

[MS-WMHTTP]: Windows Media HTTP Push Distribution Protocol Specification

Intellectual Property Rights Notice for Protocol Documentation

- This protocol 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 protocols, and may distribute portions of it in your implementations of the protocols or your documentation as necessary to properly document the implementation. This permission also applies to any documents that are referenced in the protocol documentation.
- Microsoft does not claim any trade secret rights in this documentation.
- Microsoft has patents that may cover your implementations of the protocols. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. If you are interested in obtaining a patent license, please contact protocol@microsoft.com.
- 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.
- All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

This protocol documentation is intended for use in conjunction with publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

A protocol specification does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them.

Revision Summary

Date	Revision History	Revision Class	Comments
04/03/2007	0.01		MCPP Milestone Longhorn Initial Availability
07/03/2007	1.0	Major	MLonghorn+90
07/20/2007	1.0.1	Editorial	Revised and edited the technical content.
08/10/2007	1.1	Minor	Updated the technical content.

Date	Revision History	Revision Class	Comments
09/28/2007	1.1.1	Editorial	Revised and edited the technical content.
10/23/2007	1.1.2	Editorial	Revised and edited the technical content.
11/30/2007	1.1.3	Editorial	Revised and edited the technical content.
01/25/2008	1.1.4	Editorial	Revised and edited the technical content.

Table of Contents

1	Introduction	5
1.1	Glossary	5
1.2	References	5
1.2.1	Normative References	5
1.2.2	Informative References.....	6
1.3	Protocol Overview (Synopsis).....	6
1.4	Relationship to Other Protocols.....	7
1.5	Prerequisites/Preconditions	7
1.6	Applicability Statement	7
1.7	Versioning and Capability Negotiation.....	7
1.8	Vendor-Extensible Fields	8
1.9	Standards Assignments.....	8
2	Messages	9
2.1	Transport	9
2.2	Message Syntax	9
2.2.1	HTTP Header Fields	9
2.2.1.1	Content-Type.....	9
2.2.1.1.1	application/x-wms-pushsetup	10
2.2.1.1.2	application/x-wms-pushstart	10
2.2.1.2	Cookie	10
2.2.1.2.1	push-id	10
2.2.1.3	Pragma.....	10
2.2.1.3.1	no-cache	11
2.2.1.3.2	timeout	11
2.2.1.4	Server	11
2.2.1.5	Set-Cookie	11
2.2.1.6	Supported	11
2.2.1.7	User-Agent.....	11
2.2.1.8	X-Accept-Authentication	12
2.2.2	Request Types.....	12
2.2.2.1	PushSetup Request.....	12
2.2.2.1.1	Template-URL	14
2.2.2.1.2	AutoDestroy	14
2.2.2.2	PushStart Request.....	15
2.2.3	Packet Types.....	16
2.2.3.1	Common Definitions	16
2.2.3.1.1	Framing Header.....	16
2.2.3.2	\$C (Stream Change Notification)	17
2.2.3.3	\$D (Data)	17
2.2.3.4	\$E (End-of-Stream Notification).....	17
2.2.3.5	\$F (Filler).....	18
2.2.3.6	\$H (Header)	18
3	Protocol Details	19
3.1	Client Details	19
3.1.1	Abstract Data Model	19
3.1.2	Timers	19
3.1.3	Initialization	19
3.1.4	Higher-Layer Triggered Events.....	19
3.1.4.1	Request to Configure the Server.....	19
3.1.4.1.1	Sending the PushSetup Request.....	20

3.1.4.2	Request to Start Streaming Content.....	20
3.1.4.2.1	Sending the PushStart Request.....	21
3.1.4.3	ASF Packet Is Available to Send	22
3.1.4.4	Notification of the Last Packet	22
3.1.4.5	Notification of New ASF Header File.....	23
3.1.5	Message Processing Events and Sequencing Rules	24
3.1.5.1	Receiving a PushSetup Response.....	24
3.1.5.2	Receiving a PushStart Response	25
3.1.6	Timer Events.....	25
3.1.7	Other Local Events	26
3.1.7.1	TCP Connection Is Disconnected	26
3.2	Server Details.....	26
3.2.1	Abstract Data Model	26
3.2.2	Timers	26
3.2.3	Initialization.....	26
3.2.4	Higher-Layer Triggered Events.....	27
3.2.4.1	Administrative Disconnect	27
3.2.5	Message Processing Events and Sequencing Rules	27
3.2.5.1	Receiving a PushSetup Request.....	27
3.2.5.2	Receiving a PushStart Request	29
3.2.5.2.1	Sending a PushStart Response.....	30
3.2.5.3	Receiving a \$H Packet.....	30
3.2.5.4	Receiving a \$D Packet.....	30
3.2.5.5	Receiving a \$E Packet.....	31
3.2.5.6	Receiving a \$C Packet.....	31
3.2.5.7	Receiving a \$F Packet	31
3.2.6	Timer Events.....	32
3.2.6.1	Idle-Timeout Timer Expires.....	32
3.2.6.2	Inactivity-Timeout Timer Expires	32
3.2.7	Other Local Events	32
3.2.7.1	TCP Connection Is Disconnected	32
4	Protocol Examples	33
4.1	General Push Distribution Sequence	33
4.2	Push Distribution with AutoDestroy	35
4.3	Message Exchange During Push Distribution.....	36
5	Security	39
5.1	Security Considerations for Implementers.....	39
5.2	Index of Security Parameters	39
6	Appendix A: Windows Behavior	40
7	Index.....	41

1 Introduction

The Windows Media HTTP Push Distribution Protocol is an HTTP-based (as specified in [RFC2616](#)) Microsoft proprietary protocol used for transferring real-time multimedia data (for example, audio and video) from a client to a server. The client of the Windows Media HTTP Push Distribution Protocol is likely to be an encoder software application, perhaps implemented by using the Windows Media Encoder SDK. For more information, see [WMESDK](#).

1.1 Glossary

The following terms are defined in [\[MS-GLOS\]](#):

Advanced Systems Format (ASF)
Little-Endian

The following terms are defined in [\[MS-WMSP\]](#):

Content
Playlist
Session
Stream

The following terms are specific to this document:

Encode (or Encoding): The application of a compression/decompression (codec) algorithm and **stream** format to analog or digital multimedia data and subsequent reproduction as a digital file or **stream**.

Encoder: A device that uses software and/or hardware to **encode content**.

Push (or Push Distribution): A method by which a client initiates and manages the transmission of **content** to a server.

Streaming: The act of transferring **content** from a sender to a receiver. The data is transmitted at a fixed rate or at a rate that is related to the rate at which the data will be consumed (for example, displayed) by the receiver.

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

1.2 References

1.2.1 Normative References

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

[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-DTYP] Microsoft Corporation, "[Windows Data Types](#)", January 2007.
- [MS-GLOS] Microsoft Corporation, "[Windows Protocols Master Glossary](#)", March 2007.
- [MS-NLMP] Microsoft Corporation, "[NT LAN Manager \(NTLM\) Authentication Protocol Specification](#)", June 2007.
- [MS-WMSP] Microsoft Corporation, "[Windows Media HTTP Streaming Protocol Specification](#)", March 2007.
- [RFC2109] Kristol, D., and Montulli, L., "HTTP State Management Mechanism", RFC 2109, February 1997, <http://www.ietf.org/rfc/rfc2109.txt>
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>
- [RFC2616] Fielding, R., 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., Lawrence, S., Leach, P., Luotonen, A., and Stewart, L., "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999, <http://www.ietf.org/rfc/rfc2617.txt>
- [RFC3629] Yergeau, F., "UTF-8, A Transformation Format of ISO 10646", RFC 3629, November 2003, <http://www.ietf.org/rfc/rfc3629.txt>
- [RFC3986] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005, <http://www.ietf.org/rfc/rfc3986.txt>
- [RFC4234] Crocker, D., Ed. and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005, <http://www.ietf.org/rfc/rfc4234.txt>
- [RFC4559] Jaganathan, K., Zhu, L., and Brezak, J., "SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows", RFC 4559, June 2006, <http://www.ietf.org/rfc/rfc4559.txt>

1.2.2 Informative References

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

1.3 Protocol Overview (Synopsis)

The Windows Media HTTP Push Distribution Protocol is used for transferring 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 by using the Windows Media Encoder SDK. For more information, see [\[WMESDK\]](#).

The protocol depends on HTTP for the transfer of all protocol messages, including the transfer of the multimedia data. In this specification, the entity that initiates the HTTP connection is referred to as the client, and the entity that responds to the HTTP connection is referred to as the server. With the Windows Media HTTP Push Distribution Protocol, multimedia data flows from the client to the server—the opposite of other **streaming** protocols, such as the [Windows Media HTTP Streaming Protocol](#) as specified in [MS-WMSP].

For the purposes of this specification, the terms client and encoder have the same meaning and are used interchangeably.

1.4 Relationship to Other Protocols

This protocol depends on the Hypertext Transfer Protocol as specified in [\[RFC2616\]](#).

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

1.5 Prerequisites/Preconditions

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

1.6 Applicability Statement

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

This protocol may also 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.

If none of the above applies, it may be more appropriate to use the [Windows Media HTTP Streaming Protocol](#) (as specified in [MS-WMSP]) to transfer the data from the client to the server, instead of using this protocol.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas.

- **Supported Transports:** This protocol can be implemented on top of HTTP, as specified in section [2.1](#).
- **Protocol Versions:** Clients specify the protocol version by using the [User-Agent \(section 2.2.1.7\)](#) header. Servers specify the protocol version by using the [Server \(section 2.2.1.4\)](#) header.
- **Security and Authentication Methods:** This protocol supports the HTTP access authentication, as specified in [\[RFC2616\]](#) section 11.
- **Localization:** This specification does not specify any localization-dependent protocol behavior.
- **Capability Negotiation:** This protocol does explicit capability negotiation by using the [X-Accept-Authentication \(section 2.2.1.8\)](#) header.

This protocol does not use operating system versioning. The reason it does not is because operating systems typically include multiple client implementations with different capabilities. Furthermore, the client software components are frequently updated independently of the rest of the operating system. Instead, the protocol versioning mechanism relies on the version number of the software product that is sending the request or the response to be stated on the User-Agent (section 2.2.1.7) and Server (section 2.2.1.4) headers, respectively.

1.8 Vendor-Extensible Fields

This protocol uses HRESULTs as specified in [\[MS-DTYP\]](#). Vendors are free to choose their own values, as long as the C bit (0x20000000) is set, indicating that it is a customer code.

1.9 Standards Assignments

None.

2 Messages

The following sections specify how Windows Media HTTP Push Distribution Protocol messages are transported and the message syntax.

2.1 Transport

The Windows Media HTTP Push Distribution Protocol uses HTTP, as specified in [\[RFC2616\]](#), as the transport layer.

A TCP port has not been reserved for this protocol. TCP port 80 is commonly used because many HTTP proxy servers forward only HTTP traffic that uses port 80.

The protocol uses the access authentication functionality of the HTTP layer. The supported HTTP access authentication schemes are implementation-specific. Clients SHOULD use the [X-Accept-Authentication \(section 2.2.1.8\)](#) header to specify the preferred list of authentication schemes. Details about HTTP access authentication are as specified in [\[RFC2616\]](#) section 11.

2.2 Message Syntax

This section includes the following:

- [HTTP Header Fields \(section 2.2.1\)](#) specifies the syntax for HTTP headers defined by this protocol.
- [Request Types \(section 2.2.2\)](#) specifies the types of requests that are defined by this protocol and how each request type is mapped to HTTP.
- [Packet Types \(section 2.2.3\)](#) specifies the syntax for the binary packet format that is used in the payloads of the HTTP messages.

2.2.1 HTTP Header Fields

The Windows Media HTTP Push Distribution Protocol utilizes existing headers as specified in [\[RFC2616\]](#) and [\[MS-WMSP\]](#). Some headers defined by these specifications are further restrained by the Windows Media HTTP Push Distribution Protocol in how they can be used. These additionally restrained headers are defined in this section.

Unless specified otherwise, the headers defined in this specification and any tokens (also called tags or directives) used on those headers are defined for use in both requests and responses.

If a client or server receives an HTTP header that is not defined in this section, or if the header is not defined in the current context (for example, receiving a request-only header in a response), the header MUST be interpreted as specified in [\[RFC2616\]](#).

If a client or server receives an HTTP header defined in this section, and the header contains an unknown token or the token is not defined in the current context (for example, receiving a request-only token in a response), the token MUST be ignored.

This section defines the syntax of the HTTP headers that use the Augmented Backus-Naur Form (ABNF) syntax, as specified in [\[RFC4234\]](#). Any ABNF syntax rules that are not specified in [\[RFC4234\]](#) use the ABNF extensions that are as specified in [\[RFC2616\]](#) or [\[MS-WMSP\]](#).

2.2.1.1 Content-Type

The Content-Type header specifies the type of data that is included in the message payload (that is, the message body of a POST request).

The syntax of the Content-Type header is defined as follows:

```
Ctype          = "application/x-wms-pushsetup" /  
                  "application/x-wms-pushstart"  
  
Content-Type   = "Content-Type: " Ctype [";charset=UTF-8"] CRLF
```

Example: Content-Type: application/x-wms-pushsetup; charset=UTF-8

2.2.1.1.1 application/x-wms-pushsetup

This content-type is used in a POST request to initiate the push distribution **session** with a server. For more details, see the definition of the [PushSetup request \(section 2.2.2.1\)](#).

2.2.1.1.2 application/x-wms-pushstart

This content-type specifies that the message body of the POST request contains packet types as specified in section [2.2.3](#). For more details, see the definition of the [PushStart request \(section 2.2.2.2\)](#).

2.2.1.2 Cookie

The syntax of the Cookie header MUST conform to the format as specified in [\[RFC2109\]](#).

This header is defined for use in requests sent to the server. The Cookie header MUST be specified with a [push-id \(section 2.2.1.2.1\)](#).

The syntax of the Cookie header is defined as follows:

```
Cookie          = "Cookie: "  
                  push-id          ; section 2.2.1.2.1  
                  CRLF
```

Example: Cookie: push-id=0

2.2.1.2.1 push-id

The value of this cookie, which consists of an array of characters, identifies the streaming session. The session identifier is assigned by the server in the response to the [PushSetup request](#). The identifier "0" indicates that the client requests the server to create a new session.

The syntax of the push-id cookie is defined as follows:

```
session-id      = 1*VCHAR          ; any combination of characters except 0  
push-id         = "push-id="       ( "0" / session-id )
```

2.2.1.3 Pragma

The Windows Media HTTP Push Distribution Protocol uses the HTTP Pragma header field to communicate information specific to the operation of the protocol. The Pragma header consists of one or more comma-separated tokens, as specified in [\[RFC2616\]](#) section 14.32.

Details about the handling of error conditions related to Pragma header tokens are as specified in [\[MS-WMSP\]](#) section 2.2.1.4.

The Pragma header tokens used by the Windows Media HTTP Push Distribution Protocol are defined in the following two sections.

2.2.1.3.1 no-cache

Details about the no-cache token are as specified in [\[MS-WMSP\]](#) section 2.2.1.4.12.

2.2.1.3.2 timeout

Details about the timeout token are as specified in [\[MS-WMSP\]](#) section 2.2.1.4.29.

2.2.1.4 Server

The Server header specifies the major and minor version numbers of the software product that is responding to the HTTP request.

This header is defined only for use in responses sent to a client.

Details about the Server header are as specified in [\[MS-WMSP\]](#) section 2.2.1.5.

2.2.1.5 Set-Cookie

The syntax of the Set-Cookie header MUST conform to the format as specified in [\[RFC2109\]](#).

This header is defined for use in responses sent to a client. The Set-Cookie header MUST be specified with a [push-id \(section 2.2.1.2.1\)](#).

The syntax of the Set-Cookie header is defined as follows:

```
Set-Cookie      = "Set-Cookie: "  
                  push-id      ; section 2.2.1.2.1  
                  CRLF
```

Example: Set-Cookie: push-id=1234567890

2.2.1.6 Supported

The Supported header is used for specifying features of the protocol that are supported and that can be used in the current session.

This header is defined only for use in responses sent to a client.

Details about the Supported header are as specified in [\[MS-WMSP\]](#) section 2.2.1.7.

2.2.1.7 User-Agent

The User-Agent header specifies the major and minor version number of the software product that is sending the HTTP request.

This header is defined only for use in requests sent to a server.

The syntax of the User-Agent header is defined as follows:

```

major = 1*2DIGIT
minor = 1*2DIGIT [ "." 1*4DIGIT "." 1*4DIGIT ]
product = ; as defined in section 3.8 of [RFC2616]
user-agent-data = "WMEncoder" "/" major "." minor
                  *( SP product )
User-Agent= "User-Agent: " user-agent-data CRLF

```

Clients MUST assign the values of the major and minor ABNF syntax elements as follows:

Product name	Major	Minor
Windows Media Format 9 Series SDK	9	0
Windows Media Format 9.5 SDK	10	0
Windows Vista	11	0

2.2.1.8 X-Accept-Authentication

The X-Accept-Authentication header specifies the authentication schemes that the client supports.

Details about the X-Accept-Authentication header are as specified in [\[MS-WMSP\]](#) section 2.2.1.9.

2.2.2 Request Types

The Windows Media HTTP Push Distribution Protocol defines requests that a client can send to a server.

The requests from the client and the corresponding responses from the server are exchanged using HTTP request methods. Each request defined by the Windows Media HTTP Push Distribution Protocol is mapped to the HTTP POST request method.

This section defines the syntax of those requests that use ABNF syntax as specified in [\[RFC4234\]](#). Any ABNF syntax rules that are not specified in [\[RFC4234\]](#) use the ABNF extensions that are specified in [\[RFC2616\]](#) or [\[MS-WMSP\]](#).

In addition to complying with the ABNF syntax, all requests MUST also include a request line, all of the required headers, and one or more [Pragma](#) headers with the required Pragma header tokens. In the ABNF syntax for each request type, these components are indicated by the inclusion of "-Line", "-Header-REQ", and "-Token-REQ" in the associated ABNF rule name, respectively.

The following are some common constructions used throughout this section.

```

HTTP-Header-Types      = *( ( general-header
                               / request-header
                               / entity-header ) CRLF )

```

2.2.2.1 PushSetup Request

The purpose of the PushSetup request is to request permission to start streaming ASF data packets to the server.

The URL specified by the client in the HTTP request line of the POST request identifies the "publishing point". The concept of a publishing point is similar to that of a broadcast channel. A

server might have multiple publishing points, and different publishing points can have different access permissions associated with them. For example, some publishing points may require HTTP access authentication (as specified in [\[RFC2616\]](#) section 11). As another example, publishing points may also allow only clients that connect from a specific IP address.

The Windows Media HTTP Push Distribution Protocol allows for publishing points to be created by the PushSetup request. The optional [Template-URL \(section 2.2.2.1.1\)](#) syntax element in the message body of the POST request specifies the path to an existing publishing point on the server that is to be as a template when creating the new publishing point.

The client can specify if the server should remove the publishing point after the streaming session has ended. This removal is performed by using the [AutoDestroy \(section 2.2.2.1.2\)](#) syntax element in the message body of the POST request.

A server that receives a POST request can identify it as a PushSetup request through the [Content-Type \(section 2.2.1.1\)](#) header. The media type specified on the Content-Type header MUST be "application/x-wms-pushsetup".

The syntax of the PushSetup request is defined as follows:

```
WMS-PushSetup-Request      = WMS-PushSetup-Req-Line
                             WMS-PushSetReq-Headers
                             CRLF
                             WMS-PushSetReq-Body

WMS-PushSetup-Req-Line     = "POST" SP Request-URI SP HTTP-Version CRLF

WMS-PushSetReq-Headers= *( PushSetReq-Header-REQ
                           / PushSetReq-Header-OPT
                           / HTTP-Header-Types )

PushSetReq-Header-REQ      = Content-Length ; section 14.13 of [RFC2616]
                           / Content-Type ; section 2.2.1.1
                           / Host ; section 14.23 of [RFC2616]
                           / User-Agent ; section 2.2.1.7

PushSetReq-Header-OPT      = Authorization ; [RFC2616] section 14.8
                           / Cache-Control; [RFC2616] section 14.9
                           / Cookie ; section 2.2.1.2
                           / Proxy-Authorization ; [RFC2616] section 14.34
                           / X-Accept-Authentication ; section 2.2.1.8

WMS-PushSetReq-Body        = [ *1Template-Url ; section 2.2.2.1.1
                             AutoDestroy ] ; section 2.2.2.1.2
```

The syntax of the PushSetup response is defined as follows:

```
WMS-PushSetup-Response     = Status-Line
                             WMS-PushSetResp-Headers

WMS-PushSetResp-Headers    = *( PushSetResp-Header-REQ
                              / PushSetResp-Header-OPT
                              / PushSetResp-Pragma
                              / HTTP-Header-Types )

PushSetResp-Header-REQ     = Cache-Control; section 2.2.1.1 of [MS-WMSP]
                              / Server ; section 2.2.1.4
                              / Set-Cookie ; section 2.2.1.5
```

```

PushSetResp-Header-OPT = Content-Length ; section 14.13 of [RFC2616]
                        / Location; [RFC2616] section 14.30
                        / Proxy-Authenticate ; [RFC2616] section 14.33
                        / Supported ; section 2.2.1.6
                        / Via; [RFC2616] section 14.45
                        / WWW-Authenticate; [RFC2616] section 14.47

PushSetResp-Pragma      = "Pragma: " #PushSetResp-Pragma-Types CRLF

PushSetResp-Pragma-Types = PushSetResp-Token-REQ
                        / PushSetResp-Token-OPT

PushSetResp-Token-REQ    = no-cache ; section 2.2.1.3.1

PushSetResp-Token-OPT    = timeout ; section 2.2.1.3.2

```

2.2.2.1.1 Template-URL

This directive instructs the server to create a new publishing point that will be identified by using the URL that is specified on the request line of the POST request. The new publishing point **SHOULD** be created using the same settings as an existing publishing point on the server identified by the path-absolute syntax element.

Exactly which settings are copied from the existing publishing point to the new publishing point is implementation-specific.

If the publishing point identified on the request line of the POST request already exists, the server **MUST** ignore the Template-URL directive.

The Template-URL directive **MUST** use ASCII characters. Thus, if the existing publishing point is identified using a string of Unicode characters, those characters **MUST** first be **encoded** by using UTF-8, as specified in [\[RFC3629\]](#), and any unsafe characters in the resulting string **MUST** be encoded using percent-encoding, as specified in [\[RFC3986\]](#) section 2.1.

The syntax of the Template-URL directive is defined as follows:

```

Template-URL= "Template-URL:" [SP] %x22
              path-absolute; section 3.3 of [RFC3986]
              %x22 CRLF

```

Example: Template-URL: "/pub"

2.2.2.1.2 AutoDestroy

This directive requests the server to destroy the publishing point at the end of the streaming session.

A value of 1 means that the server has been requested to remove the publishing point at the end of the streaming session. A value of 0 means that the server has been requested to keep the publishing point at the end of the streaming session. By keeping the publishing point, the client (or possibly a different client) can connect to the server at a later time and start streaming data to the same publishing point.

If the AutoDestroy directive is not specified, the server **MUST** assume a value of 0.

The syntax of the AutoDestroy directive is defined as follows:

```
AutoDestroy= "AutoDestroy:" [SP] ("0" / "1") CRLF
```

Example: AutoDestroy: 1

2.2.2.2 PushStart Request

The purpose of the PushStart request is to stream ASF data packets to the server. The message body of the first PushStart request that the client sends after the [PushSetup request](#) always begins with a [\\$H](#) packet (which contains the ASF file header), which is then followed by the [\\$D](#) packets (each of which contains an ASF data packet). Subsequent PushStart requests can start either with a [\\$C](#) packet or a \$D packet.

A server that receives a POST request can identify it as a PushStart request through the [Content-Type \(section 2.2.1.1\)](#) header. The media type specified on the Content-Type header MUST be "application/x-wms-pushstart".

The syntax of the PushStart request is defined as follows:

```
WMS-PushStart-Request      = WMS-PushStart-Req-Line
                             WMS-PushStrtReq-Headers
                             CRLF
                             WMS-PushStrtReq-Body

WMS-PushStart-Req-Line     = "POST" SP Request-URI SP HTTP-Version CRLF

WMS-PushStrtReq-Headers    = *( PushStrtReq-Header-REQ
                                / PushStrtReq-Header-OPT
                                / HTTP-Header-Types )

PushStrtReq-Header-REQ     = Content-Length ; section 14.13 of [RFC2616]
                            / Content-Type; section 2.2.1.1
                            / Cookie; section 2.2.1.2
                            / Host; section 14.23 of [RFC2616]
                            / User-Agent; section 2.2.1.7

PushStrtReq-Header-OPT     = Authorization      ; [RFC2616] section 14.8
                            / Cache-Control     ; section 2.2.1.1 of [MS-WMSP]
                            / Proxy-Authorization ; [RFC2616] section 14.34
                            / X-Accept-Authentication; section 2.2.1.8

Playlist-Entry             = *<$D Data packet> ; section 2.2.3.3
                            ( 1*<$E EOS packet>; section 2.2.3.4
                              / 1*<$F Fill packet> ) ; section 2.2.3.5

WMS-PushStrtReq-Body       = *1<$H Header packet>; section 2.2.3.6
                            Playlist-Entry
                            *( <$C packet> ; section 2.2.3.2
                              Playlist-Entry )
```

The syntax of the PushStart response is defined as follows:

```
WMS-PushStart-Response     = Status-Line
                             WMS-PushStrtResp-Headers

WMS-PushStrtResp-Headers    = *( PushStrtResp-Header-REQ
```

```

/ PushStrtResp-Header-OPT
/ PushStrtResp-Pragma
/ HTTP-Header-Types )

PushStrtResp-Header-REQ = Cache-Control; section 2.2.1.1 of [MS-WMSP]
/ Server; section 2.2.1.4

PushStrtResp-Header-OPT = Proxy-Authenticate ; [RFC2616] section 14.33
/ Set-Cookie; section 2.2.1.5
/ Supported ; section 2.2.1.6
/ Via; [RFC2616] section 14.45
/ WWW-Authenticate; [RFC2616] section 14.47

PushStrtResp-Pragma = "Pragma: " #PushStrtResp-Pragma-Types CRLF
PushStrtResp-Pragma-Types = PushStrtResp-Token-REQ
/ PushStrtResp-Token-OPT

PushStrtResp-Token-REQ = no-cache; section 2.2.1.3.1

PushStrtResp-Token-OPT = timeout; section 2.2.1.3.2

```

2.2.3 Packet Types

This section defines the packet types used by the Windows Media HTTP Push Distribution Protocol. The packets appear in the message body of a [PushStart request \(section 2.2.2.2\)](#) sent by the client to the server.

All packet types start with a [framing header \(section 2.2.3.1.1\)](#). All packet types, except the [\\$E](#) packet, are followed by a variable-size field called the **Payload** field. The interpretation of the **Payload** field, is specified in the definition of each packet type, when applicable.

The remainder of this section includes the following:

- [Common definitions](#) defines data structures and field definitions that are common to multiple packet types.
- The remaining sections describe individual packet types.

2.2.3.1 Common Definitions

All integer fields are transmitted in **little-endian** byte order. If a field is set to an invalid value, clients are free to handle that situation in an implementation-specific manner.

2.2.3.1.1 Framing Header

The Framing header is used by all packet types and is used as specified in [\[MS-WMSP\]](#) section 2.2.3.1.1 with the following additional details.

B: This flag MUST always be set to 0.

PacketLength: This field MUST be set to the size of the **Payload** field, if any, that follows the Framing header, plus the size of the **Reason** field, if any. Thus, the value of this field MUST be equal to the number of bytes in the packet, counted starting from the end of this field.

2.2.3.2 \$C (Stream Change Notification)

The \$C (Stream Change Notification) packet is used to send the new ASF file header to the server when the client has switched to the next entry in a **playlist** or otherwise made a change to the ASF file header.

The \$C packet MUST start with a [Framing header \(section 2.2.3.1.1\)](#) with the following additional details.

PacketID: This field MUST be set to the character "C" (0x43).

Reason: This field MUST be set to 0x00000000.

The variable-size **Payload** field MUST contain the ASF file header. The ASF file header consists of the entire ASF Header Object (as specified in [\[ASF\]](#) section 3.1), plus the 50-byte fixed initial portion of the ASF Data Object (as specified in [\[ASF\]](#) section 5.1). The size of the ASF file header MUST NOT be larger than 65527 bytes.

2.2.3.3 \$D (Data)

The \$D (Data) packet is used by the client to transfer an ASF data packet to the server.

The \$D packet MUST start with a [Framing header \(section 2.2.3.1.1\)](#) with the following additional details.

PacketID: This field MUST be set to the "D" (0x44) character.

Reason: This field MUST NOT be present.

The variable-size **Payload** field MUST contain exactly one complete ASF data packet. If the ASF data packet contains a **Padding Data** field (defined in [\[ASF\]](#) section 5.2.4), that field SHOULD be removed before encapsulating the ASF data packet in the \$D packet. If the **Padding Data** field is removed, the **Padding Length** field in the ASF payload parsing information section ([\[ASF\]](#) section 5.2.2) MUST be updated to indicate a nonexistent **Padding Data** field.

2.2.3.4 \$E (End-of-Stream Notification)

The \$E (End-of-Stream Notification) packet is used by the client to specify that the last [\\$D \(Data\)](#) packet for the **content** has been transmitted. The \$E packet also specifies whether this was the last content in a playlist or if the client should expect to receive a [\\$C \(Stream Change Notification\)](#) packet.

The \$E packet is defined as a [Framing header \(section 2.2.3.1.1\)](#) with the following additional details.

PacketID: This field MUST be set to the "E" (0x45) character.

Reason: This field MUST be present. The HRESULT code specifies the error, if any, that caused the server to send the \$E packet. HRESULT codes that have special meaning in the context of a \$E packet are defined in the following table.

Value	Meaning
S_OK 0x00000000	The server has finished streaming and no more \$D packets will be transmitted until the next PushStart request .

Value	Meaning
S_FALSE 0x00000001	The server has finished streaming the current playlist entry. Other playlist entries still remain to be streamed. The server will transmit a \$C packet when it switches to the next entry.

Unlike other packet types, the \$E packet does not have a **Payload** field.

2.2.3.5 \$F (Filler)

The purpose of the \$F (Filler) packet is to increase the size of the message body of the POST request.

The \$F packet MUST start with a [Framing header \(section 2.2.3.1.1\)](#) with the following additional details.

PacketID: This field MUST be set to the character "F"(0x46).

Reason: This field MUST NOT be present.

The variable-size **Payload** field MUST consist of zero or more padding bytes. The maximum size of this field is 65531 bytes. Each byte SHOULD be set to 0x00 and MUST be ignored by the server.

2.2.3.6 \$H (Header)

The \$H (Header) packet is used to send the first ASF file header to the server.

The \$H packet MUST start with a [Framing header \(section 2.2.3.1.1\)](#) with the following additional details.

PacketID: This field MUST be set to the character "H" (0x48).

Reason: This field MUST NOT be present.

The variable-size **Payload** field MUST contain an ASF file header. The ASF file header consists of the entire ASF Header Object (as specified in [\[ASF\]](#) section 3.1) plus the 50-byte fixed initial portion of the ASF Data Object (as specified in [\[ASF\]](#) section 5.1). The size of the ASF file header MUST NOT be larger than 65531 bytes.

3 Protocol Details

3.1 Client Details

3.1.1 Abstract Data Model

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

Length-Remaining: An unsigned numerical value that stores the number of bytes that the client has to transmit until it has transmitted the entire message body in the [PushStart request](#). The initial value of this variable is 0.

Push-ID: A string variable that stores the value of the [push-id](#) HTTP cookie. The initial value of this variable is "0".

UsingProxy: A flag that is set to 1 if the client is connecting to the server through a proxy server. The initial value of this variable is 0.

3.1.2 Timers

None.

3.1.3 Initialization

Initialization of the protocol occurs when the higher layer configures the server to receive streaming content. This event is specified in section [3.1.4.1](#).

The variables specified by the abstract data model MUST initially assume their default values, if any.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Request to Configure the Server

When the higher layer needs to configure the server to receive content, it MUST cause the client to send a [PushSetup request](#) to the server as detailed in the remainder of this section..

The higher layer MUST provide the URL that will be specified in all requests sent by the client. The URL specifies a "publishing point" on the server, which is the destination of the ASF data packets that the client intends to stream to the server. (A publishing point is similar to a broadcast channel.)

If this is the first request that is sent by the client, the client MUST perform the initialization of the protocol as specified in section [3.1.3](#).

The client MUST then establish a TCP connection to the server, using the IP address and port number obtained by parsing the URL. Use of HTTP proxy servers is permitted, in which case the TCP connection is made to the proxy server instead of to the server specified in the URL. If a proxy server is used, the value of the UsingProxy flag in the abstract data model MUST be set to 1.

Next, the client MUST send the PushSetup request to the server, as specified in section [3.1.4.1.1](#).

3.1.4.1.1 Sending the PushSetup Request

The [PushSetup request](#) MUST adhere to the syntax specified in section [2.2.2.1](#).

The request sent by the client MUST NOT specify any of the headers and tokens specified in section [2.2.1](#), which are defined only for use in responses.

The client MUST specify the [User-Agent \(section 2.2.1.7\)](#) header in the request.

The client MUST specify the [Content-Type \(section 2.2.1.1\)](#) header.

The client MUST specify the Content-Length header as specified in [\[RFC2616\]](#) section 14.13. Thus, it follows that chunked transfer coding (as specified in [\[RFC2616\]](#) section 3.6.1) MUST NOT be used.

The client SHOULD specify the [X-Accept-Authentication \(section 2.2.1.8\)](#) header. <1>

If the client is responding to an HTTP authentication challenge, it MUST include the Authorization header (as specified in [\[RFC2616\]](#) section 14.8) if the challenge is from a server or the Proxy-Authorization header (as specified in [\[RFC2616\]](#) section 14.34) if the challenge is from a proxy.

The [Cookie](#) header SHOULD be included in the request. The [push-id](#) cookie MUST be included on the Cookie header, and the value of the cookie MUST be equal to the value of the Push-ID variable.

If the higher layer specifies a "template" publishing point on the server, the name of that publishing point MUST be included by using the [Template-URL \(section 2.2.2.1.1\)](#) directive in the message body of the POST request. (If the publishing point identified by the URL on the HTTP request line does not yet exist, the server might use the Template-URL publishing point as a "template" when creating the publishing point.)

If the higher-layer requests that the publishing point identified by the URL on the HTTP request line be removed at the end of the streaming session, the message body of the POST request MUST include the [AutoDestroy \(section 2.2.2.1.2\)](#) directive. If the Template-URL directive is included in the message body, the AutoDestroy directive SHOULD also be included, even if the higher layer does not request the publishing point to be removed.

After sending the request, the client MUST wait for the response to be received. How to process the response is specified in section [3.1.5.1](#).

3.1.4.2 Request to Start Streaming Content

When a higher layer requests that the streaming of content starts, it causes the client to send a [PushStart request](#) to the server.

As a prerequisite for this event, the higher layer MUST already have configured the server to receive content, as specified in section [3.1.4.1](#), and the client MUST have received a response to the [PushSetup request](#) that indicated that the PushSetup request succeeded (as specified in section [3.1.5.1](#)), but the client MUST NOT have sent any PushStart requests previously.

The higher layer MUST provide an ASF file header to send to the server. The client MUST encapsulate it in a [\\$H](#) packet, according to the rules specified in section [2.2.3.6](#).

Next, the client MUST send the PushStart request to the server, as specified in section [3.1.4.2.1](#).

Immediately after sending the HTTP request line and the HTTP headers of the PushStart request, the client MUST send the \$H packet to the server. Thus, the message body of the PushStart request begins with the \$H packet.

3.1.4.2.1 Sending the PushStart Request

If the TCP connection that was used for sending the most recent HTTP request to the server was closed, the client MUST establish a new TCP connection to the server by using the IP address and port number obtained by parsing the URL. (This is the same URL that was used for the [PushSetup request](#).) Using HTTP proxy servers is permitted, in which case the TCP connection is made to the proxy server instead of to the server specified in the URL. If a proxy server is used, the value of the **UsingProxy** flag in the abstract data model MUST be set to 1.

The [PushStart request](#) MUST adhere to the syntax specified in section [2.2.2.2](#).

The request sent by the client MUST NOT specify any of the headers and tokens specified in section [2.2.1](#) that are defined only for use in responses.

The client MUST specify the [User-Agent \(section 2.2.1.7\)](#) header in the request.

The client MUST specify the [Content-Type \(section 2.2.1.1\)](#) header.

If the value of the **UsingProxy** variable in the abstract data model is 1, the value of the **Length-Remaining** variable in the abstract data model SHOULD be set to approximately the number of bytes that the client expects to stream to the server in a 60-second period. **Length-Remaining** MAY be set to 2147483647.

If the value of the **UsingProxy** variable in the abstract data model is 0, the value of the **Length-Remaining** variable SHOULD be set to 2147483647.

Under any circumstances, the value of Length-Remaining MUST NOT be set to a value that is less than the size of three maximum-size [\\$D](#) packets or less than the maximum size of a [\\$H](#) or [\\$C](#) packet that the client intends to send plus 4. A maximum-size \$D packet is a \$D packet where the ASF data packet did not have a **Padding Data** field to remove prior to encapsulation in the \$D packet. Details are specified in section [2.2.3.3](#).

The client MUST specify the Content-Length header (as specified in [\[RFC2616\]](#) section 14.13), and the numerical value specified on that header MUST be equal to the value of the Length-Remaining variable. It follows from the use of a Content-Length header that chunked transfer coding (as specified in [\[RFC2616\]](#) section 3.6.1) MUST NOT be used.

The client SHOULD specify the [X-Accept-Authentication \(section 2.2.1.8\)](#) header. [<2>](#)

If the client is responding to an HTTP authentication challenge, it MUST include the Authorization header (as specified in [\[RFC2616\]](#) section 14.8) if the challenge is from a server, or the Proxy-Authorization header (as specified in [\[RFC2616\]](#) section 14.34) if the challenge is from a proxy.

The [Cookie \(section 2.2.1.2\)](#) header MUST be included in the request. The [push-id](#) cookie MUST be included on the Cookie header, and the value of Push-ID variable in the abstract data model MUST be used as the value for that cookie.

Next, the client MUST send the request line and headers of the PushStart request, but not the message body.

If this is the first PushStart request that is being sent, the client MUST send the \$H packet at this time, as specified in section [3.1.4.2](#).

The client MUST then notify the higher layer that it has sent the PushStart request because the sending of this request might trigger events from the higher layer.

The client MUST then wait for a higher-layer triggered event or for the response to the PushStart request to be received. How to process the response is specified in section [3.1.5.2](#).

3.1.4.3 ASF Packet Is Available to Send

Notification that an ASF packet is available to send occurs while the client is sending a [PushStart request](#), as specified in section [3.1.4.2.1](#). At this point, the client has already sent the request line and the HTTP headers of the PushStart request, but has not yet completed sending the message body of the request.

The value of the **Length-Remaining** variable in the abstract data model MUST be greater than zero. If the value of the **Length-Remaining** variable is zero, the ASF data packet cannot be accepted at this time, and the higher layer MUST hold on to the ASF data packet (that is, queue it) until **Length-Remaining** becomes greater than zero.

The higher layer MUST provide an ASF data packet to send to the server. The client MUST encapsulate it in a [\\$D \(Data\)](#) packet, according to the rules specified in section [2.2.3.3](#).

If the size of the \$D packet, in bytes, plus 4, is larger than the value of the **Length-Remaining** variable in the abstract data model, the client MUST send a [\\$F \(Filler\)](#) packet to the server, as specified in section [2.2.3.5](#). The size of the **Payload** field in the \$F packet MUST be equal to the value of **Length-Remaining** minus 4, except if **Length-Remaining** is greater than 65535. If **Length-Remaining** is greater than 65535, the client MUST send multiple \$F packets, choosing the size of each \$F packet such that the sum of all sizes of \$F packets equal the value of **Length-Remaining**. The value of **Length-Remaining** MUST then be set to zero.

If the client is sending one or more \$F packets, the ASF data packet MUST be returned to the higher layer (that is, queued at the higher layer). Because **Length-Remaining** is zero, the ASF data packet cannot be sent at this time.

If the value of **Length-Remaining** is greater than zero, the client MUST send the \$D packet to the server and the value of **Length-Remaining** MUST be decreased by the number of bytes in the \$D packet.

If the value of **Length-Remaining** is now equal to zero, the client MUST wait for the response to the PushStart request to be received.

Otherwise, the client MUST wait for either another higher-layer triggered event or for the response to the PushStart request to be received.

How to process the response to the PushStart request is specified in section [3.1.5.2](#).

3.1.4.4 Notification of the Last Packet

Notification that the last ASF packet has been sent occurs while the client is sending a [PushStart request](#), as specified in section [3.1.4.2.1](#). At this point, the client has already sent the PushStart request line and the HTTP headers, but has not yet completed sending the message body of the request.

The value of the **Length-Remaining** variable in the abstract data model MUST be greater than zero. If the value of the **Length-Remaining** variable is zero, the notification cannot be accepted at this time, and the higher layer MUST hold on to the notification (that is, queue it) until **Length-Remaining** becomes greater than zero.

If the value of **Length-Remaining** is less than eight, the client MUST send a [\\$F \(Filler\)](#) packet to the server, as specified in section [2.2.3.5](#). The size of the **Payload** field in the \$F packet MUST be equal to four. The value of **Length-Remaining** MUST then be set to zero.

If the client is sending a \$F packet, the notification MUST be returned to the higher layer (that is, queued at the higher layer). Because **Length-Remaining** is zero, the notification cannot be processed at this time.

In case the value of **Length-Remaining** is greater than or equal to 8, the client MUST create a [\\$E \(End-of-Stream Notification\)](#) (section [2.2.3.4](#)) packet.

The higher layer MUST specify if a new ASF file header is forthcoming, that is, there are additional entries in the server side playlist, or if this was the last entry in the playlist, so that the **Reason** field in the \$E packet can be filled in correctly.

If the value of **Length-Remaining** is greater than or equal to eight, then the client MUST send the \$E packet to the server, and the value of **Length-Remaining** MUST be decreased by eight.

If the \$E packet was sent to the server, and the value of the **Reason** field was not equal to 0x00000001 (which means that there is not a new ASF file header forthcoming), the client SHOULD close the TCP connection to the server. This action also ends the streaming session. All session state SHOULD be deleted, and this event MUST be reported to the higher layer.

If the TCP connection to the server is still open, and if the value of Length-Remaining is now equal to zero, the client MUST wait for the response to the PushStart request to be received.

Otherwise, as long as the TCP connection is still open, the client MUST wait for either another higher-layer triggered event or for the response to the PushStart request to be received.

How to process the response to the PushStart request is specified in section [3.1.5.2](#).

3.1.4.5 Notification of New ASF Header File

As a prerequisite for notification that a new ASF header file is available, the higher layer MUST already have notified the server that it has sent the last ASF data packet for the previous playlist entry, as specified in section [3.1.4.4](#).

This event occurs while the client is sending a [PushStart request](#) (section [2.2.2.2](#)). At this point, the client has already sent the request line and the HTTP headers of the PushStart request, and has sent a [\\$E](#) packet, but has not yet completed sending the message body of the request.

The value of the **Length-Remaining** variable in the abstract data model MUST be greater than zero. If the value of the **Length-Remaining** variable is zero, the ASF file header cannot be accepted at this time, and the higher layer MUST hold on to the ASF file header (that is, queue it) until **Length-Remaining** becomes greater than zero.

The higher layer MUST provide an ASF file header to send to the server. The client MUST encapsulate it in a [\\$C](#) packet, according to the rules specified in section [2.2.3.2](#).

If the size of the \$C packet, in bytes, plus 4, is larger than the value of the **Length-Remaining** variable in the abstract data model, the client MUST send a [\\$F](#) packet to the server. The size of the **Payload** field in the \$F packet MUST be equal to the value of **Length-Remaining** minus 4, except if **Length-Remaining** is greater than 65535. If **Length-Remaining** is greater than 65535, the client MUST send multiple \$F packets, choosing the size of each \$F packet such that the sum of all sizes of \$F packets equal the value of **Length-Remaining**. The value of **Length-Remaining** MUST then be set to zero.

If the client is sending one or more \$F packets, the ASF file header MUST be returned to the higher layer (that is, queued at the higher layer). Because **Length-Remaining** is zero, the ASF file header cannot be sent at this time.

If the value of **Length-Remaining** is greater than zero, the client MUST send the \$C packet to the server and the value of **Length-Remaining** MUST be decreased by the number of bytes in the \$C packet.

If the value of **Length-Remaining** is now equal to zero, the client MUST wait for the response to the PushStart request to be received.

Otherwise, the client MUST wait for either another higher-layer triggered event or for the response to the PushStart request to be received.

How to process the response to the PushStart request is specified in section [3.1.5.2](#).

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Receiving a PushSetup Response

The client MUST verify that the response adheres to the syntax specified in section [2.2.2.1](#).

If the HTTP status code in the response is in the range 300 through 305, the server is requesting the client to connect to another server. The client MUST connect to the server specified in the response, by following the rules as specified in [\[RFC2616\]](#) section 10.3. A brief summary of the rules is: If the status code is 305, the URL on the Location header (as specified in [\[RFC2616\]](#) section 14.30) is for a proxy, and the URL that is used in the PushSetup request MUST remain unchanged. For status codes 300 through 304, the URL on the Location header MUST replace the URL used in the PushSetup request. The server MUST close the current TCP connection and establish a new TCP connection to the server, or proxy server, as appropriate depending on the status code. If a proxy server is used, the value of the **UsingProxy** flag in the abstract data model MUST be set to one. Otherwise, it MUST be set to zero. The client MUST then continue by following the steps specified in section [3.1.4.1.1](#).

If the HTTP status code in the response is 401, the server requires access authentication. Status code 407 means that a proxy server requires access authentication. The rules for access authentication, as specified in [\[RFC2616\]](#) section 11, MUST be followed. The client SHOULD keep the TCP connection open because it will be needed to resubmit the request. When the client is ready to resubmit the HTTP request with the authentication credentials that the server requested, the HTTP request MUST be sent on the same TCP connection on which the 401 or 407 response was received, unless that TCP connection has been closed. If the TCP connection has been closed, the client MUST establish a new TCP connection to the server and send the HTTP request on that connection. When resubmitting the request, the client MUST follow the steps specified in section [3.1.4.1.1](#).<3>

Otherwise, if the HTTP status code indicates that the request succeeded, the client MUST perform the following steps.

1. The client MUST inspect the [Server \(section 2.2.1.4\)](#) header in the response. If the Server header is missing or if the server-token parameter on the Server header does not match any of the values used by this protocol (as specified in [\[MS-WMSP\]](#) section 2.2.1.5), the server is not compatible with the Windows Media HTTP Push Distribution Protocol and is probably a regular HTTP Web server. Because this protocol does not interoperate with incompatible servers, this incompatibility MUST be treated as an error and reported as such to the higher layer.
2. If the [Set-Cookie \(section 2.2.1.5\)](#) header is present in the response, the value of the **Push-ID** variable in the abstract data model MUST be set to the value of the [push-id](#) cookie. Any other

cookies that are specified SHOULD be saved for inclusion in the [Cookie](#) header in subsequent requests.

3. If the Via header (as specified in [\[RFC2616\]](#) section 14.45) is present in the response, the value of the **UsingProxy** flag in the abstract data model MUST be set to one.
4. The client SHOULD keep the TCP connection to the server open, so that it can be used for sending the [PushStart request](#).
5. The server MUST inform the higher layer that the PushSetup request succeeded and that it is now possible to start streaming data to the server. The client MUST wait for a higher-layer triggered event.

3.1.5.2 Receiving a PushStart Response

The client MUST verify that the response adheres to the syntax specified in section [2.2.2.2](#).

If the HTTP status code in the response is 401, the server requires access authentication. Status code 407 means that a proxy server requires access authentication. The rules for access authentication as specified in [\[RFC2616\]](#) section 11 MUST be followed. If the value of the **Length-Remaining** variable in the abstract data model is greater than zero, the client MUST close the TCP connection. Otherwise, the client SHOULD keep the TCP connection open because it will be needed to resubmit the request. When the client is ready to resubmit the HTTP request with the authentication credentials that the server requested, the HTTP request MUST be sent on the same TCP connection on which the 401 or 407 response was received, unless that TCP connection has been closed. If the TCP connection has been closed, the client MUST establish a new TCP connection to the server and send the HTTP request on that connection. When resubmitting the request, the client MUST follow the steps specified in section [3.1.4.2.1.<4>](#)

Otherwise, if the HTTP status code indicates that the request succeeded, the client MUST perform the following steps.

1. The client MUST inspect the [Server \(section 2.2.1.4\)](#) header in the response. If the Server header is missing, or if the server-token parameter on the Server header does not match any of the values used by this protocol (as specified in [\[MS-WMSP\]](#) section 2.2.1.5), the server is not compatible with the Windows Media HTTP Push Distribution Protocol and is probably a regular HTTP Web server. Because this protocol does not interoperate with incompatible servers, this incompatibility MUST be treated as an error and reported as such to the higher layer.
2. If the [Set-Cookie \(section 2.2.1.5\)](#) header is present in the response, the value of the Push-ID variable in the abstract data model MUST be set to the value of the [push-id](#) cookie. Any other cookies specified SHOULD be saved for inclusion in the [Cookie](#) header in subsequent requests.
3. If the Via header (as specified in [\[RFC2616\]](#) section 14.45) is present in the response, the value of the UsingProxy flag in the abstract data model MUST be set to 1.
4. The client MUST send a [PushStart request](#) to the server, as specified in section [3.1.4.2.1](#).

3.1.6 Timer Events

None.

3.1.7 Other Local Events

3.1.7.1 TCP Connection Is Disconnected

If the TCP connection to the server is disconnected when the client has sent a request (but before it has completely received the response), and it was not the client itself that initiated the disconnection, the client **MUST** report this as an error to the higher layer.

3.2 Server Details

3.2.1 Abstract Data Model

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

AutoDestroy: This flag is set to TRUE if the resource identified by the URL in the [PushSetup](#) or [PushStart](#) requests (sometimes called "publishing point") should be deleted at the end of the streaming session. The initial value of this flag is FALSE.

Push-ID: This string variable stores an identifier assigned by the server for the current streaming session. The initial value is undefined.

3.2.2 Timers

[Idle-Timeout:](#) This timer is used for cleaning up unused session states. If no requests are received from the client, the Idle-Timeout timer will expire and the server is then free to delete the session state. The minimum allowed value for the time-out period is 10 seconds and the maximum value is 4,294,967,295 milliseconds. The default value of this timer is 60 seconds.

[Inactivity-Timeout:](#) This timer is used for cleaning up unused session states. If no requests are received from the client, the Inactivity-Timeout timer will expire and the server is then free to delete the session state. The minimum allowed value for the time-out period is 0 seconds; there is no maximum value. The default value of this timer is 300 seconds.

3.2.3 Initialization

Initialization of the protocol occurs when a [PushSetup request](#) is received and the request did not specify a [push-id](#) cookie on the [Cookie](#) header or the value of the push-id cookie did not match the value of the **Push-ID** variable.

The variables defined by the abstract data model **MUST** initially assume their default values. Variables that do not have a default defined **MUST** be initialized as follows:

The value of the **Push-ID** variable **SHOULD** be assigned to a random alphanumerical string. If the server allows multiple simultaneous streaming sessions, each instance of the protocol state **MUST** use a different value for the **Push-ID** variable. The **Push-ID** variable **MUST NOT** be set to the string "0".

The Idle-Timeout timer **MUST** be started.

3.2.4 Higher-Layer Triggered Events

3.2.4.1 Administrative Disconnect

This event occurs when the server administrator wants to disconnect a client from the server. If the server is currently receiving a [PushStart request](#), the server MUST send a [PushStart](#) response with some suitable HTTP status code, such as 500. The response MUST be sent immediately, that is, without waiting for the complete message body of the request to be received. How to send a PushStart response is specified in section [3.2.5.2.1](#).

If the value of the **AutoDestroy** flag in the abstract data model is TRUE, the server SHOULD remove the "publishing point", that is, the resource identified by the client in the PushStart request.

After having sent the response, if any, the server MUST close the TCP connection to the client.

The server MUST delete all session state.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Receiving a PushSetup Request

The server MUST inspect the [User-Agent \(section 2.2.1.7\)](#) header in the HTTP request. If the User-Agent header is missing, or if the user-agent-data parameter on the User-Agent header does not adhere to the syntax specified in section [2.2.1.7](#), the client is not compatible with the Windows Media HTTP Push Distribution Protocol and is probably a regular HTTP Web client. Because this protocol does not interoperate with incompatible clients, this incompatibility MUST be treated as an error. The server MAY respond to the request with an HTTP error status code, or it MAY respond with a success code, or return data that redirects the Web client to some other suitable URL.[<5>](#)

The server MUST validate that the HTTP request adheres to the syntax for [PushSetup requests \(section 2.2.2.1\)](#). If the validation fails, the server MUST respond with some HTTP error status code, such as 400, or 501, as appropriate.

The server MUST check with the higher layer to determine whether the client will be redirected to a different server or to a proxy server. (The presence, or absence, of a Via header in the request can be used to determine whether the request was delivered directly by the client or through a proxy server. The Via header is as specified in [\[RFC2616\]](#) section 14.45.)

If the higher layer indicates that the client shall be redirected to another server, the server MUST respond with status code 302. If the client shall be redirected to a proxy server, the server SHOULD respond with status code 305. In both cases, the URL of the server, or proxy server, MUST be specified on the Location header (as specified in [\[RFC2616\]](#) section 14.30) in the response.

After having sent a response with status codes 302 or 305, the server MUST delete the session state, if any, and close the TCP connection to the client.

If the higher layer requires the client to authenticate itself, the server MUST process the Authorization header (as specified in [\[RFC2616\]](#) section 14.8) if it is present in the request. If the server is acting as a proxy server, it MUST process the Proxy-Authorization header (as specified in [\[RFC2616\]](#) section 14.34) instead of the Authorization header.

If it is necessary to send an authentication challenge to the client (for example, because the Authorization header was missing or specified incorrect credentials), the server SHOULD use one of the authentication schemes that the client listed on the [X-Accept-Authentication \(section 2.2.1.8\)](#) header in the request, if that header is present.

If the server sends an authentication challenge to the client, it MUST be specified using the WWW-Authenticate header (as specified in [\[RFC2616\]](#) section 14.34), and the status code of the response MUST be 401. If the server is acting as a proxy server, it MUST specify the Proxy-Authenticate header (as specified in [\[RFC2616\]](#) section 14.33) instead of the WWW-Authenticate header, and the status code of the response MUST be 407. [<6>](#)

After having sent a response with status codes 401 or 407, if the authentication scheme used in the authentication challenge is NTLM (as specified in [\[MS-NLMP\]](#)), the server SHOULD NOT close the TCP connection to the client.

If the server is not sending a response with an error status code, and if the request includes a [Cookie](#) header with a [push-id \(section 2.2.1.2.1\)](#) cookie, and the value of the push-id cookie is not equal to "0", the server MUST load the state that has a Push-ID variable with the same value as the value of the push-id cookie. If the matching state cannot be found, this MUST be treated as an error. The server MUST respond to the client with some suitable HTTP error status code, such as 400. If the matching state is found and the server is currently receiving streaming data from the client on another TCP connection (that is, a [PushStart request](#) is in progress), this is an error situation. In this case, the server MUST respond to the PushSetup request with a suitable HTTP error status code, such as 409.

The server MUST check with the higher layer that the URL that the client specified in the request is valid. A valid URL might specify a resource that does not yet exist, in which case the message body of the request MUST include the [Template-URL \(section 2.2.2.1.1\)](#) directive that specifies an existing resource that the server is supposed to use as a template when creating the new resource.

If the URL is invalid, this is an error, and the server MUST respond with some suitable HTTP error status code, such as 404.

If the [AutoDestroy](#) directive is present in the message body, the value of the AutoDestroy flag in the abstract data model MUST be set to TRUE if the directive specified the value 1.

If the server is not sending a response with an error status code (for example, 302 or 305), and if the request does not specify a push-id cookie on a Cookie header or if the value of the push-id cookie is "0", the server MUST create a new state by performing the initialization procedure as specified in section [3.2.3](#). Otherwise, the Idle-Timeout timer MUST be restarted.

The [PushSetup](#) response MUST follow the rules in PushSetup request.

Because the PushSetup response does not have a message body, the status code in the response SHOULD be 204. The status code MAY be 200.

The response MUST include a [Set-Cookie \(section 2.2.1.5\)](#) header, and the value of the push-id cookie on that header MUST be equal to the value of the Push-ID variable in the abstract data model.

The response sent by the server MUST NOT specify any of the headers and tokens specified in section [2.2.1](#) that are defined only for use in requests.

After having sent the response, the TCP connection to the client SHOULD be left open as the client is likely to want to send a PushStart request on the same connection.

The Inactivity-Timeout timer MUST be started if it is not already running, or restarted if it is already running.

The server MUST wait for the PushStart request to be received. As soon as the request line and all of the HTTP headers in the PushStart request have been received, the server MUST process the rules

as specified in section [3.2.5.2](#). The server MUST NOT wait for the entire message body of the PushStart request to be received before processing the request.

3.2.5.2 Receiving a PushStart Request

The server MUST validate that the HTTP request line and the HTTP message headers adhere to the syntax for [PushStart requests \(section 2.2.2.2\)](#). If the validation fails, the server MUST respond with some HTTP error status code, such as 400, or 501, as appropriate.

If the higher layer requires the client to authenticate itself, the server MUST process the Authorization header (as specified in [\[RFC2616\]](#) section 14.8), if it is present in the request. If the server is acting as a proxy server, it MUST process the Proxy-Authorization header (as specified in [\[RFC2616\]](#) section 14.34) instead of the Authorization header.

If it is necessary to send an authentication challenge to the client (for example, because the Authorization header was missing or specified incorrect credentials), the server SHOULD use one of the authentication schemes that the client listed on the [X-Accept-Authentication \(section 2.2.1.8\)](#) header in the request if that header is present.

If the server sends an authentication challenge to the client, it MUST be specified by using the WWW-Authenticate header (as specified in [\[RFC2616\]](#) section 14.47), and the status code of the response MUST be 401. If the server is acting as a proxy server, it MUST specify the Proxy-Authenticate header (as specified in [\[RFC2616\]](#) section 14.33) instead of the WWW-Authenticate header, and the status code of the response MUST be 407. [<7>](#)

After having sent a response with status codes 401 or 407, if the authentication scheme used in the authentication challenge is NTLM (as specified in [\[MS-NLMP\]](#)) the server SHOULD NOT close the TCP connection to the client.

If the server is not sending a response with an error status code, the server MUST load the state that has a **Push-ID** variable with the same value as the value of the [push-id \(section 2.2.1.2.1\)](#) cookie on the [Cookie \(section 2.2.1.2\)](#) header. If the matching state cannot be found, or if the client did not specify a push-id cookie, this MUST be treated as an error. The server MUST respond to the client with a suitable HTTP error status code, such as 400. If the matching state is found and the server is currently receiving streaming data from the client on another TCP connection (that is, a previous PushStart request is still in progress), this is an error situation. In this case, the server MUST respond to the new PushStart request with a suitable HTTP error status code, such as 409.

The server MUST check with the higher layer that the URL that the client specified in the request is valid. If the URL is invalid, this is an error, and the server MUST respond with a suitable HTTP error status code, such as 404.

The Idle-Timeout and Inactivity-Timeout timer MUST be stopped.

If the server is sending a response with an error status code, or if the entire message body of the PushStart request has been received, as determined by the Content-Length header (as specified in [\[RFC2616\]](#) section 14.13), the server MUST send a [PushStart](#) response. How to send a PushStart response is specified in section [3.2.5.2.1](#).

Otherwise, the Idle-Timeout timer MUST be started and the server MUST wait for a higher-layer triggered event or one of the packet types specified in section [2.2.3](#) to be received as part of the HTTP message body. If this is the first PushStart request received for this session, the first packet MUST be a [\\$H](#) packet. Otherwise, it can be a [\\$C](#) packet, [\\$D](#) packet, [\\$E](#) packet, or [\\$F](#) packet.

How to process a \$C packet is specified in section [3.2.5.6](#). How to process a \$D packet is specified in section [3.2.5.4](#). How to process a \$E packet is specified in section [3.2.5.5](#). How to process a \$F packet is specified in section [3.2.5.7](#). How to process a \$H packet is specified in section [3.2.5.3](#).

3.2.5.2.1 Sending a PushStart Response

The PushStart response MUST follow the rules as specified in section [2.2.2.2](#).

If the response is not intended to convey a particular error status code, the status code in the response SHOULD be 204 because the PushStart response does not have a message body. The status code MAY be 200.

The response SHOULD include a [Set-Cookie \(section 2.2.1.5\)](#) header, and the value of the [push-id](#) cookie on that header MUST be equal to the value of the **Push-ID** variable in the abstract data model.

The response sent by the server MUST NOT specify any of the headers and tokens specified in section [2.2.1](#), which are defined only for use in requests.

After having sent the response, if the status code did not indicate an error, the TCP connection to the client SHOULD be left open because the client likely will send another PushStart request on the same connection.

Unless the session is being terminated, the server MUST stop the Idle-Timeout timer, MUST start the Inactivity-Timeout timer, and MUST wait for a [PushSetup request](#) or a PushStart request to be received.

How to process a PushSetup request is specified in section [3.2.5.1](#). If a PushStart request is being received, as soon as the request line and all of the HTTP headers in that request have been received, the server MUST process the rules as specified in section [3.2.5.2](#). The server MUST NOT wait for the entire message body of the PushStart request to be received before processing the request.

3.2.5.3 Receiving a \$H Packet

The server MUST validate that the packet adheres to the syntax for [\\$H \(Header\) \(section 2.2.3.6\)](#) packets.

The server MUST make the ASF file header available to the higher layer.

If there are no bytes left to receive in the message body of the [PushStart request](#), the server MUST send a [PushStart](#) response, as is specified in section [3.2.5.2.1](#).

Otherwise, the server MUST restart the Idle-Timeout timer and MUST wait for a higher-layer triggered event or for a [\\$D](#), [\\$E](#), or [\\$F](#) packet.

How to process a \$D packet is specified in section [3.2.5.4](#). How to process a \$E packet is specified in section [3.2.5.5](#). How to process a \$F packet is specified in section [3.2.5.7](#).

3.2.5.4 Receiving a \$D Packet

The server MUST validate that the packet adheres to the syntax for [\\$D \(Data\) \(section 2.2.3.3\)](#) packets.

The server MUST make the ASF data packet available to the higher layer.

If there are no bytes left to receive in the message body of the [PushStart request](#), the server MUST send a [PushStart](#) response, as is specified in section [3.2.5.2.1](#).

Otherwise, the server MUST restart the Idle-Timeout timer and MUST wait for a higher-layer triggered event or for a \$D, [\\$E](#), or [\\$F](#) packet.

How to process a \$E packet is specified in section [3.2.5.5](#). How to process a \$F packet is specified in section [3.2.5.7](#).

3.2.5.5 Receiving a \$E Packet

The server MUST validate that the packet adheres to the syntax for [\\$E \(End-of-Stream Notification\) \(section 2.2.3.4\)](#) packets.

The server MUST report to the higher layer that the last ASF data packet for the current playlist entry has been received.

If the value of the **Reason** field in the \$E packet is not 0x00000001, the server SHOULD close the TCP connection to the client without sending a [PushStart](#) response and MUST delete the session state. If the value of the **AutoDestroy** flag is TRUE, deleting the session state includes removing the "publishing point", that is, the resource identified by the client in the [PushStart request](#).

Otherwise, if there are no bytes left to receive in the message body of the PushStart request, the server MUST send a PushStart response. How to send a PushStart response is specified in section [3.2.5.2.1](#).

Otherwise, the server MUST restart the Idle-Timeout timer and MUST wait for a higher-layer triggered event or for a [\\$C](#), [\\$E](#), or [\\$F](#) packet.

How to process a \$C packet is specified in section [3.2.5.6](#). How to process a \$F packet is specified in section [3.2.5.7](#).

3.2.5.6 Receiving a \$C Packet

The server MUST validate that the packet adheres to the syntax for [\\$C \(Stream Change Notification\) \(section 2.2.3.2\)](#) packets.

The server MUST make the ASF file header available to the higher layer.

If there are no bytes left to receive in the message body of the [PushStart request](#), the server MUST send a [PushStart](#) response, as is specified in section [3.2.5.2.1](#).

Otherwise, the server MUST restart the Idle-Timeout timer and MUST wait for a higher-layer triggered event or for a [\\$D](#), [\\$E](#), or [\\$F](#) packet.

How to process a \$D packet is specified in section [3.2.5.4](#). How to process a \$E packet is specified in section [3.2.5.5](#). How to process a \$F packet is specified in section [3.2.5.7](#).

3.2.5.7 Receiving a \$F Packet

The server MUST validate that the packet adheres to the syntax for [\\$F \(Filler\) \(section 2.2.3.5\)](#) packets.

If there are no bytes left to receive in the message body of the [PushStart request](#), the server MUST send a [PushStart](#) response, as is specified in section [3.2.5.2.1](#).

Otherwise, the server MUST restart the Idle-Timeout timer and MUST wait for a higher-layer triggered event or for a [\\$C](#), [\\$D](#), [\\$E](#), or [\\$F](#) packet.

How to process a [\\$C](#) packet is specified in section [3.2.5.6](#). How to process a [\\$D](#) packet is specified in section [3.2.5.4](#). How to process a [\\$E](#) packet is specified in section [3.2.5.5](#).

3.2.6 Timer Events

3.2.6.1 Idle-Timeout Timer Expires

If the server is currently receiving a [PushStart request](#), the server MUST send a [PushStart](#) response with some suitable HTTP status code, such as 408. The response MUST be sent immediately, that is, without waiting for the complete message body of the request to be received. How to send a [PushStart](#) response is specified in section [3.2.5.2.1](#).

If the value of the **AutoDestroy** flag in the abstract data model is TRUE, the server SHOULD remove the "publishing point", that is, the resource identified by the client in the [PushStart request](#).

After having sent the response, if any, the server MUST close the TCP connection to the client.

The server MUST delete all session state.

3.2.6.2 Inactivity-Timeout Timer Expires

If the value of the **AutoDestroy** flag in the abstract data model is TRUE, the server SHOULD remove the "publishing point," that is, the resource identified by the client in the [PushStart request](#).

If the server has a TCP connection to the client, the connection MUST be closed.

The server MUST then delete all session state.

3.2.7 Other Local Events

3.2.7.1 TCP Connection Is Disconnected

If the TCP connection to the client is disconnected when the server has started to receive a request but before it has completely received the entire request (including the message body) and it was not the server itself that initiated the disconnection, the server SHOULD NOT treat this as a fatal error. Instead, the Idle-Timeout timer SHOULD be left running, allowing the client to resume the streaming session if it connects to the server again and sends a new request before the Idle-Timeout timer expires.

If the TCP connection to the client is disconnected between requests (for example, after a [PushStart request](#)) and its message body has been completely received, the Idle-Timeout timer MUST be stopped and the Inactivity-Timeout timer MUST be started.

4 Protocol Examples

The following sections describe several operations as used in common scenarios to illustrate the function of the Windows Media HTTP Push Distribution Protocol.

4.1 General Push Distribution Sequence

The following sequence occurs between a client and a server during a push distribution. Either HTTP version 1.1 or 1.0 can be used with this protocol.

1. The client sends a [PushSetup request](#).
2. If the server requires the client to be authenticated, the server and client exchange access authentication HTTP headers as specified in [\[RFC2616\]](#) section 11.
Note The HTTP exchanges required for authentication are defined by the selected authentication scheme.
3. If authentication is not required, or if authentication has succeeded, the server responds with a "204 No Content" HTTP response.
4. The client sends a [PushStart request](#), followed by a [\\$H \(Header\)](#) packet and [\\$D \(Data\)](#) packets.
5. When the client has sent the amount of data allowed in the PushStart request, the server responds with a "204 No Content" HTTP response. The client then sends a new PushStart request followed by more \$D (Data) packets.
6. After all \$D (Data) packets have been sent to the server, the client sends an [\\$E \(End-of-Stream Notification\)](#) packet to indicate that the data transfer has been completed.

Figure 1 shows a message sequence with a single PushStart request. Figure 2 shows a message sequence with three PushStart requests. Multiple PushStart requests are typically used when the client connects to the server through a HTTP proxy server, while a single PushStart request is typically sufficient when HTTP proxy servers are not used.

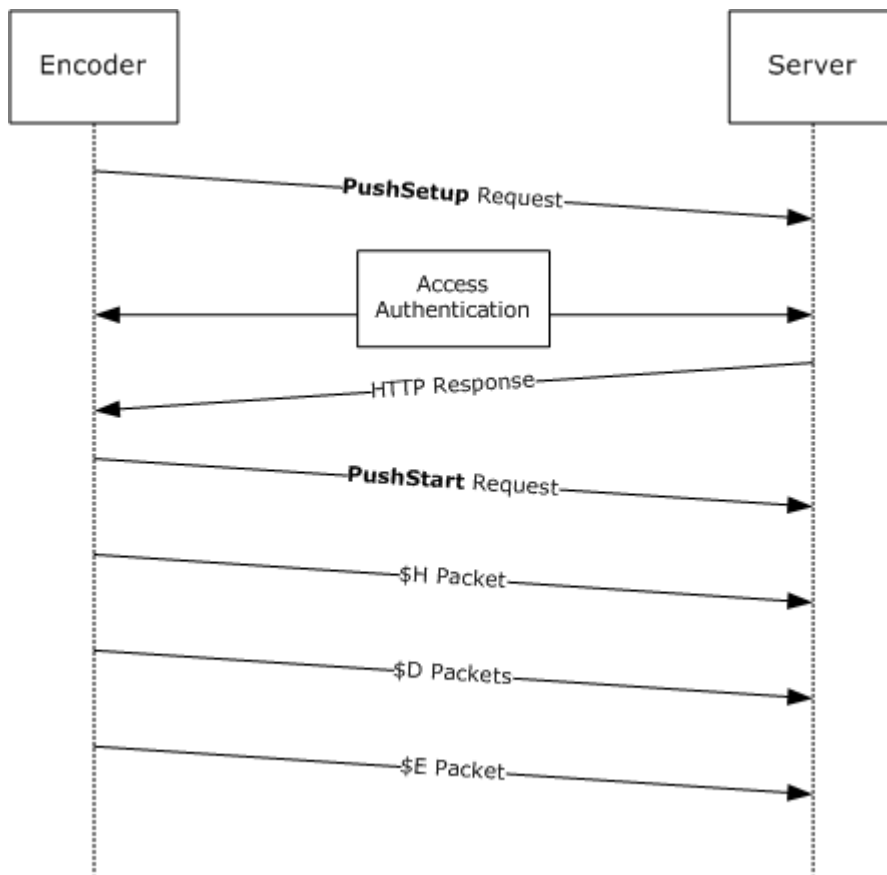


Figure 1: General push distribution sequence with a single PushStart request

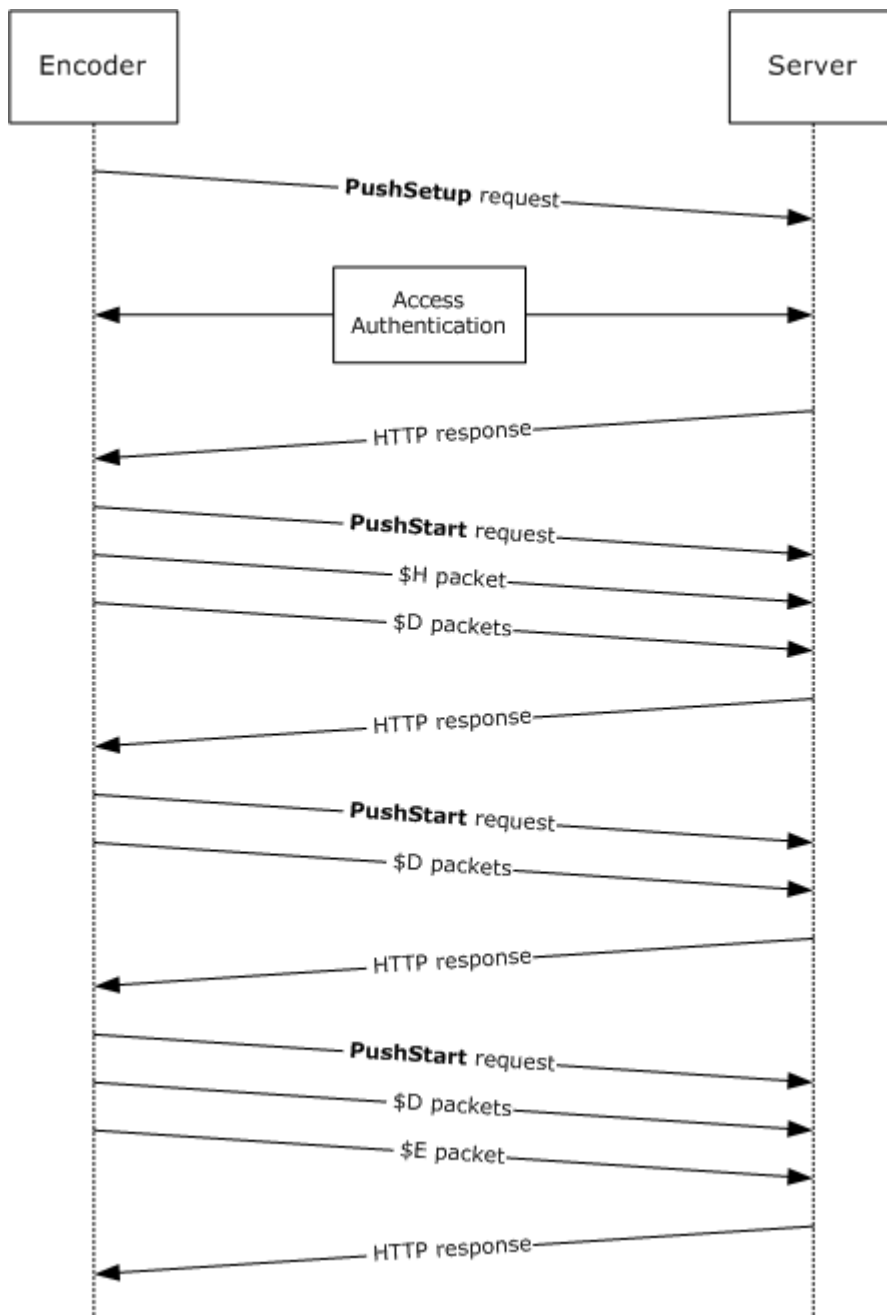


Figure 2: General push distribution sequence with multiple PushStart requests

4.2 Push Distribution with AutoDestroy

The following sequence occurs between a client and a server during a push distribution when the [AutoDestroy \(section 2.2.2.1.2\)](#) directive is specified.

Note Access authentication has been omitted from this example for brevity.

1. The client sends a [PushSetup request](#).
2. The server responds with a "204 No Content" HTTP response.
3. The publishing point specified in the PushSetup request is created by the server.
4. The client sends a [PushStart request](#), followed by an [\\$H \(Header\)](#) packet and [\\$D \(Data\)](#) packets.
5. When the client has sent the amount of data allowed in the PushStart request, the server responds with a "204 No Content" HTTP response. The client then sends a new PushStart request followed by more \$D (Data) packets.
6. After all \$D (Data) packets have been sent to the server, the client sends an [\\$E \(End-of-Stream Notification\)](#) packet to indicate that the data transfer has been completed.
7. The publishing point created in Step 3 is removed by the server.

Figure 3 shows the sequence described above. For simplicity, only a single PushStart request is shown.

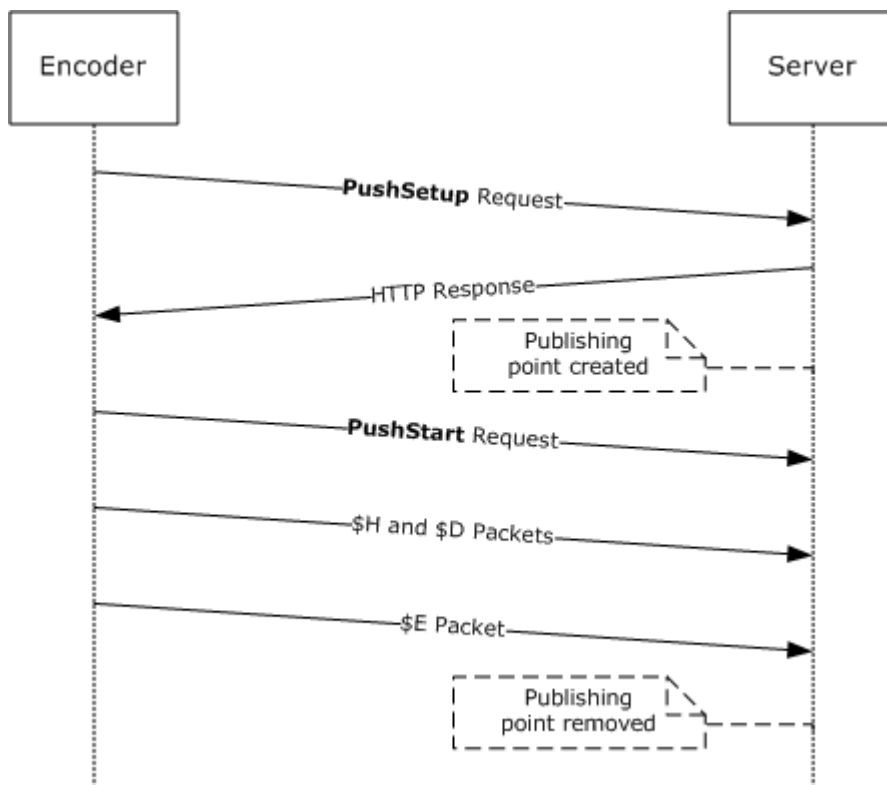


Figure 3: Push distribution with AutoDestroy

4.3 Message Exchange During Push Distribution

The following example illustrates the exchange of messages between a client and a server during a push distribution.

```
POST /pubPoint HTTP/1.1
```

Content-Type: application/x-wms-pushsetup
X-Accept-Authentication: Negotiate, NTLM, Digest
User-Agent: WMEncoder/11.0.5721.5145
Host: MediaServer
Content-Length: 16
Connection: Keep-Alive
Cache-Control: no-cache Cookie:
push-id=0

AutoDestroy: 1

HTTP/1.1 401 Unauthorized
Server: Cougar/9.5.5732.6324
WWW-Authenticate: Negotiate
Date: Mon, 01 Jan 2007 17:45:09 GMT
Pragma: no-cache, timeout=60000
Set-Cookie: push-id=4124758405
Supported: com.microsoft.wm.srvppair, com.microsoft.wm.sswitch,
com.microsoft.wm.predstrm, com.microsoft.wm.fastcache,
com.microsoft.wm.startupprofile
Content-Length: 0

POST /pubPoint HTTP/1.1
Content-Type: application/x-wms-pushsetup
X-Accept-Authentication: Negotiate, NTLM, Digest
User-Agent: WMEncoder/11.0.5721.5145
Host: MediaServer
Content-Length: 0
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: push-id=4124758405
Authorization: Negotiate
TlRMTVNTUAABAAAAB4IIogAAAAAAAAAAAAAAAAAAAAAFASgKAAAADw==

HTTP/1.1 401
Unauthorized Server: Cougar/9.5.5732.6324
WWW-Authenticate: Negotiate
TlRMTVNTUAACAAADgAOADgAAAAFgoqio8hNyf
74ihkAAAAAAAAAAAFgAWABGAAAAABgBkFgAAAA9XAE0AUwAyADIAMAA1AAIADgBXAE0AU
wAyADIAMAA1AAEADgBXAE0AUwAyADIAMAA1AAQADgBXAE0AUwAyADIAMAA1AAMADgBXA
E0AUwAyADIAMAA1AACACABEkqlgclfHAQAAAAA=
Date: Mon, 01 Jan 2007 17:45:16 GMT
Pragma: no-cache, timeout=60000
Set-Cookie: push-id=4124758405
Supported: com.microsoft.wm.srvppair,
com.microsoft.wm.sswitch, com.microsoft.wm.predstrm,
com.microsoft.wm.fastcache, com.microsoft.wm.startupprofile
Content-Length: 0

POST /pubPoint HTTP/1.1
Content-Type: application/x-wms-pushsetup
X-Accept-Authentication: Negotiate, NTLM, Digest
User-Agent: WMEncoder/11.0.5721.5145
Host: MediaServer
Content-Length: 16
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: push-id=4124758405

Authorization: Negotiate
TlRMTVNTUAADAAAAGAYAIIAAAAYABgAmgAAABAAEABIAAAAGgAaAFgAAAAQABAACg
AAAAAAACyAAAABYKIogUBKaoAAAAAPQwBSAE0AVQBSAC0AWAAxAEAAZABtAGkAbgBpA
HMAAdABYAGEAdABvAHIAQwBSAE0AVQBSAC0AWAAxAOxCYicp6dvUAAAAAAAAAAAAAAAA
AAAAAMBTlr8DoynvziDWlWTSjE40MitgYYsXDA==

AutoDestroy: 1

HTTP/1.1 204 No
Content Server: Cougar/9.5.5732.6324
Content-Length: 0
Date: Mon, 01 Jan 2007 17:45:16 GMT
Pragma: no-cache, timeout=60000
Cache-Control: no-cache
Set-Cookie: push-id=4124758405
Supported: com.microsoft.wm.srvppair, com.microsoft.wm.sswitch,
com.microsoft.wm.predstrm, com.microsoft.wm.fastcache,
com.microsoft.wm.startupprofile

POST /pubPoint HTTP/1.1
Content-Type: application/x-wms-pushstart
X-Accept-Authentication: Negotiate, NTLM, Digest
User-Agent: WMEncoder/11.0.5721.5145
Host: MediaServer
Content-Length: 2147483647
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: push-id=4124758405

\$H...

\$D...

5 Security

The following sections specify security considerations for implementers of the Windows Media HTTP Push Distribution Protocol.

5.1 Security Considerations for Implementers

The protocol is vulnerable to a session hijacking attack in which the attacker guesses the value of the [push-id \(section 2.2.1.2.1\)](#) cookie on the [Set-Cookie](#) header and the TCP port number used by the client. This approach works because the attacker makes it appear to the server that the TCP connection to the client has been lost. The attacker then establishes its own TCP connection to the server and sends a request with a [Cookie](#) header by using the victim's push-id value. To mitigate the attack, server implementations should use a good random number generator when creating push-id values. Also, if HTTP access authentication is used, the server should authenticate access at least once on each new URL or TCP connection, or preferably, on each [PushSetup](#) and [PushStart](#) request.

5.2 Index of Security Parameters

Security parameter	Section
HTTP access authentication	As specified in section 2.1

6 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows Server 2003
- Windows Vista
- Windows XP SP2
- Windows XP SP1

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

<1> Section 3.1.4.1.1: Windows Media Format 9 Series SDK, Windows Media Format 9.5 SDK, and Windows Vista list support for NTLM authentication (as specified in [\[MS-NLMP\]](#)) and Digest authentication (as specified in [\[RFC2617\]](#)). Support for SPNEGO-based Kerberos authentication, as specified in [\[RFC4559\]](#), is also listed if this is enabled in Internet Explorer. Although Basic authentication, as specified in [\[RFC2617\]](#), is supported, it is not listed.

<2> Section 3.1.4.2.1: Windows Media Format 9 Series SDK, Windows Media Format 9.5 SDK, and Windows Vista list support for NTLM authentication (as specified in [\[MS-NLMP\]](#)) and Digest authentication (as specified in [\[RFC2617\]](#)). Support for SPNEGO-based Kerberos authentication, as specified in [\[RFC4559\]](#), is also listed if this is enabled in Internet Explorer. Although Basic authentication (as specified in [\[RFC2617\]](#)) is supported, it is not listed.

<3> Section 3.1.5.1: Windows Media Format 9 Series SDK, Windows Media Format 9.5 SDK, and Windows Vista support NTLM authentication (as specified in [\[MS-NLMP\]](#)), Basic authentication (as specified in [\[RFC2617\]](#)), Digest authentication (as specified in [\[RFC2617\]](#)), and SPNEGO-based Kerberos authentication (as specified in [\[RFC4559\]](#)).

<4> Section 3.1.5.2: Windows Media Format 9 Series SDK, Windows Media Format 9.5 SDK, and Windows Vista support NTLM authentication (as specified in [\[MS-NLMP\]](#)), Basic authentication (as specified in [\[RFC2617\]](#)), Digest authentication (as specified in [\[RFC2617\]](#)), and SPNEGO authentication (as specified in [\[RFC4559\]](#)).

<5> Section 3.2.5.1: Windows Media Services will respond to the request with HTTP status code 200, and the message response of the request will contain an ASX file that contains the same URL that the client provided in the request. Because ASX files are understood by Windows Media Player, it is the expectation that the Web client will invoke Windows Media Player to parse the ASX file.

<6> Section 3.2.5.1: Windows Media Services support NTLM authentication (as specified in [\[MS-NLMP\]](#)), Digest authentication (as specified in [\[RFC2617\]](#)), and SPNEGO-based Kerberos authentication (as specified in [\[RFC4559\]](#)). Windows Media Services does not support acting as a proxy server for this protocol.

<7> Section 3.2.5.2: Windows Media Services support NTLM authentication (as specified in [\[MS-NLMP\]](#)), Digest authentication (as specified in [\[RFC2617\]](#)), and SPNEGO-based Kerberos authentication (as specified in [\[RFC4559\]](#)). Windows Media Services does not support acting as a proxy server for this protocol.

7 Index

\$C (Stream Change Notification) ([section 2.2.3.2](#), [section 3.2.5.6](#))
\$D (Data) ([section 2.2.3.3](#), [section 3.2.5.4](#))
\$E (End-of-Stream Notification) ([section 2.2.3.4](#), [section 3.2.5.5](#))
\$F (Filler) ([section 2.2.3.5](#), [section 3.2.5.7](#))
\$H (Header) ([section 2.2.3.6](#), [section 3.2.5.3](#))

A

Abstract data model
[client](#)
[server](#)
[Administrative Disconnect](#)
[Applicability](#)
[application/x-wms-pushsetup](#)
[application/x-wms-pushstart](#)
[ASF header file](#)
[ASF packet](#)
[AutoDestroy](#)

C

[Capability negotiation](#)
Client
[abstract data model](#)
[higher-layer triggered events](#)
[initialization](#)
[local events](#)
[message processing](#)
[overview](#)
[sequencing rules](#)
[timer events](#)
[timers](#)
[Configure the server](#)
[Content-Type](#)
[Cookie](#)

D

Data model - abstract
[client](#)
[server](#)
Disconnect
TCP connection ([section 3.1.7.1](#), [section 3.2.7.1](#))

E

Examples
[general push distribution sequence example](#)
[overview](#)
[push distribution with AutoDestroy example](#)
[sample message exchange during push distribution](#)

F

[Fields - vendor-extensible](#)
[Framing Header](#)

G

[General push distribution sequence example](#)
[Glossary](#)

H

Higher-layer triggered events
[client](#)
[server](#)
[HTTP header fields](#)

I

[Idle-Timeout Timer Expires](#)
[Implementer - security considerations](#)
[Inactivity-Timeout Timer Expires](#)
[Index of security parameters](#)
[Informative references](#)
Initialization
[client](#)
[server](#)
[Introduction](#)

L

[Last packet](#)
Local events
[client](#)
[server](#)

M

Message processing
[client](#)
[server](#)
Messages
[HTTP header fields](#)
[overview](#)
[packet types](#)
[request types](#)
[syntax](#)
[transport](#)

N

[no-cache](#)
[Normative references](#)
Notification
[last packet](#)
[new ASF header file](#)

O

[Overview](#)

P

[Packet types](#)

[Parameters - security index](#)

[Pragma](#)

[Preconditions](#)

[Prerequisites](#)

[Push distribution with AutoDestroy example](#)

[push-id](#)

[PushSetup Request](#) ([section 2.2.2.1](#), [section 3.1.4.1.1](#), [section 3.2.5.1](#))

[PushSetup Response](#)

[PushStart Request](#) ([section 2.2.2.2](#), [section 3.1.4.2.1](#), [section 3.2.5.2](#))

[PushStart Response](#) ([section 3.1.5.2](#), [section 3.2.5.2.1](#))

R

[References](#)

[informative](#)

[normative](#)

[overview](#)

[Relationship to other protocols](#)

[Request types](#)

S

[Sample message exchange during push distribution example](#)

[Security](#)

[implementer considerations](#)

[overview](#)

[parameter index](#)

[Sequencing rules](#)

[client](#)

[server](#)

[Server](#)

[abstract data model](#)

[higher-layer triggered events](#)

[initialization](#)

[local events](#)

[message processing](#)

[overview](#)

[request to configure](#)

[sequencing rules](#)

[timer events](#)

[timers](#)

[Server header](#)

[Set-Cookie](#)

[Standards assignments](#)

[Streaming Content](#)

[Supported](#)

[Syntax](#)

T

[TCP connection](#) ([section 3.1.7.1](#), [section 3.2.7.1](#))

[Template-URL](#)

[timeout](#)

[Timer events](#)

[client](#)

[server](#)

[Timers](#)

[client](#)

[server](#)

[Transport](#)

[Triggered events - higher-layer](#)

[client](#)

[server](#)

U

[User-Agent](#)

V

[Vendor-extensible fields](#)

[Versioning](#)

W

[Windows behavior](#)

X

[X-Accept-Authentication](#)