[MS-METAWEB]:
MetaWeblog Extensions Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability
06/27/2008	1.0	Major	Revised and edited the technical content
12/12/2008	1.01	Editorial	Revised and edited the technical content
07/13/2009	1.02	Major	Changes made for template compliance
08/28/2009	1.03	Editorial	Revised and edited the technical content
11/06/2009	1.04	Editorial	Revised and edited the technical content
02/19/2010	2.0	Editorial	Revised and edited the technical content
03/31/2010	2.01	Editorial	Revised and edited the technical content
04/30/2010	2.02	Editorial	Revised and edited the technical content
06/07/2010	2.03	Editorial	Revised and edited the technical content
06/29/2010	2.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	2.05	Editorial	Changed language and formatting in the technical content.
09/27/2010	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
01/20/2012	2.6	Minor	Clarified the meaning of the technical content.
04/11/2012	2.6	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	2.7	Minor	Clarified the meaning of the technical content.
09/12/2012	2.7	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2012	2.7	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

The MetaWeblog Extensions Protocol is a set of extensions to the MetaWeblog API to enable more secure authentication mechanisms.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

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

Hypertext Transfer Protocol (HTTP)
NT LAN Manager (NTLM) Authentication Protocol
server
Unicode
XML

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

blog
category

The following terms are specific to this document:

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

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical documents, which are updated frequently. References to other documents include a publishing year when one is available.

1.2.1 Normative References

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

[Blogger API] Williams, E., "Blogger API", August 2001, http://www.blogger.com/developers/api/1_docs/

[MS-NTHT] Microsoft Corporation, "NTLM Over HTTP Protocol Specification".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

[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

[RFC-MWA] Winer, D., "RFC: MetaWeblog API", March 2002, http://www.xmlrpc.com/metaWeblogApi

[XML-RPC] Winer, D., "XML-RPC Specification", June 1999, http://www.xmlrpc.com/spec

1.2.2 Informative References

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

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

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary".

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

1.3 Overview

The RFC: MetaWeblog API, as described in [RFC-MWA], is a protocol that allows client software to get and set the text and attributes of posts on a **blog (1)**. The RFC: MetaWeblog API uses the **XML**-RPC communication protocol, as described in [XML-RPC], for communication between client applications and a blog (1) **server (1)**. The client sends XML-RPC method call requests to the server (1), and the server (1) returns a response to the client. The server (1) never initiates any communication with the client.

This protocol extends the RFC: MetaWeblog API to enable the client sending empty usernames and passwords in MetaWeblog API methods, and the server (1) authenticating blog (1) users with other mechanisms in the underlying transport. A typical scenario is that a blog (1) user wants to add a new post to a blog (1), and the client software sends a method request with empty username and password parameters.

1.4 Relationship to Other Protocols

This protocol is a set of extensions to the RFC: MetaWeblog API that enables the server (1) to use other authentication methods. The RFC: MetaWeblog API is an extension of the Blogger API, as described in [Blogger API]. Therefore, many blog (1) tools and editors that support the RFC: MetaWeblog API also support the Blogger API.

1.5 Prerequisites/Preconditions

It is assumed that a MetaWeblog client has obtained the name of a server (1) that supports the RFC: MetaWeblog API before this protocol is invoked.

The protocol server (1) endpoint is formed by appending "_layouts/metaweblog.aspx" to the URL of the blog (1) site; for example, http://www.example.com/_layouts/metaweblog.aspx.

1.6 Applicability Statement

The RFC: MetaWeblog API is applicable wherever there is a need to get and set the text and attributes of posts on a blog (1). This protocol extension is applicable when a client using the RFC: MetaWeblog API does not send the user's name and password directly in the messages, but rather uses the authentication performed by an underlying transport.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- **Supported Transports:** This protocol can only be implemented using **Hypertext Transfer Protocol (HTTP)**, as described in section 2.1.

Security and Authentication Methods: The MetaWeblog API methods each contain a *username* and a *password* parameter for user authentication purposes. Some MetaWeblog servers also support alternate authentication methods such as **NT LAN Manager (NTLM) Authentication Protocol**, as described in [MS-NLMP], and HTTP authentication, as described in [RFC2617].

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

A MetaWeblog message is an HTTP version 1.1 POST request, as specified in [RFC2616]. The body of the request is in XML. A procedure executes on the server (1), and the response that is returned is formatted in XML.

2.2 Common Message Syntax

The format of the XML body for an XML-RPC request and response is specified in [XML-RPC].

The format of the method request and response for each MetaWeblog method is specified in [RFC-MWA].

The Blogger API functions are specified in [Blogger API].

Each method in the MetaWeblog API provides *username* and *password* parameters for authenticating the blog (1) user with the server (1). If the server (1) is capable of authenticating the user by using the authentication methods described in section 1.7, these parameters are not necessary and a client SHOULD<1><2> pass them as empty elements.

2.2.1 metaWeblog.newPost Extension

The **metaWeblog.newPost** method posts a new entry to a blog (1). The structure of this method, as specified in [RFC-MWA], is as follows.

```
public string metaWeblog.newPost(string blogid,
string username,
string password,
struct struct,
bool publish);
```

The use of the following parameters differs from that which is specified in [RFC-MWA].

username: An XML-encoded **Unicode** string that contains the login for the blog (1) user, which SHOULD<3> be empty.

password: An XML-encoded Unicode string that contains the user's password, which SHOULD<4> be empty.

2.2.2 metaWeblog.editPost Extension

The **metaWeblog.editPost** method edits an existing entry on a blog (1). The structure of this method, as specified in [RFC-MWA], is as follows.

```
public bool metaWeblog.editPost(string postid,
string username,
string password,
struct struct,
bool publish);
```

The use of the following parameters differs from that which is specified in [RFC-MWA].

username: An XML-encoded Unicode string containing the login for the blog (1) user, which SHOULD<5> be empty.

password: An XML-encoded Unicode string containing the user's password, which SHOULD<6> be empty.

2.2.3 metaWeblog.getPost Extension

The **metaWeblog.getPost** method returns a specific entry from a blog (1). The structure of this method, as specified in [RFC-MWA], is as follows.

```
public struct metaWeblog.getPost(string postid,
string username,
string password);
```

The use of the following parameters differs from that which is specified in [RFC-MWA].

username: An XML-encoded Unicode string containing the login for the blog (1) user, which SHOULD<7> be empty.

password: An XML-encoded Unicode string containing the user's password, which SHOULD<8> be empty.

2.2.4 metaWeblog.newMediaObject Extension

The **metaWeblog.newMediaObject** method uploads a file from a user's computer to the user's blog (1). The structure of this method, as specified in [RFC-MWA], is as follows.

```
public struct metaWeblog.newMediaObject(string blogid,
string username,
string password,
struct struct);
```

The use of the following parameters differs from that which is specified in [RFC-MWA].

username: An XML-encoded Unicode string containing the login for the blog (1) user, which SHOULD<9> be empty.

password: An XML-encoded Unicode string containing the user's password, which SHOULD<10> be empty.

2.2.5 metaWeblog.getCategories Extension

The **metaWeblog.getCategories** method returns the list of **categories (1)** that have been used in the blog (1). The structure of this method, as specified in [RFC-MWA], is as follows.

```
public struct[] metaWeblog.getCategories(string blogid,
string username,
string password);
```

The use of the following parameters differs from that which is specified in [RFC-MWA].

username: An XML encoded Unicode string containing the login for the blog (1) user, which SHOULD<11> be empty.

password: An XML encoded Unicode string containing the user's password, which SHOULD<12> be empty.

2.2.6 metaWeblog.getRecentPosts Extension

The **metaWeblog.getRecentPosts** method returns the most recent draft and non-draft blog (1) posts in descending order by publish date. The structure of this method, as specified in [RFC-MWA], is as follows.

```
public struct[] metaWeblog.getRecentPosts(string blogid,
string username,
string password,
int numberOfPosts);
```

The use of the following parameters differs from that which is specified in [RFC-MWA].

username: An XML encoded Unicode string containing the login for the blog (1) user, which SHOULD<13> be empty.

password: An XML encoded Unicode string containing the user's password, which SHOULD<14> be empty.

2.2.7 blogger.getUsersBlogs Extension

The **blogger.getUsersBlogs** method returns a list of blogs (1) to which the current authenticated user has posting privileges. The structure of this method, as specified in [Blogger API], is as follows.

```
public struct[] blogger.getUsersBlogs(string appkey,
string username,
string password);
```

The use of the following parameters differs from that which is specified in [Blogger API].

username: An XML-encoded Unicode string containing the login for the blog (1) user, which SHOULD<15> be empty.

password: An XML-encoded Unicode string containing the user's password, which SHOULD<16> be empty.

3 Protocol Details

3.1 Common Details

3.1.1 Abstract Data Model

The abstract data model follows the specifications in [RFC-MWA] and [XML-RPC].

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

When the server (1) receives a protocol message containing an empty *username* or an empty *password*, the server (1) MUST ensure that authentication has been established through some other mechanism compatible with the HTTP transport, as specified in [RFC2616], before performing the requested action.<17>

If the authentication through other mechanism fails to authenticate a blog (1) user to the MetaWeblog service, an error MUST be returned as an XML-RPC <methodResponse> with a <fault> item, as specified in [XML-RPC]. If the authentication through another mechanism succeeds in authenticating the blog (1) user, the server (1) MUST perform the requested action and return a response, as specified in [RFC-MWA].

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

4 Protocol Examples

4.1 Client Messages

Each method, as described in [RFC-MWA], follows a similar request and response pattern. The following example demonstrates a request with empty *username* and *password* parameters.

4.1.1 metaWeblog.newPost

When a blog (1) user adds a new post to a blog (1), the client software will send a **metaWeblog.newPost** request, as described in [XML-RPC], with a body formatted as XML as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
   <methodName>metaWeblog.newPost</methodName>
   <params>
      <param>
         <value>
            <string>MyBlog</string>
         </value>
      </param>
      <param>
         <value>
            <string></string>
         </value>
      </param>
      <param>
         <value>
            <string></string>
         </value>
      </param>
      <param>
         <value>
            <struct>
               <member>
                  <name>title</name>
                  <value>
                     <string>My Title</string>
                  </value>
               </member>
               <member>
                  <name>description</name>
                  <value>
                     <string>My description</string>
                  </value>
               </member>
               <member>
                  <name>categories</name>
                  <value>
                     <array>
                        <data>
                           <value>
                              <string>My Category</string>
                           </value>
                        </data>
                     </array>
```

```
                  </value>
               </member>
            </struct>
         </value>
      </param>
      <param>
         <value>
            <boolean>1</boolean>
         </value>
      </param>
   </params>
</methodCall>
```

The server (1) processes the **metaWeblog.newPost** method request and returns a response, as described in [XML-RPC], with a body formatted as XML as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
   <params>
      <param>
         <value>1829</value>
      </param>
   </params>
</methodResponse>
```

5 Security

5.1 Security Considerations for Implementers

The protocol described in [RFC-MWA] uses HTTP version 1.1 as its transport mechanism. The security considerations for HTTP 1.1 are described in [RFC2616], section 15.

The protocols described in [Blogger API] and [RFC-MWA] use clear-text *username* and *password* parameters for authentication. This makes the protocol vulnerable to replay attacks, and permits the recovery of the user's password via the recording of client and server (1) protocol exchanges.

The vulnerabilities of the RFC: MetaWeblog API can be mitigated by relying on authentication methods in the underlying transport such as the NTLM Over HTTP protocol, as described in [MS-NTHT] or in [RFC2617], and by sending empty *username* and *password* parameters in the message, as described in [RFC-MWA].

5.2 Index of Security Parameters

None.

6 Appendix A: XML-RPC Schema

For ease of implementation the following XML-RPC Schema is provided.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

 <xsd:element name="methodCall">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="methodName">
          <xsd:simpleType>
            <xsd:restriction base="ASCIIString">
              <xsd:pattern value="([A-Za-z0-9]|/|\.|:|_)*" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="params" minOccurs="0" maxOccurs="1">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="param"  type="ParamType"
                           minOccurs="0" maxOccurs="unbounded" />
            </xsd:sequence>
          </xsd:complexType>
         </xsd:element>
      </xsd:all>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="methodResponse">
    <xsd:complexType>
      <xsd:choice>
        <xsd:element name="params">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="param" type="ParamType" />
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="fault">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="value">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="struct">
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="member"
                                       type="MemberType" />
                          <xsd:element name="member"
                                       type="MemberType" />
                        </xsd:sequence>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
```

```
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="ParamType">
  <xsd:sequence>
    <xsd:element name="value" type="ValueType" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ValueType" mixed="true">
  <xsd:choice>
    <xsd:element name="i4"            type="xsd:int" />
    <xsd:element name="int"           type="xsd:int" />
    <xsd:element name="string"        type="ASCIIString" />
    <xsd:element name="double"        type="xsd:decimal" />
    <xsd:element name="Base64"        type="xsd:base64Binary" />
    <xsd:element name="boolean"       type="NumericBoolean" />
    <xsd:element name="dateTime.iso8601" type="xsd:dateTime" />
    <xsd:element name="array"         type="ArrayType" />
    <xsd:element name="struct"        type="StructType" />
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="StructType">
  <xsd:sequence>
    <xsd:element name="member" type="MemberType"
                 maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="MemberType">
  <xsd:sequence>
    <xsd:element name="name"  type="xsd:string" />
    <xsd:element name="value" type="ValueType" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ArrayType">
  <xsd:sequence>
    <xsd:element name="data">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="value"  type="ValueType"
                       minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="ASCIIString">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="([ -~]|\n|\r|\t)*" />
  </xsd:restriction>
</xsd:simpleType>
```

```
    <xsd:simpleType name="NumericBoolean">
      <xsd:restriction base="xsd:boolean">
        <xsd:pattern value="0|1" />
      </xsd:restriction>
    </xsd:simpleType>

  </xsd:schema>
```

7 Appendix B: Product Behavior

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

- The 2007 Microsoft® Office system
- Microsoft® Office 2010 suites
- Microsoft® Office 2013
- Microsoft® Office SharePoint® Server 2007
- Microsoft® SharePoint® Server 2010
- Windows® SharePoint® Services 3.0
- Microsoft® SharePoint® Foundation 2010
- Microsoft® SharePoint® Foundation 2013
- Microsoft® Office Word 2007

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

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

<1> Section 2.2: Office Word 2007 sends an empty *password* parameter when communicating with Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010. Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010 can be configured to allow a non-empty password, but by default it will send a failure response if the client sends a non-empty password.

<2> Section 2.2: Office Word 2007 sends an empty *username* parameter when communicating with Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010. Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010 can be configured to allow a nonempty *username*, but by default it will send a failure response if the client sends a nonempty *username*.

<3> Section 2.2.1: Office Word 2007 sends an empty *username* parameter when communicating with Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010. Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010 can be configured to allow a nonempty *username*, but by default it will send a failure response if the client sends a nonempty *username*.

<4> Section 2.2.1: Office Word 2007 sends an empty *password* parameter when communicating with Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010. Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010 can be configured to allow a non-empty password, but by default it will send a failure response if the client sends a non-empty password.

<5> Section 2.2.2: Office Word 2007 sends an empty *username* parameter when communicating with Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010. Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010 can be configured to allow a nonempty *username*, but by default it will send a failure response if the client sends a nonempty *username*.

<6> Section 2.2.2: Office Word 2007 sends an empty *password* parameter when communicating with Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010. Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010 can be configured to allow a non-empty password, but by default it will send a failure response if the client sends a non-empty password.

<7> Section 2.2.3: Office Word 2007 sends an empty *username* parameter when communicating with Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010. Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010 can be configured to allow a nonempty *username*, but by default it will send a failure response if the client sends a nonempty *username*.

<8> Section 2.2.3: Office Word 2007 sends an empty *password* parameter when communicating with Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010. Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010 can be configured to allow a non-empty password, but by default it will send a failure response if the client sends a non-empty password.

<9> Section 2.2.4: Office Word 2007 sends an empty *username* parameter when communicating with Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010. Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010 can be configured to allow a nonempty *username*, but by default it will send a failure response if the client sends a nonempty *username*. metaWeblog.newMediaObject method is not supported on Office server and server (1) will send a failure response if client calls this method.

<10> Section 2.2.4: Office Word 2007 sends an empty *password* parameter when communicating with Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010. Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010 can be configured to allow a non-empty password, but by default it will send a failure response if the client sends a non-empty password. metaWeblog.newMediaObject method is not supported on Office server and server (1) will send a failure response if client calls this method.

<11> Section 2.2.5: Office Word 2007 sends an empty *username* parameter when communicating with Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010. Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010 can be configured to allow a nonempty *username*, but by default it will send a failure response if the client sends a nonempty *username*.

<12> Section 2.2.5: Office Word 2007 sends an empty *password* parameter when communicating with Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010. Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010 can be configured to allow a non-empty password, but by default it will send a failure response if the client sends a non-empty password.

<13> Section 2.2.6: Office Word 2007 sends an empty *username* parameter when communicating with Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010. Office SharePoint Server 2007, SharePoint Server 2010, Windows

SharePoint Services 3.0, SharePoint Foundation 2010 can be configured to allow a nonempty *username*, but by default it will send a failure response if the client sends a nonempty *username*.

<14> Section 2.2.6: Office Word 2007 sends an empty *password* parameter when communicating with Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010. Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010 can be configured to allow a non-empty password, but by default it will send a failure response if the client sends a non-empty password.

<15> Section 2.2.7: Office Word 2007 sends an empty *username* parameter when communicating with Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010. Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010 can be configured to allow a nonempty *username*, but by default it will send a failure response if the client sends a nonempty *username*.

<16> Section 2.2.7: Office Word 2007 sends an empty *password* parameter when communicating with Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010. Office SharePoint Server 2007, SharePoint Server 2010, Windows SharePoint Services 3.0, SharePoint Foundation 2010 can be configured to allow a non-empty password, but by default it will send a failure response if the client sends a non-empty password.

<17> Section 3.1.5: Office SharePoint Server 2007 authenticates blog (1) users using [MS-NTHT].

8 Change Tracking

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

9 Index