[MS-XWDNOTIF]: Web Distributed Authoring and Versioning (WebDAV) Extensions for Notifications

---

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
12/03/2008	1.0		Initial availability.
02/04/2009	1.01		Revised and edited technical content.
03/04/2009	1.02		Revised and edited technical content.
04/10/2009	2.0		Deprecated for Exchange 2010.
07/15/2009	3.0	Major	Changes made for template compliance.
11/04/2009	4.0.0	Major	Updated and revised the technical content.
02/10/2010	4.1.0	Minor	Updated the technical content.
05/05/2010	4.2.0	Minor	Updated the technical content.
08/04/2010	4.3	Minor	Clarified the meaning of the technical content.
11/03/2010	4.4	Minor	Clarified the meaning of the technical content.

Contents

1 Introduction

This document specifies the Web Distributed Authoring and Versioning (WebDAV) Extensions for Notifications, which extend the HTTP Extensions for Distributed Authoring – WebDAV protocol, as specified in [RFC4918], to provide **event notification** for content that is contained on a **WebDAV server**.

This specification extends **WebDAV** by introducing new **HTTP** request and response headers that enable clients to subscribe to **events** on a WebDAV server and for the server to notify the client that an event has occurred. This document also specifies three new WebDAV methods and one message that are used to manage and create event notifications.

1.1 Glossary

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

Augmented Backus-Naur Form (ABNF)
endpoint
GUID
Hypertext Transfer Protocol (HTTP)
Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)
mailbox
message
Secure Sockets Layer (SSL)
store
Uniform Resource Identifier (URI)
Uniform Resource Locator (URL)
User Datagram Protocol (UDP)
WebDAV
WebDAV client
WebDAV server
XML

The following terms are specific to this document:

event: Any change in a **WebDAV server** resource.

event notification: A message sent by a **subscribed resource** to its **subscribers** or **implied subscribers**, notifying them of an **event** on that resource.

implied subscriber: A **resource** that did not negotiate a subscription with a subscription server, but will still be notified of events on that server.

resource: Any entity with a **URI** address that participates in this notification protocol.

subscriber: A **resource** that negotiates a subscription with a subscription server.

subscribed resource: A resource that is the subject of a subscription.

top-level hierarchy: A hierarchy of content locations that are based on the **URI** of a **resource**. For example, the resource located at /exchange/resources/folder is contained within the top-level hierarchy /exchange/resources.

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.

[MS-XWDEXT] Microsoft Corporation, "Web Distributed Authoring and Versioning (WebDAV) Core Extensions", April 2009.

[RFC2068] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997, http://www.ietf.org/rfc/rfc2068.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

[RFC2246] Dierks, T., and Allen, C., "The TLS Protocol Version 1.0", RFC 2246, January 1999, http://www.ietf.org/rfc/rfc2246.txt

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

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.ietf.org/rfc/rfc2818.txt

[RFC3986] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005, http://www.ietf.org/rfc/rfc3986.txt

[RFC4918] Dusseault, L., Ed., "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, June 2007, http://www.webdav.org/specs/rfc4918.html

[RFC768] Postel, J., "User Datagram Protocol", RFC 768, August 1980, http://www.ietf.org/rfc/rfc768.txt

1.2.2 Informative References

[MS-OXGLOS] Microsoft Corporation, "Exchange Server Protocols Master Glossary", April 2008.

1.3 Overview

The WebDAV Extensions for Notifications extend the WebDAV protocol to add support for event notifications for content stored on the WebDAV server. Client applications can subscribe to events on the server and then receive notifications from the server when those events occur.

The WebDAV Extensions for Notifications specify the following extensions to the base WebDAV protocol:

- A header that identifies a subscription to event notifications.
- A header that indicates the event that a client is subscribing to.
- A header that specifies the call back address for event notifications.
- A header that specifies a time period for grouping event notifications.

- A header that specifies the life time of the eventsubscription.
- Extensions that enable clients to subscribe and unsubscribe to events.
- An extension that enables the client to poll the WebDAV server for event notices.
- An extension that enables a WebDAV server to notify subscribed clients when an event occurs.

1.4 Relationship to Other Protocols

The WebDAV Extensions for Notifications rely on WebDAV, as defined in [RFC4918], and the WebDAV core extensions, as defined in [MS-XWDEXT]. WebDAV in turn relies on HTTP 1.1, as specified in [RFC2068]. These extensions can use **Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)** for data protection, as specified in [RFC2818]. Server notification of clients relies on the **User Datagram Protocol (UDP)**, as specified in [RFC768].

1.5 Prerequisites/Preconditions

The WebDAV Extensions for Notifications require a WebDAV server, as specified in [RFC4918].

These extensions also require that **WebDAV clients** have **URLs** that point to WebDAV servers, and that client applications that are using the NOTIFY Message have a **Uniform Resource Identifier (URI)** that identifies a UDP **endpoint**.

1.6 Applicability Statement

These extensions are applicable in scenarios that require client applications to receive notifications of events that occur on a WebDAV server.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- **Supported Transports**: These extensions use Hypertext Transfer Protocol (HTTP) as the primary transport. NOTIFY messages use UDP transport.
- **Protocol Versions**: This document introduces no new versioning mechanisms beyond those that already exist in [RFC4918] and in [RFC2616].
- **Capability Negotiation**: This document introduces no new capability negotiations beyond those that already exist in [RFC4918] and [RFC2616] via the **OPTIONS** method.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

The following sections describe transport requirements and the syntax of the WebDAV Extensions for Notifications.

2.1 Transport

Messages are transported by using HTTP, as specified in [RFC4918] and [RFC2068]. NOTIFY messages are transported by using UDP, as specified in [RFC768].

These extensions can be used with **Secure Sockets Layer (SSL)** or Transport Layer Security (TLS), as specified in [RFC2246].

Port 80 is the standard port assignment for HTTP, and port 443 is the standard port assignment for HTTP over SSL or TLS; however, individual implementations can support other ports.

2.2 Message Syntax

The extension headers defined in this document conform to the form and behavior of other custom HTTP headers, as specified in [RFC2068] section 4.2, and are consistent with the WebDAV verbs and headers, as specified in [RFC4918] sections 8 and 9.

This section specifies the following header extensions:

- The Notification-Type header
- The Call-Back header
- The Notification-Delay header
- The Subscription-Lifetime header
- The Subscription-ID header
- The Subscribe-Group header

This section specifies the following methods:

- **SUBSCRIBE**
- **UNSUBSCRIBE**
- **POLL**

This section specifies the following message:

- **NOTIFY**

2.2.1 Notification-Type Header

The Notification-Type header is used by a WebDAV client in a **SUBSCRIBE** request to indicate the events for which the server should notify the client.

The Notification-Type header is used by a WebDAV server in a **SUBSCRIBE** response to indicate the events for which the server will notify the client.

This header extension is defined as follows, using the **Augmented Backus-Naur Form (ABNF)** syntax, as specified in [RFC2068] section 2.1.

```
Notification-Type:= "notification-type:" ntype
ntype = "update" |
   "update/newmember" |
   "delete" |
   "move" |
   "pragma/<http://schemas.microsoft.com/exchange/newmail>"
```

The Notification-Type header MUST appear in a new **SUBSCRIBE** request. The Notification-Type header SHOULD NOT appear in a **SUBSCRIBE** request if the Subscription-ID header is specified to renew a subscription.

The Notification-Type header MUST appear in the **SUBSCRIBE** 200 OK response for a new subscription.

The table in section 3.2.1.1 provides information about the events that can be specified in the Notification-Type header.

2.2.2 Call-Back Header

The Call-Back header is used by a WebDAV client in a **SUBSCRIBE** request to specify the URI to which the WebDAV server sends notifications.

The Call-Back header can appear in a **SUBSCRIBE** request. If the header is present and includes a call-back address, the client has chosen the **NOTIFY** delivery model; the server will send a NOTIFY message, as specified in section 2.2.10, to the specified call-back URI when an event occurs.

If the Call-Back header does not appear in a **SUBSCRIBE** request, the client has chosen the **POLL** delivery model and must poll the server for events by using the POLL method, as specified in section 2.2.9.

This header extension is partially defined as follows, using the ABNF syntax, as specified in [RFC2068] section 2.1. For a complete definition of a URI, see [RFC3986].

```
Call-Back := "Call-Back:" URI
URI := "httpu://" computer_name[ ":" port][ "/" path]
```

The URI MUST begin with "httpu". The port and path elements are optional. If the port is not specified, port 80 MUST be used. The path enables multiple clients to subscribe to the same event without conflict.

The maximum length of the value part of the Call-Back header is 419 characters.

2.2.3 Subscription-Lifetime Header

The Subscription-Lifetime header indicates the minimum length of time that a subscription for an event will be valid. The lifetime is specified in seconds.

The Subscription-Lifetime header can appear in a **SUBSCRIBE** request from a WebDAV client.

If the server does not accept the length of the subscription, the server MUST return the actual subscription lifetime in the response.

This header extension is defined as follows, using the ABNF syntax, as specified in [RFC2068] section 2.1.

```
Subscription-Lifetime := "Subscription-Lifetime:" 1*DIGIT
```

2.2.4 Notification-Delay Header

The Notification-Delay header specifies the delay, expressed in milliseconds, that the server SHOULD use between notifications of events. The delay only applies to the **NOTIFY** delivery model. If the subscription request does not contain a Call-Back header, the server MUST ignore the Notification-Delay header.

The client can include the Notification-Delay header in a subscription request. If the server does not accept the length of the delay, the server SHOULD<1> return the actual delay value that is set by the server in the response.

If the Notification-Delay header is not set in the original SUBSCRIBE request, the server SHOULD use a notification delay of 1000 milliseconds (1 second); if the Notification-Delay is set to a value that is less than 1000 milliseconds, the server SHOULD send the requested value in the response, but MUST use 1000 milliseconds as the delay value.

The client can include a Notification-Delay header in a subscription renewal request. If the server does not support changing the notification delay but will otherwise renew the subscription, the server MUST respond with a 200 (OK) response and SHOULD<2> include a Notification-Delay header with the original delay value.

When multiple subscriptions are made on the same resource with different notification delays, the server SHOULD use the shortest value specified in a Notification-Delay header.

This header extension is defined as follows, using ABNF syntax, as specified in [RFC2068] section 2.1.

```
Notification-Delay := "Notification-Delay:" 1*DIGIT
```

2.2.5 Subscription-ID Header

The Subscription-ID header contains a number assigned by the WebDAV server to uniquely identify a subscription. The WebDAV client uses this number to identify the subscription when using the **POLL** and **UNSUBSCRIBE** methods, and when using the **SUBSCRIBE** method to renew a subscription. The WebDAV server uses this number to identify the subscription when using the NOTIFY message and in response messages.

A WebDAV client can have multiple subscriptions on the same **resource** for the same event; these will be distinguished only by the Subscription-ID header. The WebDAV server always includes the Subscription-ID header so that clients can identify a specific subscription.

This header extension is defined as follows, using ABNF syntax, as specified in [RFC2068] section 2.1.

```
Subscription-ID:= "Subscription-ID:" 1*DIGIT 0*["," 1*DIGIT]
```

Multiple subscription IDs can be specified in a **SUBSCRIBE**, **POLL**, or **UNSUBSCRIBE** request. The IDs are separated by commas, as follows:

```
Subscription-ID: 1, 18, 243
```

The Subscription-ID header is used as follows:

- **SUBSCRIBE** method: The Subscription-ID header can be present. If the header is present and the subscription IDs match the specified content location, the server will renew the subscriptions. The client SHOULD NOT send the Depth header, Notification-Type header, or Subscription-Lifetime header when renewing the subscription.
- 200 Response to **SUBSCRIBE**: The WebDAV server MUST include the subscription IDs of the requested subscriptions in the body of the response message.
- 412 Response to **SUBSCRIBE**: When the **SUBSCRIBE** request includes any invalid subscription IDs, the WebDAV server responds with a 207 multi-status response and MUST include the invalid IDs in a 412 status element within the **XML** body of the response.
- **UNSUBSCRIBE** method: The Subscription-ID MUST be present. If the header is present and the subscription IDs match the specified content location, the server will cancel the subscription.
- 207 Response to **UNSUBSCRIBE**: The server MUST include the subscription identifiers of the canceled subscription in the XML body of the response.
- 412 Response to **UNSUBSCRIBE**: When the **UNSUBSCRIBE** request includes only invalid subscription IDs, the WebDAV server responds with a 207 multi-status response and MUST include the invalid IDs in a 412 status element within the XML body of the response.
- **POLL** method: The Subscription-ID header MUST be present.
- 207 Response to **POLL**: The WebDAV server returns a multi-status response body with 200 OK status elements for subscriptions on which events occurred. The server returns a 204 No Content status element for subscriptions on which events have not occurred.
- 412 Response to **POLL**: When the **POLL** request includes only invalid subscription IDs, the WebDAV server responds with a 207 multi-status response and MUST include the invalid IDs in a 412 status element within the XML body of the response.
- NOTIFY message: The Subscription-ID header MUST be present. The header will include all subscription IDs that are at the same content location on which events happened.

2.2.6 Subscribe-Group Header

The Subscribe-Group header contains a base64-encoded **GUID** that is assigned by the WebDAV server to uniquely identify a **message store**. The Subscribe-Group header is included in **SUBSCRIBE** response messages and NOTIFY messages.

The server MUST include the Subscribe-Group header in the NOTIFY message.

This header extension is defined as follows, using ABNF syntax, as specified in [RFC2068] section 2.1.

```
Subscribe-group := "Subscribe-group:" subscribe-group-spec
Subscribe-group-spec := 24*24VCHAR
```

2.2.7 SUBSCRIBE Method

WebDAV clients use the **SUBSCRIBE** method to subscribe to and renew notifications for events that occur on the WebDAV server. This method is used to specify the details of the monitored event: where to look for it, how long to monitor the event, what the notification mechanism is, and how long to delay before generating the notification of the event. If the **SUBSCRIBE** method specifies an existing notification subscription, the subscription is renewed.

Clients can subscribe to the following events:

- New mail (pragma/<http://schemas.microsoft.com/exchange/newmail)
- Object created (update/newmember)
- Object deleted (delete)
- Object modified (update)
- Object moved (move)

The client includes a Notification-Type header to indicate the type of event to which it is subscribing, and possibly includes a Call-Back header to specify the URI of the UDP listener to which the WebDAV server sends event notifications.

If the Subscription-ID header is present, the **SUBSCRIBE** method call is a subscription renewal; the Notification-Type header MUST NOT be present. If the Subscription-ID header appears with the Notification-Type header, it is a bad request and the server MUST refuse the request. When the Subscription-ID header is present, the Notification-Delay, Depth, and Call-Back headers SHOULD be ignored.

The **SUBSCRIBE** request returns one of the WebDAV protocol status codes that are listed in the following table. This list is not comprehensive. For details about the 500-level status codes, see [RFC4918].

Status code	Meaning
200 (OK)	The subscription was successful. The server might have changed some of the parameters.
207 (Multi-Status)	Multiple response codes are contained in the XML body.
400 (Bad Request)	One or more included headers were invalid.
401 (Unauthorized)	User does not have permission to subscribe to this resource.
403 (Forbidden)	A subscription resource cannot be created by using the specified target URL.
412 (Precondition Failed)	The subscription ID(s) in the header did not match the specified resource.
501 (Not Implemented)	The server does not support the specified notification method.

2.2.8 UNSUBSCRIBE Method

WebDAV clients use the **UNSUBSCRIBE** method to remove the subscription to an event notification that occurs on the WebDAV server. The client includes a Subscription-ID header in the **UNSUBSCRIBE** request to indicate which event notification has to be removed.

The **UNSUBSCRIBE** request returns one of the WebDAV protocol status codes that are listed in the following table. This list is not comprehensive; for details about the 500-level status codes, see [RFC4918].

Status Code	Meaning
200 (OK)	The subscription was canceled.
207 (Multi-Status)	Multiple response codes are contained in the XML body.
400 (Bad Request)	An illegal combination of headers was included in the request.
412 (Precondition Failed)	A subscription ID that was specified in the UNSUBSCRIBE request was not valid.

2.2.9 POLL Method

WebDAV clients use the **POLL** method to inquire about events that have occurred on the WebDAV server. The client includes a Subscription-ID header that contains one or more event subscriptions that it is querying, and the server sends a multi-status response that contains the identifiers of all subscribed events that have occurred since the last **POLL** request.

The URI that is used in the **POLL** method request that is sent by the client MUST be the one that is provided in the Content-Location header or fall within the **top-level hierarchy** of the **URI** in the **POLL** request and it MUST match the Subscription-ID header that is sent in the same request. If more than one Subscription-ID header is used, they MUST be valid for all resources within the same top-level hierarchy. When the client uses the **POLL** method, the server MUST renew the subscription for all polled subscription IDs.

The **POLL** method can be used with or without NOTIFY messages being sent from server to client.

The **POLL** request returns one of the WebDAV protocol status codes that are listed in the following table. This list is not comprehensive; for details about the 500-level status codes, see [RFC4918].

Status Code	Meaning
200 (OK)	Successful poll, events occurred since the last **POLL** request on the specified subscriptions. This response code occurs only within a 207 (Multi-Status) response body.
204 (No Content)	Successful poll, but no events occurred since the last **POLL** request on the specified subscriptions. This response code occurs only within a 207 (Multi-Status) response body.
207 (Multi-Status)	Status codes appear in the XML body for the polled subscriptions.
401 (Unauthorized)	The user does not have access permissions or authorization to poll this resource.
404 (Not Found)	Resource was not found.
406 (Non Acceptable)	The **POLL** request contained an Accept header that could not be satisfied by the server. This response code can appear outside of a 207 (Multi-Status) response.
412 (Precondition Failed)	A subscription ID specified in the **POLL** request did not match the resource named. The subscription ID(s) that did not match are included as a list within the XML body of the status response.

2.2.10 NOTIFY Message

WebDAV servers send the NOTIFY message to notify subscribed clients that an event has occurred on the subscribed event. The NOTIFY message is sent to the URI that is specified in the Call-Back header in the associated **SUBSCRIBE** request, and MUST include the Subscription-ID header that indicates the event subscription that triggered the notification.

Because UDP packets might be lost, the server MUST keep track of events that have subscriptions. The server MUST maintain subscriptions for which events occurred until the client sends an acknowledgement by using the Subscription-ID header in any method request.

The server MUST continue to send the NOTIFY messages until either the subscription expires or the client sends an acknowledgement. The interval for the NOTIFY messages is based on the Notification-Delay header. <3>

3 Protocol Details

As specified in [RFC4918], the WebDAV protocol operates between an initiator (a WebDAV client) and a responder (a WebDAV server). In this section the client and server behaviors for the WebDAV Extensions for Notifications are specified.

3.1 Abstract Data Model

A server that provides event notifications MUST maintain a list of **subscribers**, keeping for each subscriber the information listed in the following table.

Information	Description
Unique subscription identifier	Required. MUST be unique over the lifetime of the subscription. Generated by the server in response to a subscription message.
Delivery URI for event messages	Optional. The URI of a UDP endpoint for event notifications.
Subscription duration	Required. The amount of time in seconds, or the duration until the subscription expires.

The server SHOULD accept as many subscriptions as it can reasonably maintain and deliver.

The list of subscribers is updated via subscription, renewal, cancellation, and expiration. Requests that affect the subscription list are specified in section 3.2.

3.2 Subscription Details

Clients send subscription requests to the server to subscribe to event notifications. The client can either provide a Call-Back header to specify a UDP endpoint for notifications, or leave out the Call-Back header and use polling to determine when an event has occurred. The following sections provide details about creating, renewing, and canceling subscriptions.

3.2.1 Subscribing

The following sections provide details for subscribing to server events.

3.2.1.1 Subscription Types

The following table provides information about the events to which the client can subscribe. The events are specified in the Notification-Type header.

The Depth column indicates the extent of the subscription. The values in the Depth column are interpreted as follows:

- 0 – The subscription is for the entity only.
- 1 – The subscription is for the entity and any items contained within that entity.
- infinity – The subscription is for the entity and all items contained within that entity or within the children of that entity. The value infinity is only valid on new mail subscriptions on the root **mailbox**.

Event type	Target	Depth	Store event	Description
delete	Any	0	Delete	The entity was deleted.
delete	Collection	1	Delete	The collection or any member of the collection up to depth 1 was deleted.
move	Any	0	Move	The entity was moved.
move	Collection	1	Move	The collection or any member of the collection up to depth 1 was moved.
pragma/<http://schemas.microsoft.com/exchange/newmail>	Mailbox or Collection	infinity	NewMail	Special new mail update.
update	Non-collection entity	0	Property update or body change.	The content of the entity was changed.
update	Collection	0	Property update	The properties on a collection were modified.
update	Collection	1	Create, Modify, Delete, Move, Property update, Copy	An entity (collection or non-collection) was created in, copied to, moved to or from, or deleted from the collection; or the

Event type	Target	Depth	Store event	Description
				collection's properties were updated.
update/newmember	Any	0	None	Not valid – server MUST return a 400 (Bad Request) error response.
update/newmember	Collection	1	Create, Move, Copy	An entity (collection or non-collection) was created in, moved to, or copied to the collection.

3.2.1.2 Request

To subscribe to an event, the client MUST send a request with method **SUBSCRIBE** by using the following format.

```
SUBSCRIBE publisher path HTTP/1.1
Host: server:port
Notification-Type: one of the Notification-Type header values
Call-Back: UDP endpoint
Subscription-Lifetime: lifetime in seconds
Notification-Delay: delay in milliseconds
```

The following table provides details about the request line and headers shown in the previous example.

Item	Description
SUBSCRIBE	The method used to initiate or renew a subscription.
publisher path	The path component of the content to subscribe. A single, relative URL.
http/1.1	HTTP version.
Host	Required. Domain name or IP address and optional port components of the server URL. If the port is missing, port 80 is assumed.
Notification-Type	Required. One of the Notification-Type header values that indicate the event that the client wants notifications for.

Item	Description
Call-Back	Optional. The UDP endpoint that receives notification messages when an event occurs on the server. If the Call-Back header is not included in the subscription request, the client must poll the server for events.
Subscription-Lifetime	Optional. Requested duration of the subscription expressed in seconds.
Notification-Delay	Optional. When a notification delay is specified, the server treats any number of events that occur during the specified time period as a single event. Expressed in milliseconds.

3.2.1.3 Response

If there are enough resources to maintain the subscription, the server SHOULD accept it. To accept the subscription, the server MUST send a response in the following format.

```
HTTP/1.1 200 OK
Date: when the response was generated
Server: server/version
Notification-Type: one of the notification types
Subscription-Lifetime: lifetime in seconds
Notification-Delay: delay in milliseconds
Call-Back: UDP endpoint
Content-Location: fully-qualified content location
Subscribe-group: unique message store identifier
Subscription-ID: unique subscription identifier
```

The following table provides details about the request line and headers shown in the previous example.

Item	Description
Date	Required. The date and time at which the response was generated.
Server	Optional. Server name and version of the server that generated the response.
Notification-Type	Required. One of the Notification-Type header values that indicate the event that the client wants notifications for. This MUST be the same as the value specified in the request.
Subscribe-group	Required. Unique identifier of the message store where the subscription was made.
Subscription-Lifetime	Required. Duration of the subscription expressed in seconds. Can be different than the lifetime requested by the client.
Notification-Delay	Optional, required when the client includes the Notification-Delay header. Can be different than the notification delay requested by the client.
Call-Back	Optional. The UDP endpoint that receives notification messages when an event occurs on the server. If the Call-Back header is not included in the subscription request, the server MUST NOT include the Call-Back header. If specified, it MUST be the same as the value specified in the request.
Content-Location	Required. Fully qualified URI that describes the content location for which events will be received.

Item	Description
Subscription-ID	Required. Unique identifier assigned to the subscription by the server. MUST be unique within the subscription lifetime.

3.2.1.4 Errors

If the server cannot accept the subscription, it MUST respond with one of the error codes that are listed in the following table.

Status code	Meaning
400 (Bad Request)	One or more included headers were invalid.
401 (Unauthorized)	User does not have permission to subscribe to this resource.
403 (Forbidden)	A subscription resource cannot be created by using the specified target URL.
415 (Unsupported Media Type)	The request type of the body is not supported by the server.
501 (Not Implemented)	The server does not support the specified notification method.

3.2.2 Renewing a Subscription

Clients MUST renew a subscription before the duration expires; otherwise, the server MUST remove the subscription from the subscription list.

A subscription is renewed each time a client sends a **POLL** request for the subscription. If the client is using UDP notification, it can renew the subscription by sending a **SUBSCRIBE** request.

3.2.2.1 Request

The following request format is used to renew a subscription by using method **SUBSCRIBE**.

```
SUBSCRIBE publisher path HTTP/1.1
Host: server:port
Subscription-ID: unique subscription identifier
```

The following table lists the details for the request line and headers that appear in the previous example.

Item	Description
SUBSCRIBE	The method to initiate or renew a subscription.
publisher path	The path component of the content to renew. A single, relative URL.
HTTP/1.1	HTTP version.
Host	Required. Domain name or IP address and optional port components of the server URL. If the port is missing, port 80 is assumed.
Subscription-	Required. The unique identifier of the subscription to renew. The subscription ID MUST

Item	Description
ID	be a subscription for events on the specified publisher path.

3.2.2.2 Response

To accept a renewal, the server MUST send a response that contains the subscription ID of the renewed subscription.

3.2.2.3 Errors

If the server cannot accept the subscription renewal, it MUST respond with one of the error codes that are listed in the following table.

Status code	Meaning
207 (Multi-Status)	Multiple response codes are contained in the XML body.
400 (Bad Request)	One or more included headers were invalid.
401 (Unauthorized)	User does not have permission to subscribe to this resource.
403 (Forbidden)	A subscription resource cannot be created by using the specified target URL.
412 (Precondition Failed)	The subscription ID(s) in the header did not match the specified resource.
415 (Unsupported Media Type)	The request type of the body is not supported by the server.
501 (Not Implemented)	The server does not support the specified notification method.

3.2.3 Canceling a Subscription

When a client no longer needs to receive notifications from an event, the client SHOULD cancel the subscription.

3.2.3.1 Request

To cancel a subscription, the client SHOULD send a request with method **UNSUBSCRIBE** by using the following format.

```
UNSUBSCRIBE publisher path HTTP/1.1
Host: server:port
Subscription-ID: subscription identifier
```

The following table lists the details for the request line and headers that appear in the previous example.

Item	Description
UNSUBSCRIBE	The method to cancel the subscription for an event.

Item	Description
publisher path	The path component of the content to unsubscribe. A single, relative URL.
HTTP/1.1	HTTP version.
Host	Required. Domain name or IP address and optional port components of the server URL. If the port is missing, port 80 is assumed.
Subscription-ID	Required. The subscription identifier of the subscription to cancel.

3.2.3.2 Response

To cancel the subscription, the server MUST respond in the following format.

```
HTTP/1.1 207 Multi-Status
Content-Type: text/xml
Content-Length: response length

<?xml version="1.0"?>
<a:multistatus xmlns:b="http://schemas.microsoft.com/Exchange/" xmlns:a="DAV:">
  <a:response>
    <a:href>fully qualified path to content</a:href>
    <a:status>HTTP/1.1 Status Code Message</a:status>
    <b:subscriptionID>
      <li>subscription identifier</li>
      <li>additional subscription identifiers</li>
    </b:subscriptionID>
  </a:response>
  <a:response>
    <a:href>fully qualified path to content</a:href>
    <a:status>HTTP/1.1 Status Code Message</a:status>
    <b:subscriptionID>
      <li>subscription identifier</li>
      <li>additional subscription identifiers</li>
    </b:subscriptionID>
  </a:response>
</a:multistatus>
```

3.2.3.3 Errors

If the server cannot respond to the **UNSUBSCRIBE** request, it MUST respond with one of the errors status codes that are listed in the following table.

Status Code	Meaning
207 (Multi-Status)	Multiple response codes are contained in the XML body.
400 (Bad Request)	An illegal combination of headers was included or the Subscription-ID header was invalid.

3.3 Notification Details

Clients subscribe to events to receive notifications when those events occur. If the client includes a Call-Back header in the subscription request, the client is requesting that the server send notifications via UDP to the specified call back URI. If the Call-Back header is not included, the client will use the **POLL** method to request information about events that have been raised. This section describes the two notification methods in detail.

If a client subscribes to events on a folder and the folder is later deleted, the subscription remains valid for **POLL** requests until the first **POLL** request is sent; subsequent **POLL** requests will return a 412 (Precondition Failed) response. Clients using UDP notification will receive NOTIFY events on the subscription until the first acknowledgement is sent to the server, and then NOTIFY events will cease.

If the pragma/<http://schemas.microsoft.com/exchange/newmail> event is subscribed on a public folder and a message is posted to that folder, the event will not be raised.

3.3.1 Poll Notification

Clients that do not include a Call-Back header in the subscription request MUST send a poll request to receive notification of events. When a client sends a poll request, the subscription is also renewed.

3.3.1.1 Request

To poll the server for events that have occurred since the last poll request, the client MUST send a request with method **POLL** by using the following format:

```
POLL publisher path HTTP/1.1
Host: server:port
Subscription-ID: one or more subscription identifiers
```

The following table lists the details for the request line and headers that appear in the previous example.

Item	Description
POLL	The method to poll the server for notification of events.
publisher path	The path component of the content to poll. A single, relative URL.
HTTP/1.1	HTTP version.
Host	Required. Domain name or IP address and optional port components of the server URL. If the port is missing, port 80 is assumed.
Subscription-ID	Required. One or more subscription identifiers. If more than one subscription identifier is included, all of the subscription identifiers must be for subscriptions to the content specified in the publisher path.

3.3.1.2 Response

To respond to the poll request, the server MUST return the following response.

```
HTTP/1.1 207 Multi-Status
Content-Type: text/xml
Content-Length: response length

<?xml version="1.0"?>
<a:multistatus xmlns:b="http://schemas.microsoft.com/Exchange/" xmlns:a="DAV:">
  <a:response>
    <a:href>fully qualified path to content</a:href>
    <a:status>HTTP/1.1 Status Code Message</a:status>
    <b:subscriptionID>
      <li>subscription identifier</li>
      <li>additional subscription identifiers</li>
    </b:subscriptionID>
  </a:response>
  <a:response>
    <a:href>fully qualified path to content</a:href>
    <a:status>HTTP/1.1 Status Code Message</a:status>
    <b:subscriptionID>
      <li>subscription identifier</li>
      <li>additional subscription identifiers</li>
    </b:subscriptionID>
  </a:response>
</a:multistatus>
```

The following table lists the details for the content in the previous example.

Element	Description
<response>	The server MUST include one **<response>** element for each status code returned for the subscription identifiers in the poll request.
<subscriptionID>	The server MUST include a ** ** element for each subscription identifier in the poll request that returns the associated HTTP status code.

3.3.1.3 Errors

If the server cannot respond to the poll request, it MUST respond with one of errors status codes that are listed in the following table.

Status Code	Meaning
204 (No Content)	Successful poll, but no events occurred since the last **POLL** request on the specified subscriptions. This response code occurs only within a 207 (Multi-Status) response body.
207 (Multi-Status)	Status codes appear in the XML body for the polled subscriptions.
401 (Unauthorized)	The user does not have access permissions or authorization to poll this resource.
404 (Not Found)	Resource was not found. This response code only occurs within a 207 (Multi-Status) response body.
412 (Precondition Failed)	A subscription ID specified in the **POLL** request did not match the resource named. The subscription ID(s) that did not match are returned in the body of the 412 response.

3.3.2 UDP notification

When a client includes a Call-Back header in the initial subscription request, the server MUST notify the client of events on the subscription by sending a UDP datagram to the URI that is specified in the Call-Back header. The UDP datagram MUST be in the following format:

```
NOTIFY UDP call back URI HTTP/1.1
Subscribe-group: unique message store identifier
Subscription-ID: subscription identifier
```

The following table lists the details for the request line and headers that appear in the previous example.

Item	Description
NOTIFY	The method to notify the client that a subscribed event occurred.
UDP call back URI	The URI specified in the Call-Back header that is sent with the subscription request.
HTTP/1.1	HTTP version.
Subscribe-group	Required. Unique identifier of the message store where the subscription was made.
Subscription-ID	Required. The subscription identifier of the event subscription that raised the event.

4 Protocol Examples

4.1 Subscribe Example

4.1.1 Subscribing to Notifications on a Resource

The following example creates a subscription on a resource. The Call-Back header is used in this subscription to instruct the server to send a NOTIFY message for the UDP server at www.fourthcoffee.com by using port 8080.

4.1.1.1 Request

```
SUBSCRIBE /public/subtest HTTP/1.1
Host: www.contoso.com
Notification-type: Update
Call-Back: httpu://www.fourthcoffee.com:8080/510
```

4.1.1.2 Response

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Fri, 06 Jul 2001 18:37:44 GMT
Notification-type: Update
Subscription-lifetime: 3600
Call-Back: httpu://www.fourthcoffee.com:8080/510
Content-Location: http://www.contoso.com/public/subtest/
Content-Length: 0
Subscribe-group: sRKxaABd3kGPV0chHGtCpw==
Subscription-id: 3
```

4.1.2 Setting the Subscription Lifetime

The following example creates a subscription on a resource that expires after 600 seconds. No Call-Back header is specified, so the client must use the **POLL** method to poll for notifications.

4.1.2.1 Request

```
SUBSCRIBE /public/subtest HTTP/1.1
Host: www.contoso.com
Notification-type: Update
Subscription-Lifetime: 600
```

4.1.2.2 Response

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Fri, 06 Jul 2001 18:3928 GMT
Notification-Type: Update
Subscription-Lifetime: 600
Content-Location: http://www.contoso.com/public/subtest/
Content-Length: 0
Subscribe-group: sRKxaABd3kGPV0chHGtCpw==
```

```
Subscription-ID: 4
```

4.1.3 Setting the Notification Delay

The following example creates a subscription on a resource and sets the notification delay to 4 seconds. This means that any number of events that occur during the 4 second window will result in only one notification. Because a Call-Back header is specified, the server will call the NOTIFY message for the UDP server specified in the Call-Back header.

4.1.3.1 Request

```
SUBSCRIBE /public/subtest HTTP/1.1
Host: www.contoso.com
Notification-Type: Update
Notification-Delay: 4000
Call-Back: httpu://www.fourthcoffee.com:8080/510
```

4.1.3.2 Response

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Fri, 06 Jul 2001 18:43:40 GMT
Notification-Type: Update
Subscription-Lifetime: 3600
Call-Back: httpu://www.fourthcoffee.com:8080/510
Content-Location: http://www.contoso.com/public/subtest/
Content-Length: 0
Subscribe-group: sRKxaABd3kGPV0chHGtCpw==
Subscription-ID: 7
Notification-Delay: 4000
```

4.1.4 Renewing a Subscription

The following example renews a previously made subscription.

4.1.4.1 Request

```
SUBSCRIBE /public/subtest HTTP/1.1
Host: www.contoso.com
Subscription-ID: 21
```

4.1.4.2 Response

```
HTTP/1.1 207 Multi-Status
Server: Microsoft-IIS/5.0
Date: Fri, 06 Jul 2001 18:45:37 GMT
Content-Type: text/xml
Content-Length: 272

<?xml version="1.0"?>
<a:multistatus xmlns:b="http://schemas.microsoft.com/Exchange/" xmlns:a="DAV:">
  <a:response>
```

```
    <a:href>http://www.contoso.com/public/subtest</a:href>
    <a:status>HTTP/1.1 200 OK</a:status>
    <b:subscriptionID>
      <li>21</li>
    </b:subscriptionID>
  </a:response>
</a:multistatus>
```

4.2 Unsubscribe Example

The following example shows a client terminating a subscription.

4.2.1 Request

```
UNSUBSCRIBE /public/subtest HTTP/1.1
Host: www.contoso.com
Subscription-ID: 16
```

4.2.2 Response

```
HTTP/1.1 207 Multi-Status
Server: Microsoft-IIS/5.0
Date: Fri, 06 Jul 2001 22:03:15 GMT
Content-Type: text/xml
Content-Length: 273

<?xml version="1.0"?>
<a:multistatus xmlns:b="http://schemas.microsoft.com/Exchange/"xmlns:a="DAV:">
  <a:response>
    <a:href>http://www.contoso.com/public/subtest</a:href>
    <a:status>HTTP/1.1 200 OK</a:status>
    <b:subscriptionID>
      <li>16</li>
    </b:subscriptionID>
  </a:response>
</a:multistatus>
```

4.3 Notify Example

The following example is packet dump of a UDP message from a WebDAV server. The URI used is the one specified in the Call-Back header of the **SUBSCRIBE** method request that is used to create the subscription. The Subscription-ID header contains the identifier of the event.

```
4e 4f 54 49 46 59 20 68    74 74 70 75 3a 2f 2f 66      NOTIFY httpu://f
65 61 6d 64 31 30 3a 38    30 38 30 20 48 54 54 50      eamd10:8080 HTTP
2f 31 2e 31 0d 0a 53 75    62 73 63 72 69 62 65 2d      /1.1..Subscribe-
67 72 6f 75 70 3a 20 4a    71 59 59 33 55 46 6e 70      group: JqYY3UFnp
45 53 4f 59 36 31 38 43    30 36 72 4a 41 3d 3d 0d      ESOY618C06rJA==.
0a 53 75 62 73 63 72 69    70 74 69 6f 6e 2d 69 64      .Subscription-id
3a 20 36 0d 0a 0d 0a 00                                 : 6.....
```

4.4 Poll Example

The following example is a **POLL** request that queries subscriptions with the specified IDs. The response includes an event for subscription ID 11 but not for the other subscription IDs.

4.4.1 Request

```
POLL /public/subtest/ HTTP/1.1
Host: www.contoso.com
Subscription-ID: 8,9,10,11,12
Content-Length: 0
```

4.4.2 Response

```
HTTP/1.1 207 Multi-Status
Content-Type: text/xml
Content-Length: 489

<?xml version="1.0"?>
<a:multistatus xmlns:b="http://schemas.microsoft.com/Exchange/" xmlns:a="DAV:">
  <a:response>
    <a:href>http://www.contoso.com/public/subtest</a:href>
    <a:status>HTTP/1.1 200 OK</a:status>
    <b:subscriptionID>
      <li>11</li>
    </b:subscriptionID>
  </a:response>
  <a:response>
    <a:href>http://www.contoso.com/public/subtest</a:href>
    <a:status>HTTP/1.1 204 No Content</a:status>
    <b:subscriptionID>
      <li>8</li>
      <li>9</li>
      <li>10</li>
      <li>12</li>
    </b:subscriptionID>
  </a:response>
</a:multistatus>
```

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products:

- Microsoft® Exchange Server 2003
- Microsoft® Exchange Server 2007

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

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

<1> Section 2.2.4: When the Exchange server receives a Notification-Delay header with the delay value set to less than the minimum allowed value (1000 milliseconds), the default value is used; however, the actual delay value that is set by the server is not returned in the response.

<2> Section 2.2.4: The Exchange server does not support changing the notification delay in a renewal request. If present, the Notification-Delay header is ignored.

<3> Section 2.2.10: When Exchange 2003 and Exchange 2007 send a NOTIFY message, the server will wait the interval specified in the Notification-Delay header for an acknowledgement before sending a second NOTIFY message. The interval between NOTIFY messages doubles between each message sent for a particular subscription. For example, when the notification delay is set to 3000 milliseconds (3 seconds), NOTIFY messages will be sent out at the following intervals:

Time

0 seconds Event occurs.

3 seconds NOTIFY message #1

9 seconds NOTIFY message #2 (6 seconds after message #1)

21 seconds NOTIFY message #3 (12 seconds after message #2)

45 seconds NOTIFY message #4 (24 seconds after message #3)

7 Change Tracking

This section identifies changes that were made to the [MS-XWDNOTIF] protocol document between the August 2010 and November 2010 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

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

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

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

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

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

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

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

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

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

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

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

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

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change Type
1.1 Glossary	56820 Moved "User Datagram Protocol (UDP)" to list of terms defined in [MS-OXGLOS].	N	Content updated.
2.2.10 NOTIFY Message	57154 Revised behavior note for the interval between NOTIFY messages.	N	Product behavior note updated.

8 Index