[MS-XWDREPL]:
Web Distributed Authoring and Versioning (WebDAV) Extensions for Replication

---

Revision Summary

Date	Revision History	Revision Class	Comments
12/03/2008	1.0		Initial Release.
03/04/2009	1.01		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	5.0.0	Major	Updated and revised the technical content.
05/05/2010	5.1.0	Minor	Updated the technical content.
08/04/2010	5.1.0	No change	No changes to the meaning, language, or formatting of the technical content.
11/03/2010	5.1.0	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

This document specifies the client-server replication of Web resources on a **WebDAV server** by means of the extension of the **HTTP** and **WebDAV** protocols.

1.1 Glossary

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

Augmented Backus-Naur Form (ABNF)
binary large object (BLOB)
Hypertext Transfer Protocol (HTTP)
Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)
message
property
Secure Sockets Layer (SSL)
Uniform Resource Identifier (URI)
Uniform Resource Locator (URL)
universal unique identifier (UUID)
WebDAV
WebDAV client
WebDAV server
XML

The following terms are specific to this document:

paged results: A data model that allows a client to request that the server return a subset of the result set rather than the entire set.

optimistic concurrency: A model for updating data in a database that does not lock records and allows for improved performance.

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.

[MS-XWDMAIL] Microsoft Corporation, "Web Distributed Authoring and Versioning (WebDAV) Extensions for E-Mail Support", December 2008.

[MS-XWDSEARCH] Microsoft Corporation, "Web Distributed Authoring and Versioning (WebDAV) Extensions for Search", December 2008.

[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

[RFC4234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005, http://www.ietf.org/rfc/rfc4234.txt

[RFC2518] Goland Y., Whitehead, E., Faizi, A., et al., "HTTP Extensions for Distributed Authoring -- WEBDAV", RFC 2518, February 1999, http://www.ietf.org/rfc/rfc2518.txt

1.2.2 Informative References

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

1.3 Overview

The WebDAV Extensions for Replication are a set of methods, headers, and elements that extend the Hypertext Transport Protocol – HTTP/1.1, as specified in [RFC2616]. The WebDAV protocol allows for the writing of data to Internet servers.

WebDAV replication is applied over the existing WebDAV operations that allow clients to do the following:

- Determine what has changed in a given collection.
- Update items by using **optimistic concurrency**.
- Locate and resolve conflicted items.

1.4 Relationship to Other Protocols

The WebDAV Extensions for Replication rely on WebDAV, as specified in [RFC2518], which in turn relies on HTTP 1.1, as specified in [RFC2616]. These extensions can use **HTTPS** for data protection, as specified in [RFC2818].

The WebDAV Extensions for Replication are also dependent on the client, server, and Microsoft extensions to [RFC2518], as specified in [MS-XWDEXT].

1.5 Prerequisites/Preconditions

The WebDAV Extensions for Replication require a WebDAV server, as specified in [RFC2518]. These extensions also require that **WebDAV clients** have **URLs** that point to WebDAV servers.

1.6 Applicability Statement

This protocol is applicable in scenarios that require client applications to synchronize data on a WebDAV server.

1.7 Versioning and Capability Negotiation

Clients can determine whether a server supports the replication extensions by sending an **OPTIONS** command, as specified in [RFC2616], to the server and examining the response. For a server to declare that it implements replication, it has to return "http://schemas.microsoft.com/repl/" in the Public-Extension header. If the server supports this protocol, it has to return an Allow-Extension header with the token "http://schemas.microsoft.com/repl/" in it.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

Messages are transported by using HTTP, as specified in [RFC2518] and [RFC2616].

This protocol 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 might support other ports.

2.2 Message Syntax

The extension headers in this protocol conform to the form and behavior of other custom HTTP headers, as specified in [RFC2616] section 4.2, and are consistent with the WebDAV verbs and headers, as specified in [RFC2518] sections 8 and 9.

2.2.1 Headers

The **Augmented Backus-Naur Form (ABNF)** notation [RFC4234] is used to specify the format of the following header.

2.2.1.1 Range Header

A WebDAV replication–compliant server MUST implement the **paged results** and the range header, as specified in [MS-XWDSEARCH], in order to improve the scalability and performance of the server. Clients SHOULD use the range header and paged results in order to reduce the load on the server.

2.2.2 XML Elements

2.2.2.1 changetype

Namespace: http://schemas.microsoft.com/repl/

Syntax: <!ELEMENT changetype (delete | change | new | read) >

The purpose of the <changetype> element is for the server to indicate to the client the type of the change on a resource when the client retrieves the manifest of a collection.

2.2.2.2 collblob

Namespace: http://schemas.microsoft.com/repl/

Syntax: <!ELEMENT collblob (EMPTY | (#PCDATA)) >

The purpose of the **collblob** element is for the client to indicate that it wants to fetch a manifest from the server. The value of the **collblob** element is used by the client to provide its original replication state, and by the server to indicate the client's updated replication state.

2.2.2.3 contenttag

Namespace: http://schemas.microsoft.com/repl/

Syntax: <!ELEMENT contenttag (#PCDATA) >

A <contenttag> element is a token generated by the server that represents the state of the contents of a WebDAV collection. This is applied only to resources immediately subordinate to the target **URI**, but the target resource itself is excluded.

2.2.2.4 repl

Namespace: http://schemas.microsoft.com/repl/

Syntax: <!ELEMENT repl (changetype | collblob | resourcetaglist)>

The **repl** element specifies the replication **properties** to be returned from a **SEARCH** method.

2.2.2.5 repl-uid

Namespace: http://schemas.microsoft.com/repl/

Syntax: <!ELEMENT repl-uid (#PCDATA)>

The **repl-uid** element is a **universal unique identifier (UUID)** of a resource. The value of this property is a URI.

2.2.2.6 resourcetag

Namespace: http://schemas.microsoft.com/repl/

Syntax: <!ELEMENT resourcetag (#PCDATA) >

A <resourcetag> element is a token generated by the server that represents the state of a WebDAV resource. This is applied only to the resource. The value of this element is a URI.

The client SHOULD keep this property to reflect the state of the replicated resource. The following is a list of functions that are served by the <resourcetag>:

- A WebDAV client that wants to avail itself of the server-side conflict detection and resolution mechanism SHOULD send its previously obtained <resourcetag> in the request headers of the **GET**, **PUT**, **POST**, **PROPFIND**, **PROPPATCH**, **MOVE**, **COPY**, **DELETE** and **MKCOL** requests.
- A WebDAV client can use the <resourcetag> property on a resource to detect whether it has already obtained the latest version of a specific resource.
- A WebDAV client can use <resourcetag> property on a resource to ensure consistency when it uploads or downloads data.

2.2.2.7 resourcetaglist

Namespace: http://schemas.microsoft.com/repl/

Syntax: <!ELEMENT resourcetaglist (resourcetag+) >

The **resourcetaglist** element is a container for **resourcetags**.

2.2.3 Methods

2.2.3.1 COPY Method

The **COPY** method, as specified in [RFC2518], is used to duplicate an existing WebDAV resource. The **COPY** method in the context of the WebDAV Extensions for Replication is used to initiate a **COPY**for a particular WebDAV resource.

A WebDAV replication–compliant server might not return the <resourcetag> as a result of the execution of a **COPY** operation, for example when copying a message to the mail submission URL, as specified in [MS-XWDMAIL].

2.2.3.2 GET Method

A client can use the **GET** method, as specified in [RFC2518], to fetch the contents of an existing WebDAV resource. The **GET** method in the context of the WebDAV Extensions for Replication protocol is used to download the content change for a particular WebDAV resource.

Every WebDAV replication–compliant server MUST return the updated **resourcetag** element of the affected WebDAV resource in the response headers.

2.2.3.3 MKCOL Method

The **MKCOL** method, as specified in [RFC2518], is used to add a new collection resource to an existing collection resource. The **MKCOL** method in the context of WebDAV replication is used to upload the creation of a new collection resource.

Every WebDAV replication–compliant server MUST return the updated **resourcetag** and **repl-uid** elements of the affected collection resource in the response headers.

2.2.3.4 MOVE Method

The **MOVE** method, as specified in [RFC2518], is used to either move or rename an existing WebDAV resource. The **MOVE** method in the context of WebDAV replication is used to initiate a MOVE or RENAME for a particular WebDAV resource.

If the server changes the <repl-uid> of the object, it MUST return a Repl-uid: header with the new value of the <repl-uid> element.

2.2.3.5 POST Method

The **POST** method, as specified in [RFC2518], is used to add a new non-collection resource to an existing collection by using a server-defined name. The **POST** method in the context of the WebDAV Extensions for Replication protocol is used to upload the contents of a new resource in a particular collection.

Every WebDAV replication–compliant server MUST return the **resourcetag** and **repl-uid** elements of the new non-collection resource in the response headers.

2.2.3.6 PROPFIND Method

The **PROPFIND** method, as specified in [RFC2518], is used to fetch the properties of an existing WebDAV resource. The **PROPFIND** method cannot be used to determine what items have changed for replication within a collection. However, this functionality is available through the **SEARCH** method.

2.2.3.7 PROPPATCH Method

The **PROPPATCH** method, as specified in [RFC2518], is used to set or remove the properties of an existing WebDAV resource. The **PROPPATCH** method in the context of the WebDAV Extensions for Replication protocol is used to upload the property changes for a particular WebDAV resource.

2.2.3.8 PUT Method

The **PUT** method, as specified in [RFC2518], is used to either add a new non-collection resource to an existing collection or update an existing non-collection resource. The **PUT** method in the context of WebDAV replication is used to upload the content change for a particular non-collection resource.

2.2.3.9 SEARCH Method

The **SEARCH** method, as specified in [MS-XWDSEARCH], is used to search the properties of an existing WebDAV resource. The **SEARCH** method in the context of the WebDAV Extensions for Replication protocol is used to search for and download the property changes for WebDAV resources. The **SEARCH** method MUST be used to fetch the manifest of a collection or collection hierarchy.

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.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

The following section specifies extensions to the existing WebDAV commands, as specified in [RFC2518]. These commands SHOULD be processed as specified in [RFC2518], except in the cases specified in this section.

To keep a client view updated as changes happen on a server, a WebDAV replication client will typically perform a sequence of steps. First it will get an initial view of the underlying data by using a **SEARCH** command. The server returns replication state information called a "collection **binary large object (BLOB)**" in the **collblob XML** element.

When the client wants to check for changes, it submits the same **SEARCH** request, this time including the **collblob** value that was returned by the server in the previous request. The server will return the results as a set of changes relative to the previous result set, omitting any unchanged resources.

If the client makes changes, it will receive a **resourcetag** that uniquely identifies the changes it made. It can include that **resourcetag** in subsequent **SEARCH** requests so that it does not have to retrieve its own changes an extra time.

3.1.5.1 COPY Method

A client can use the **COPY** method, as specified in [RFC2518], to move or rename a resource.

3.1.5.1.1 COPY Method with Version Checking

If the client has previously downloaded content or properties of a resource, the server MUST have returned the **resourcetag** of that particular resource. Under these circumstances, the client can include the **resourcetag** in the request header of a **COPY** method in the form of **If: (<resourcetag >)** or **If: (<repl-uid>)**.

The **If: (<resourcetag>)** or **If: (<repl-uid>)** condition allows for client-initiated conflict detection.

3.1.5.1.2 COPY Method with Server-Side Modifications

The **COPY** method might trigger some server-side action that results in successful overwrite from the client perspective, but modifications or transformations on the server-side that result in a content and/or properties mismatch between the client and server. In this case, the server MUST return the new status code, 210 Content Different. The response SHOULD also include information about what was affected during the execution of the **COPY** method on the server.

To solve this mismatch problem, the client might need to re-fetch the contents and/or properties of all the affected resources by using the **GET** and **PROPFIND** methods.

3.1.5.1.3 COPY Method to Prevent Inadvertent Overwrite of an Existing Resource

The client can check to determine whether the resource it is intending to **COPY** already exists at the destination, and if the resource does exist, the client might not want to overwrite the existing resource. In this case, the client MUST include the Overwrite: F request header in the **COPY** request so as to avoid overwriting an existing resource.

3.1.5.1.4 COPY Method with Client-Initiated Conflict Detection

The client SHOULD include the **If: (<resourcetag>)** or **If: (<repl-uid>)** request header for the source collection, source non-collection, and destination collection in the **COPY** request, and move the resource on the server only if the version matches. If the condition fails, the server MUST return the 412 Precondition Failed error code.

3.1.5.2 GET Method

A client can use the **GET** method, as specified in [RFC2518], to download the content change for a particular WebDAV resource.

Every WebDAV replication–compliant server MUST return the updated **resourcetag** of the affected WebDAV resource in the response headers.

If the client issues a **GET** request without any headers specific to replication, the response from the server will have the default behavior as defined by [RFC2616] except that a WebDAV replication–compliant server MUST return the **resourcetag** of the affected resource.

3.1.5.2.1 GET Method with Version Checking

If the client has previously downloaded content or properties of a resource, the server MUST have returned the **resourcetag** of that particular resource. The client can include the **resourcetag** in the request header of a **GET** method in the form of **If: (<resourcetag>)** or **If: (<repl-uid>)**.

The **If: (<resourcetag >)** or **If: (<repl-uid>)** condition allows for client-initiated conflict detection.

3.1.5.2.2 GET Method with Client-Initiated Conflict Detection

The client SHOULD include the **If: (<resourcetag>)** or **If: (<repl-uid>)** request header in the **GET** request, and fetch the resource on the server. If the condition fails, the server MUST return the 412 Precondition Failed error code.

3.1.5.3 MKCOL Method

A client can use the **MKCOL** method, as specified in [RFC2518], to upload the creation of a new collection resource. A collection cannot be made at the Request-URI until one or more intermediate collections have been created. The server MUST NOT create those intermediate collections automatically.

If client issues a **MKCOL** request without any headers that are specific to replication, the request will have the default behavior except that a WebDAV replication–compliant server MUST return the **resourcetag** of the affected resource. The **MKCOL** method will fail with **409 Conflict** if the client tries to re-create a collection that already exists.

3.1.5.4 MOVE Method

A client can use the **MOVE** method, as specified in [RFC2518], to upload a **MOVE** or **RENAME** change for a particular WebDAV resource.

3.1.5.4.1 MOVE Method with Version Checking

If the client has previously downloaded the content or properties of a resource, the server MUST have returned the **resourcetag** of that particular resource. Under these circumstances, the client can include the **resourcetag** in the request header of a **MOVE** method in the form of **If: (<resourcetag>)** or **If: (<repl-uid>).**

The **If: (<resourcetag>)** or **If: (<repl-uid>)** condition allows for client-initiated conflict detection.

3.1.5.4.2 MOVE Method with server-side modifications

A **MOVE** method can trigger some server-side action that results in successful overwrite from the client perspective, but modifications or transformations on the server-side that result in a content and/or properties mismatch between the client and server.

In this case, the server MUST return the new status code, 210 Content Different. The response SHOULD also include information about what was affected during the execution of the **MOVE** method on the server.

In order to solve this mismatch problem, the client might need to re-fetch the contents and/or properties of all the affected resources by using the **GET** and **PROPFIND** methods.

3.1.5.4.3 MOVE Method to Prevent Inadvertent Overwrite of an Existing Resource

The client might want to check to determine whether the resource it is intending to **MOVE** already exists at the destination, and if so, it might not want to overwrite the existing resource In this case, the client MUST include the Overwrite: F request header in the **MOVE** request to avoid overwriting an existing resource.

3.1.5.5 POST Method

A client can use the **POST** method, as specified in [RFC2518], to upload the contents of a new resource in a particular collection.

If a client issues a **POST** request without any headers that are specific to replication, the request will have the default behavior as defined by the HTTP and WebDAV drafts, except that a WebDAV replication–compliant server MUST return the **resourcetag** of any created or updated resource.

A server MUST ignore any request headers related to **resourcetag** or **repl-uid** for a **POST** request because they do not hold any special meaning or purpose.

3.1.5.6 PROPFIND Method

A client can use the **PROPFIND** method, as specified in [RFC2518], to download the property changes for a particular WebDAV resource.

A WebDAV replication–compliant server MUST NOT return the **resourcetag** of the affected resources in the response headers due to the possibility of large result set. However, the client can fetch the **resourcetag** as a property of every WebDAV resource reported in the response of the **PROPFIND** method.

3.1.5.7 PROPPATCH Method

A client can use the **PROPPATCH** method, as specified in [RFC2518], to upload the property changes for a particular WebDAV resource.

Every WebDAV replication–compliant server MUST return the updated **resourcetag** and **repl-uid** of the affected WebDAV resource in the response headers. The **PROPPATCH** behavior is very similar to the **PUT** method behavior, except that the **PROPPATCH** deals with properties rather than the resource contents.

3.1.5.8 PUT Method

A client can use the **PUT** method, as specified in [RFC2518], to upload the content change for a particular non-collection resource.

If the client issues a **PUT** request without any headers that are specific to replication, the request will have the default behavior as defined by the HTTP and WebDAV drafts except that a WebDAV replication–compliant server MUST return the **resourcetag** and **repl-uid** of the affected resource.

3.1.5.8.1 PUT Method with version Checking

If the client has previously downloaded the content or properties of a resource, the server MUST have returned the **resourcetag** of that particular resource. The client can include the **resourcetag** in the request header of a **PUT** method in the form of **If: (<resourcetag>)**. The client can include the **repl-uid** in the request header of a **PUT** method in the form of **If: (<repl-uid>)**.

The **If: (<resourcetag>)** or **If: (<repl-uid>)** condition allows for client-initiated conflict detection.

3.1.5.8.2 PUT Method with Server-Side Modifications

The **PUT** method can trigger some server-side action that results in successful overwrite from client perspective, but modifications or transformations on the server-side that result in a content and/or properties mismatch between the client and server. Because every **PUT** method MUST return the updated **resourcetag**, there is a mismatch between the content and/or properties on the client and the content and/or properties that are reflected by the **resourcetag**.

In this case, the server MUST return the new status code, 210 Content Different. The response SHOULD also include information about what was affected during the execution of the **PUT** method on the server.

To solve this mismatch problem, the client might need to re-fetch the contents and/or properties of the affected resource by using the **GET** and **PROPFIND** methods.

3.1.5.8.3 PUT Method to Prevent Inadvertent Overwrite of an Existing Resource

Sometimes the client might want to check to determine whether the resource it is intending to **PUT** already exists, and if so, it might not want to overwrite the existing contents.

In this case, the client SHOULD include the request header with attribute **If-None-Match: *** in the **PUT** request to avoid overwriting an existing resource.

3.1.5.8.4 PUT Method with Client-Initiated Conflict Detection

The client SHOULD include the **If: (<resourcetag>)** or **If: (<repl-uid>)** request header in the **PUT** request, and update the resource on the server only if the version matches. If the condition fails, the server MUST return a 412 Precondition Failed error code.

3.1.5.9 SEARCH Method

A client can use the **SEARCH** method, as specified in [MS-XWDSEARCH], to search for and download the property changes for WebDAV resources. It can include the **collblob** element to retrieve only the changes relative to a previous **SEARCH** request.

A WebDAV replication–compliant server SHOULD NOT return the **resourcetag** of the affected resources in the response headers due to the possibility of a large result set. However, the client can explicitly fetch the **resourcetag** as a property of every WebDAV resource reported in the response of the **SEARCH** method.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

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.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 collblob Element

The **collblob** element is an encoded string that specifies the state of a collection. It is generated by the server and the client MUST treat it as an opaque value. It can be sent to the server to communicate the state of a collection that the client has previous retrieved, and a new value is returned by the server after various operations.

3.2.5.2 changetype Element

The following are the **changetype** element attributes that are defined by the WebDAV Extensions for Replication protocol:

- **change** – An existing resource has been updated. The client MUST assume that the default **changetype** in the manifest response from the server is **change,** if there is no **changetype** given in the **response** XML element.
- **delete** – A resource has been deleted. A server MUST return the **repl-uid** property of the resource if the **changetype** on the resource is **delete**.
- **new** – A new resource has been added since the last time the collblob was returned.
- **read** – The only thing that has changed on the resource is the read/unread state. If the read/unread state has changed and any other change has occurred, the **changetype** will be **change**.

The protocol does not prevent the extensibility in terms of the other potential **changetypes** based on the client-server negotiation.

The change in the manifest applies only to the resource referenced in the **DAV:href** property in the response XML element of the manifest.

In the absence of any other additional **changetype** elements, the following occurs:

- A move operation SHOULD be indicated by the server in the source collection as **delete changetype**, and as a **change changetype** in the destination collection manifest.
- A copy operation SHOULD be indicated by the server as a **change changetype** in the destination collection manifest.
- A rename operation and a move operation SHOULD be treated as the same type of operation for the purposes of registering **changetypes**.

3.2.5.3 contenttag Element

A <contenttag> element is a token generated by the server that represents the state of the contents of a collection. This is applied only to resources immediately subordinate to the target URI but the target resource itself is excluded. Every time the contents or properties of resources within the collection change, the <contenttag> property on the collection MUST be updated to reflect the change. A server that implements WebDAV replication MUST support the http://schemas.microsoft.com/repl/contenttag property on every WebDAV collection resource that can be replicated. The server MUST guarantee that two <contenttags> are binary comparable. The client can store the <contenttags> on the collections for future comparisons. The <contenttag> element can do the following:

- Provide an efficient way for the client to determine whether a collection has changed since the last time it synchronized by comparing the value on the client. The client SHOULD use **PROPFIND** or **SEARCH** to fetch the <contenttag> property of the collection and then compare the value against its previously obtained value from the server.
- Allow an efficient and easy way to check for not only collection hierarchy changes, but also collections the contents for which have changed. The client SHOULD use **PROPFIND** or **SEARCH** to fetch the <contenttag> property of the collections in the hierarchy and then compare it against its previously obtained values from the server without the **collblob** or **repl** XML element.
- In conjunction with **repl-uid**, MAY allow the client find out whether a collection has moved.
- Be used in an if: header to make sure that an operation (especially **DELETE**) on a collection will only happen if the contents of the collection have not changed. As such, it can be used in an If: header anywhere that <resourcetag> or <repl-uid> elements are allowed.

3.2.5.4 repl-uid Element

A **repl-uid** is a universal unique identifier (UUID) that identifies a WebDAV resource. The value of this property is a URI. A server that implements WebDAV replication MUST support the **http://schemas.microsoft.com/repl/repl-uid** property on every replicated WebDAV resource.

The **http://schemas.microsoft.com/repl/repl-uid** property MAY be obtained as property on a resource by using the **PROPFIND** or **SEARCH** command.

Deletes might not have valid URL identifiers but they MUST have valid **repl-uids**.

A client can include the unique identifier in the request header only if its intention is to ensure that it is dealing with the same resource that it has always known.

Note that a server can change the **http://schemas.microsoft.com/repl/repl-uid** property on a resource, if the resource is moved, renamed, or copied.

The server MUST return the **repl-uid** of the resource as a response header in every **PUT**, **POST**, **MKCOL**, and **PROPPATCH** request.

3.2.5.5 resourcetag Element

A <resourcetag> element is a token generated by the server that represents the state of a WebDAV resource. This is applied only to the resource. A server that implements WebDAV replication MUST be able to generate the <resourcetag>. The server also MUST provide support for the http://schemas.microsoft.com/repl/resourcetag property on every replicated WebDAV **resource**. The value of this property is a URI. Note that the contents of <resourcetag> are opaque to the client. The following are the requirements that a <resourcetag> MUST meet:

- Two <resourcetags> MUST be binary comparable by the client.
- The server MUST guarantee that if two <resourcetaga> are the same when compared, the resource MUST be the same.
- The client MUST be able to fetch http://schemas.microsoft.com/repl/resourcetag as a property on the resource.
- It MUST be possible for the client to include the <resourcetag> or **repl-uid** in the If: request header of any WebDAV request.

- The server MUST return the <resourcetag> of the resource as a response header in every **GET**, **PUT**, **POST**, **MKCOL**, **PROPPATCH**, and **DELETE** requests.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

4.1 Client Has Never Fetched the Manifest of a Collection

A client has never fetched the manifest for a collection. The client includes the **searchrequest**, **repl**, and **collblob** elements to request the manifest.

```
>>Request
SEARCH /private/user0/inbox HTTP/1.1
Host: www.company.com
Content-type: text/xml
Content-length: {insert length here}

<?xml version="1.0"?>
<D:searchrequest xmlns:D="DAV:"
    xmlns:R="http://schemas.microsoft.com/repl/"
    xmlns:M="urn:schemas:mail: ">
    <R:repl>
     <R:collblob/>
    </R:repl>
    <D:sql>
     SELECT 'urn:schemas:mail:Size', 'urn:schemas:mail:Importance',
               'http://schemas.microsoft.com/repl/resourcetag'
          FROM SCOPE ('SHALLOW TRAVERSAL OF "http://www.company.com/private/user0/inbox"')
    </D:sql>
</D:searchrequest>

>>Response
HTTP/1.1 207 Multi-Status
Content-type: text/xml
Content-length: {insert length here}

<?xml version="1.0"?>
<D:multistatus xmlns:D="DAV:"
    xmlns:R="http://schemas.microsoft.com/repl/"
    xmlns:M="urn:schemas:mail:">
     <R:repl>
          <R:collblob>clientopaquedata</R:collblob>
     </R:repl>
     <D:response>
                 <D:href>http://www.company.com/private/user0/inbox/msg1</D:href>
                 <D:propstat>
                     <D:status>HTTP/1.1 200 OK</D:status>
                     <D:prop>
                         <D:Size>1000</D:Size>
                         <M:Importance>High</M:Importance>
                   <R:resourcetag>doc1-01</R:resourcetag>
                     </D:prop>
                 </D:propstat>
                 <R:changetype>change</R:changetype>
         </D:response>
         <D:response>
                 <D:href>http://www.company.com/private/user0/inbox/msg4</D:href>
                 <D:propstat>
                     <D:status>HTTP/1.1 200 OK</D:status>
                     <D:prop>
                         <D:Size>14400</D:Size>
                         <M:Importance>High</M:Importance>
```

```
                    <R:resourcetag>doc2-02</R:resourcetag>
                      </D:prop>
                    </D:propstat>
                    <R:changetype>change</R:changetype>
      </D:response>
  </D:multistatus>
```

4.2 Client-Side Detection to Avoid Unnecessary Downloads

A client has fetched the manifest and the **collblob** for a collection 'doccoll ' before, and is seeking an updated **collblob** and manifest for the collection.

```
>>Request
SEARCH /doccoll HTTP/1.1
Host: www.company.com
Content-type: text/xml
Content-length: {insert length here}

<?xml version="1.0"?>
<D:searchrequest xmlns:D="DAV:"
    xmlns:R="http://schemas.microsoft.com/repl/"
    xmlns:M="urn:schemas:mail:">
    <R:repl>
<R:collblob>clientopaquedata</R:collblob>
    </R:repl>
</D:searchrequest>

>>Response
HTTP/1.1 207 Multi-Status
Content-type: text/xml
Content-length: {insert length here}

<?xml version="1.0"?>
<D:multistatus xmlns:D="DAV:"
    xmlns:R="http://schemas.microsoft.com/repl/"
    xmlns:M="urn:schemas:mail:">
<R:repl>
<R:collblob>clientopaquedata</R:collblob>
</R:repl>
<D:response>
        <D:href>http://www.company.com/doccoll/msg1</D:href>
        <D:propstat>
            <D:status>HTTP/1.1 200 OK</D:status>
            <D:prop>
                <D:Size>1000</D:Size>
                <M:Importance>High</M:Importance>
<R:resourcetag>rt:doc1-01</R:resourcetag>
            </D:prop>
        </D:propstat>
        <R:ChangeType>change</R:ChangeType>
    </D:response>
    <D:response>
        <D:href>http://www.company.com/doccoll/msg4</D:href>
        <D:propstat>
            <D:status>HTTP/1.1 200 OK</D:status>
            <D:prop>
```

```
                <D:Size>14400</D:Size>
                <M:Importance>High</M:Importance>
<R:resourcetag>rt:doc1-01</R:resourcetag>
            </D:prop>
        </D:propstat>
        <R:ChangeType>change</R:ChangeType>
</D:response>
</D:multistatus>
```

While the client was offline, document 'docE' in the collection 'doccoll' was updated.

Client A uses a Web browser to download a document 'docE'. The server returns the contents the document 'docE' and its corresponding **resourcetag**, as follows:

```
>>Request
GET /doccoll/docE HTTP/1.1

>>Response
HTTP/1.1 200 OK
Resourcetag: <rt:19a23000c26511d18faf00600892444c>
Content-type: text/plain
Content-length: {insert length here}

This is the content of text document docE.
```

Client A saves the contents of the document docE, and its **resourcetag**.

After a while, Client A wants find out what has changed since the last time it synchronized with the same server. Client A sends a request for the manifest and **resourcetag** property for collection 'doccoll' by including its previous **collblob** in the request, and the server responds with a manifest that includes the change that corresponds to docE.

Becuase the client does not want to unnecessarily re-download documents, it compares the **resourcetag** that it obtained as part of the manifest with the **resourcetag** that it persisted before for document docE, and finds that it already has the latest version of the document docE.

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.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index