** The trigger for the legacy log is the end of the streaming session.
- **Rendering log:** The trigger for the rendering log is the media player application ending playback.
- **Streaming log:** The trigger for the streaming log is the end of the streaming session.

This is part of the use case specified in section 3.3.4.5.

The following figure shows the logging communication between a Media Player Application, the media player client role, a media server role and a Web server.

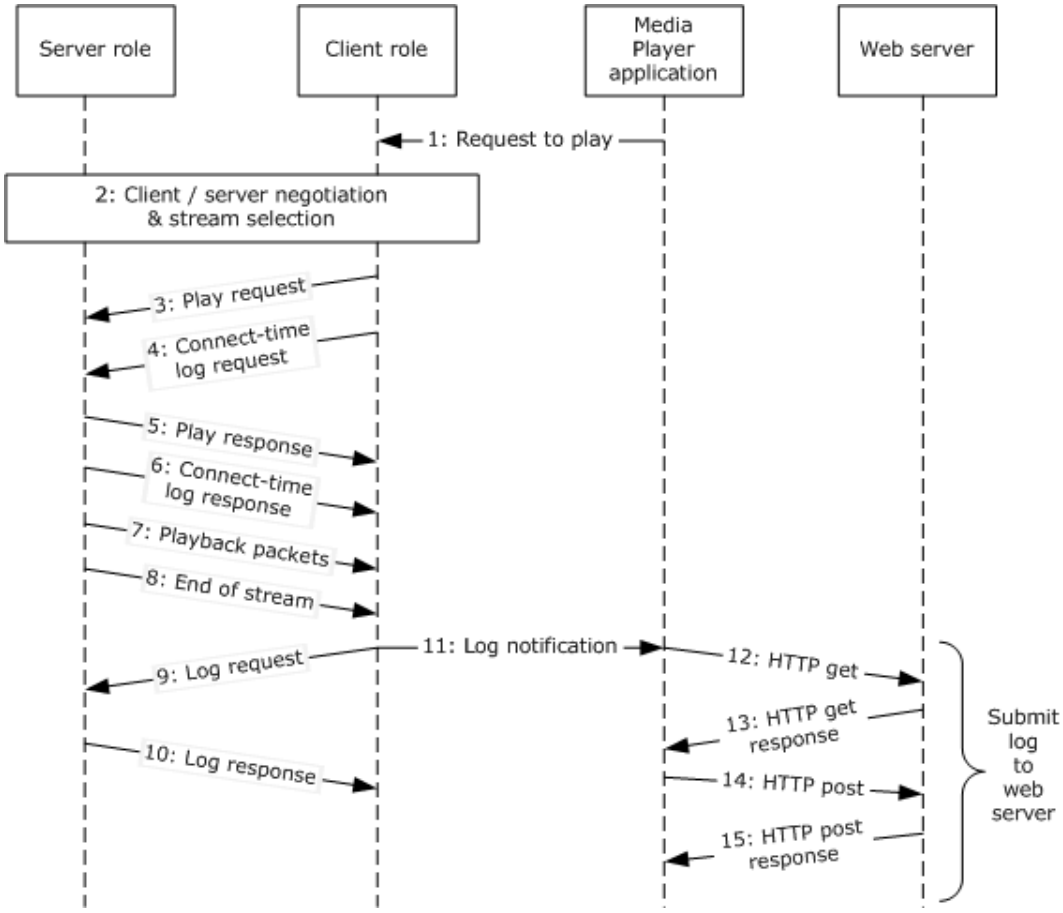

Figure 20: Logging communication flow

1: Request to play: The Media Player Application makes a request of the media player client role to play a specified URL.

2: Client/server negotiation and select a stream for playback: As documented in section 6.1.5, the media player client role and the media server role send messages in order to negotiate settings and specify the stream.

3: Play request: As documented in section 6.1.5, the media player client role makes a play request to the server.

4: Connect-time log request: After streaming has begun, the media player client role sends the connect-time log request to the media server role. The connect-time log is not supported by the web server.

5: Play response: The media server role responds to the play request as specified in section 6.1.5.

6: Connect-time log response: After receiving the connect-time log request from the media player client role, the media server role sends a response to the client.

7: Playback packets: As documented in section 6.1.5, the media server role begins to stream packets to the media player client role.

8: End of stream: When playback completes, the media server role sends the end of stream notification to the media player client role. The media player client role receives the notification and processes it. For examples of the exact headers, see the individual member protocol TDs [MS-WMSP] section 3.1.5.13, [MS-RTSP] section 3.1.5.13, [MS-MMSP] section 3.1.5.18, and [MS-MSB] section 3.2.5.4.

Upon receipt of the end of stream notification, the media player client role will notify the Media Player Application, and send a log request.

9: Log request: After receipt of the last packet, the media player client role sends a log request message to the media server role. For examples of the exact message content and format, see the individual member protocol TDs [MS-WMSP] section 3.2.5.11, [MS-RTSP] section 3.2.5.12, and [MS-MMSP] section 3.2.5.15.

10: Log response: The media player client role processes the log request response. For examples of the exact message content and format, see the individual member protocol TDs [MS-WMSP] section 3.1.5.15, [MS-RTSP] section 3.1.5.14, [MS-MMSP] section 3.2.5.15, and [MS-MSB] section 3.2.7.1.

11: Log notification: The media player client role notifies the Media Player Application. The Media Player Application determines how and whether to report the logging data.

12 through 15: Log response: The Media Player Application, after processing the log notification, submits the log to the Web server. For examples of the exact logging details, see the individual member protocol TD [MS-WMLOG] section 2.3.

6.1.9 Integrating DRM

DRM integration happens entirely outside the MSS System. This section is intended to provide an overview of the integration of DRM with the MSS System experience. This is part of the use case extensions in sections 3.3.4.4 and 3.3.4.2.

The following figure shows the logging communication between a media player application, the encoder application, the DRM packager, the DRM server, the MSS server, and the MSS client.

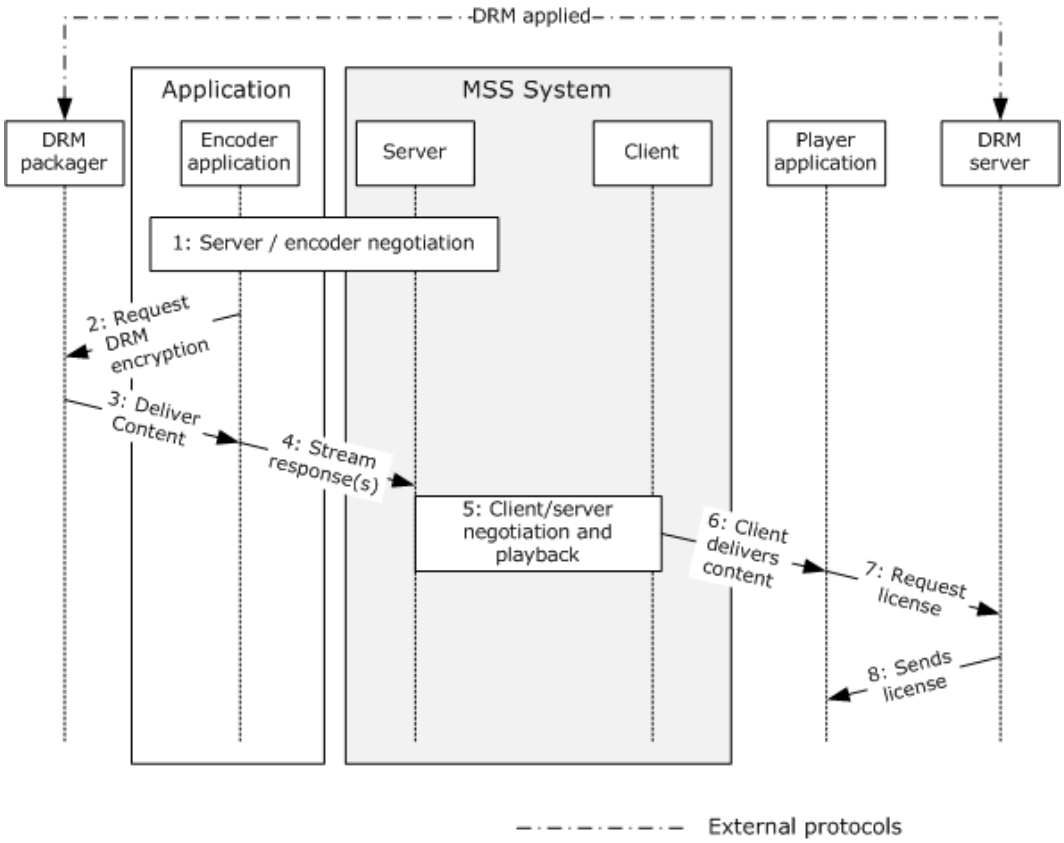

Figure 21: Integrating DRM communication flow

1: Server and encoder negotiate: The MSS server establishes a connection with the encoder for streaming, as shown in section 6.1.2.

2: Request DRM encryption: The encoder calls a [MS-DRM] application for encrypting the encoded stream.

3: DRM packager delivers encrypted content: The packaged media file has been encrypted and locked with a key. This key is stored in an encrypted license, which is distributed separately by the DRM server when requested by the player application. Other information is added to the media file, such as the URL to the DRM server where the license can be acquired. For information on the Digital Rights Management License Acquisition Data Structure, see [MS-DRM].

4: Encoder streams content: The media server pulls the stream from the encoder, as shown in section 6.1.2.

5: Client/server negotiation and playback: As documented in section 6.1.5, the media player client role and the media server role send messages in order to negotiate stream selection and stream the content.

6: Client delivers content: The MSS client streams the content to the player application for playback. The player application receives the ASF file header as part of the stream and determines that the content requires a license from the Content Encryption Object (as specified in [ASF] sections 3.14 and 3.15). The player application may continue to stream the content, but the player application will not be able to play the content until it has obtained a license to decrypt the content. The player

application obtains the URL where the license can be acquired, as mentioned in step 3, from the ASF file header of the file. For information on the Digital Rights Management License Acquisition Data Structure, see [MS-DRM].

7: Player application requests DRM license: Using the file URL obtained in step 6, the Player Application requests the license from the DRM server. The license acquisition process can be silent, or require user input. For example, the server may require a credit card transaction prior to issuing a license. Or the client may wish to obtain the user's consent prior to requesting a license from a server. For information on the Digital Rights Management License Acquisition Data Structure, see [MS-DRM].

8: DRM server sends license: After the player application has completed the steps for acquiring the license, the player application can decrypt the content for playback. For details on streaming, see section 6.1.5. For information on the Digital Rights Management License Acquisition Data Structure, see [MS-DRM].

6.2 Communication Details

The system does not define additional communication details beyond those described in the specifications of the protocols supported by the system.

6.3 Transport Requirements

The system does not define a transport beyond those described in the specifications of the protocols supported by the system, as listed in section 5.1.

6.4 Timers

There are no timer events beyond those specified in the member protocol documents.

6.5 Non-Timer Events

The system does not define any non-timer events beyond those specified in the underlying member protocol documents.

6.6 Initialization and Reinitialization Procedures

The system does not define any initialization or reinitialization procedures beyond those specified in the underlying member protocol documents.

6.7 Status and Error Returns

The system does not define any status and error returns beyond those specified in the underlying member protocol documents.

7 Security

This section documents system-wide security issues that are not otherwise described in the Technical Documents (TDs) for the Member Protocols. It does not duplicate what is already in the Member Protocol TDs unless there is some unique aspect that applies to the system as a whole.

The system does not introduce any security requirements beyond those described in the specifications of the protocols supported by the system, as listed in section 2.2.

8 Appendix A: Product Behavior

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

- Microsoft Windows® 2000 operating system
- Microsoft Windows® 2000 Server operating system
- Microsoft Windows® 2000 Advanced Server operating system
- Windows® XP operating system
- Windows Server® 2003 operating system
- Windows Server® 2003 R2 operating system
- Windows Vista® operating system
- Windows Server® 2008 operating system
- Windows Server® 2008 R2 operating system
- Windows® 7 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 3.3.4.1: UDP encapsulation mode is only used for transmitting packets to an IPv4 multicast group. The IP multicast is supported only by Windows Media Services implementations in Windows NT 4.0 and Windows 2000 Server.

<2> Section 5.3.1: The Microsoft Media Server Protocol (MMSP) as defined in [MS-MMSP] is a deprecated protocol and is supported by Windows Media Services on Windows NT 4.0, Windows 2000 Server, and Windows Server 2003.

<3> Section 5.4: This role is supported by Windows Server 2008 and Windows Server 2008 R2.

9 Change Tracking

This section identifies changes that were made to the [MS-MSSO] 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.

- 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.

10 Index

A

Abstract data model 38
Actors - supporting 21
Advanced Fast Start 17
Advanced Fast Start/Fast Start example 59
Applicability 36
Architecture
  details 51
  internal 48
  overview 38
Assumptions 32
Authentication 31

B

Black box relationships 33
Broadcast
  live 13
  on-demand 13

C

Capability negotiation 36
Change tracking 68
Client - media 13
Communication 65

D

Data model - abstract 38
Dependencies
  on MSS System 35
Digital Rights Management 12
Distribution server 14
DRM integration example 63

E

Encoder (section 3.1.2 12, section 4.1.3 32)
Encoder push to media server example 51
Environment
  authentication 31
  encoder 32
  Media Player 31
  overview 30
Error returns 65
Examples
  Advanced Fast Start/Fast Start 59
  encoder push to media server 51
  integrating DRM 63
  logging 61
  media server pull from encoder 51
  multicast playback 53
  overview 51
  packet-pair bandwidth estimation 57
  unicast playback 53

F

Failure scenarios 50
Fast Start 16
Fields - vendor-extensible 37
Foundation 12

G

Glossary 7

I

Informative references 9
Initialization 65
Introduction 7

L

List of member protocols 10
Live broadcast 13
Logging example 61
Logical dependencies
  Media Stream Broadcast Distribution Protocol 46
  Microsoft Media Server Protocol 45
  RTSP Windows Media Extensions 44
  Windows Media HTTP Push Distribution Protocol 47
  Windows Media HTTP Streaming Protocol 47

M

Media client 13
Media Player 31
Media server pull from encoder example 51
Member protocol functional relationships
  groups 48
  roles
    Media Stream Broadcast Distribution Protocol 46
    Media Stream Broadcast Protocol 45
    Microsoft Media Server Protocol 44
    overview 41
    RTSP Windows Media Extensions 43
    Windows Media HTTP Push Distribution Protocol 47
    Windows Media HTTP Streaming Protocol 46
Member protocols 10
Multicast playback example 53
Multicast streaming 19

N

Non-timer events 65
Normative references 8

O

On-demand broadcast 13
Origin server 14
Overview 10

P

Packet-pair bandwidth estimation 16
Packet-pair bandwidth estimation example 57
Packets
  Transmission Control Protocol 20
  User Datagram Protocol 19
Preconditions 32
Prerequisites
  background knowledge and system-specific concepts
    Advanced Fast Start 17
    Digital Rights Management 12
    distribution server 14
    encoder 12
    Fast Start 16
    live broadcast 13
    media client 13
    multicast streaming 19
    on-demand broadcast 13
    origin server 14
    overview 12
    packet-pair bandwidth estimation 16
    proxy server 15
    Transmission Control Protocol packets 20
    unicast streaming 18
    User Datagram Protocol packets 19
  overview 12
  system purposes 20
  system use cases 20
Product behavior 67
Proxy server 15

R

References
  informative 9
  normative 8
Reinitialization 65
Relationship to other protocols
  Media Stream Broadcast Distribution Protocol 46
  Media Stream Broadcast Protocol 46
  Microsoft Media Server Protocol 45
  RTSP Windows Media Extensions 44
  Windows Media HTTP Push Distribution Protocol 47
  Windows Media HTTP Streaming Protocol 47
Relationships
  black box 33
  member protocol
    groups 48
    roles 41
  overview 33
  system dependencies 35
  white box 38
Required information 12
Returns - status and error 65

S

Security 66
Server
  distribution 14
  origin 14
  proxy 15
Stakeholders 20
Standards assignments 11
Status returns 65
Streaming
  multicast 19
  unicast 18
Supporting actors 21
System details 51
System influences 36
System purposes 20
System summary 10
System-environment relationship 30

T

Timers 65
Tracking changes 68
Transmission Control Protocol packets 20
Transport requirements 65

U

Unicast playback example 53
Unicast streaming 18
Use cases
  descriptions
    logging statistics to servers (Media Player) 27
    publishing content to media server (encoder) 23
    publishing secure content to media server (DRM Packager) 25
    requesting license from license server (Media Player) 26
    streaming content from media server (Media Player) 25
  diagrams 21
  stakeholders 20
User Datagram Protocol packets 19

V

Vendor-extensible fields 37
Versioning 36

W

White box relationships
  encoder pull model 39
  encoder push model 38
  multicast playback 41
  overview 38
  unicast playback 40