[MS-OUTSPS]:
Lists Client Sync 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'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
04/04/2008	0.1		Initial Availability
06/27/2008	1.0	Major	Revised and edited the technical content
10/06/2008	1.01	Editorial	Revised and edited the technical content
12/12/2008	1.02	Editorial	Revised and edited the technical content
07/13/2009	1.03	Major	Revised and edited the technical content
08/28/2009	1.04	Editorial	Revised and edited the technical content
11/06/2009	1.05	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	Minor	Clarified the meaning of 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.

Table of Contents

1 Introduction

This document specifies the Lists Client Sync Protocol for transferring organized collections of data, which is "items," between a server and a client. These items conform to a specific abstract data model (section 3.2.1). The data model is implemented by a schema defined in Schema of Each Item Type (section 3.2.4.2). The items are accessed via the Lists Web Service Protocol, as described in [MS-LISTSWS].

A protocol client can use this protocol to provide a richer and more responsive experience to users by maintaining local copies of data from the protocol server.<1>

1.1 Glossary

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

Coordinated Universal Time (UTC)
GUID
Hypertext Transfer Protocol (HTTP)
Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)
Kerberos
NT LAN Manager (NTLM) Authentication Protocol
XML

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

change token
checked out
discussion item
list
list identifier
property bag
Simple Object Access Protocol (SOAP)
site
SOAP fault
Uniform Resource Locator (URL)
user identifier
Web Services Description Language (WSDL)
WSDL operation

The following terms are specific to this document:

yomigana: The phonetic rendering of Japanese kanji characters.

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-LISTSWS] Microsoft Corporation, "Lists Web Service Protocol Specification"

[MS-OXOCNTC] Microsoft Corporation, "Contact Object 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

[WSDL] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S., "Web Services Description Language (WSDL) 1.1", W3C Note, March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[XMLNS] Bray, T., Hollander, D., Layman, A., et al., Eds., "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

[XMLSCHEMA1] Thompson, H.S., Ed., Beech, D., Ed., Maloney, M., Ed., and Mendelsohn, N., Ed., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[XMLSCHEMA2] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

1.2.2 Informative References

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

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

[MS-STSSYN] Microsoft Corporation, "StsSync Data Structure"

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

[SOAP1.1] Box, D., Ehnebuske, D., Kakivaya, G., et al., "Simple Object Access Protocol (SOAP) 1.1", May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[SOAP1.2/1] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., and Nielsen, H.F., "SOAP Version 1.2 Part 1: Messaging Framework", W3C Recommendation, June 2003, http://www.w3.org/TR/2003/REC-soap12-part1-20030624

[SOAP1.2/2] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., and Nielsen, H.F., "SOAP Version 1.2 Part 2: Adjuncts", W3C Recommendation, June 2003, http://www.w3.org/TR/2003/REC-soap12-part2-20030624

1.3 Protocol Overview (Synopsis)

The general scenario that the Lists Client Sync Protocol handles is as follows: Data exists on a server that needs to be transferred to a client computer. Once transferred, the data can be modified on the client. Once data on the client has been modified, the client can choose to update or not update the server. Either way, after this protocol successfully runs, the client's data becomes an accurate copy of the server data.

This protocol specifies transfer of data that conforms to the schemas specified in Schema of Each Item Type (section 3.2.4.2) between a protocol client and a protocol server. The protocol uses [MS-LISTSWS] to transfer all data except for files. Files are transferred over HTTP or HTTPS.

1.4 Relationship to Other Protocols

This protocol uses the **SOAP** message protocol for formatting request and response messages, as described in [SOAP1.1], [SOAP1.2/1] and [SOAP1.2/2]. It transmits those messages by using **HTTP**, as described in [RFC2616], or **HTTPS**, as described in [RFC2818].

The Lists Client Sync Protocol uses [MS-LISTSWS] as shown in the following layering diagram. The protocol uses HTTP or HTTPS directly when downloading and uploading files and [MS-LISTSWS] otherwise.

The following diagram shows the underlying messaging and transport stack used by the protocol:

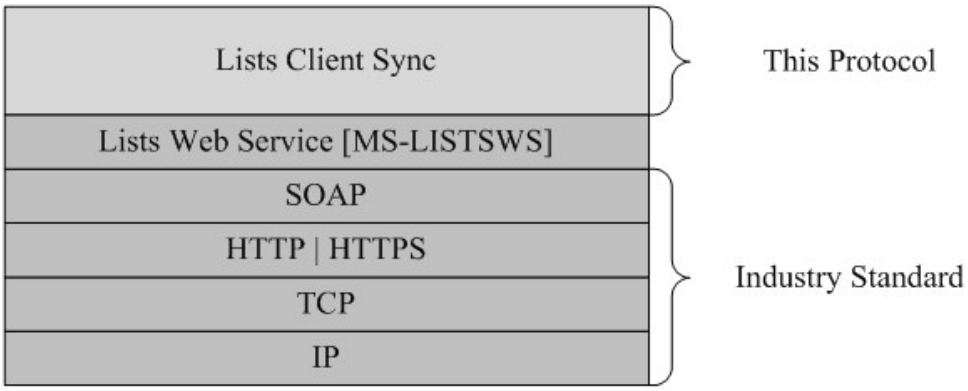

Figure 1: This protocol in relation to other protocols

Clients of this protocol can use a protocol such as that specified by the StsSync Structure Specification ([MS-STSSYN]) to obtain the information necessary to use [MS-LISTSWS]. Protocol clients can also create their own way of obtaining that information, so an understanding of [MS-STSSYN] is not required.

1.5 Prerequisites/Preconditions

This protocol assumes that the protocol client already has the following information:

A valid **URL** for a server that contains the **list** (1) where the protocol client looks to transfer items to or from.

A valid **list identifier** for the list where the protocol client looks to transfer items to or from.

The type of the list specified by the server URL and list identifier. The items in the list can be calendars, contacts, discussions, documents, or tasks. The type describes which type of items the list contains. See Abstract Data Model (section 3.2.1) for item types.

Whether HTTP or HTTPS is used to communicate with the server.

The prerequisites and preconditions of [MS-LISTSWS].

1.6 Applicability Statement

This protocol can be used in scenarios where a client and server both implement the abstract data model (section 3.2.1) and the server implements the protocol described by [MS-LISTSWS]. All restrictions in the [MS-LISTSWS] applicability statement also apply to this protocol because clients are required to use [MS-LISTSWS] to implement this protocol.

This protocol is intended to transfer the data of the abstract data model (section 3.2.1) between the protocol server and the protocol client so that both agree on the state of the items after the protocol completes.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- **Supported Transports:** This protocol uses multiple transports with SOAP as specified in Transport (section 2.1).
- **Protocol Versions:** This protocol is based on [MS-LISTSWS] and shares the same versioning.
- **Security and Authentication Methods:** This protocol supports the following authentication methods: LANMAN, **NT LAN Manager (NTLM) Authentication Protocol**, and **Kerberos**.
- **Capability Negotiation:** This protocol does explicit negotiation as specified in this section.

If a protocol server implements **GetListItemChanges** and not **GetListItemChangesSinceToken**,<2> the protocol client has to implement some way of detecting that. This is optional and a protocol client can choose to support only one of those **WSDL operations**.<3> For an example of how a protocol client might do this, see GetList (section 3.1.4.5).<4> For information about those WSDL operations, see [MS-LISTSWS].

1.8 Vendor-Extensible Fields

This protocol does not define any vendor-extensible fields, but because this protocol uses [MS-LISTSWS], any vendor extensible fields from that protocol can be used with this protocol. See [MS-LISTSWS] section 1.8.

1.9 Standards Assignments

None.

2 Messages

In the following sections, the schema definition might be less restrictive than the processing rules imposed by the protocol. The **WSDL** in this specification matches the WSDL that shipped with the product and provides a base description of the schema. The text that introduces the WSDL specifies additional restrictions that reflect protocol behavior. For example, the schema definition might allow for an element to be **empty**, **null**, or **not present** but the behavior of the protocol as specified restricts the same elements to being **non-empty**, **present**, and **not null**.

2.1 Transport

Protocol servers MUST support SOAP over HTTP. Protocol servers SHOULD additionally support SOAP over HTTPS for securing communication with clients.

This protocol uses the same transport, security model, and SOAP versions as [MS-LISTSWS].

2.2 Common Message Syntax

This section contains common definitions used by this protocol. The syntax of the definitions uses XML Schema as defined in [XMLSCHEMA1] and [XMLSCHEMA2], and Web Services Description Language (WSDL) as defined in [WSDL].

2.2.1 Namespaces

This protocol uses the same namespaces as specified in [MS-LISTSWS] section 2.2.1.

2.2.2 Messages

None.

2.2.3 Elements

None.

2.2.4 Complex Types

The following table summarizes the set of common XML Schema complex type definitions defined by this specification. XML Schema complex type definitions that are specific to a particular operation are described with the operation.

Complex type	Description
AttachProps	Contains information about an attachment.
RecurrenceRule	Defines when a recurrence takes place.
RecurrenceDefinition	Contains a RecurrenceRule.
RecurrenceXML	Contains a RecurrenceDefinition.
RepeatPattern	Contains a choice of elements which describe what days a recurrence occurs on.
TimeZoneRule	Contains daylight saving time (DST) biases and TransitionDate elements.
TimeZoneXML	Contains a TimeZoneRule.

Complex type	Description
TransitionDate	Contains DST transition dates.

2.2.4.1 AttachProps

The **AttachProps** complex type contains information about an attachment.

```
<s:complexType name="AttachProps">
  <s:sequence>
    <s:element name="File">
      <s:complexType>
        <s:simpleContent>
          <s:extension base="s:string">
            <s:attribute name="Photo" use="required">
              <s:simpleType>
                <s:restriction base="s:string">
                  <s:enumeration value="0" />
                  <s:enumeration value="-1" />
                </s:restriction>
              </s:simpleType>
            </s:attribute>
          </s:extension>
        </s:simpleContent>
      </s:complexType>
    </s:element>
  </s:sequence>
</s:complexType>
```

File: Contains the file name to which the attributes of the **File** element apply.

File.Photo: Indicates if this file is used as a photo file for the item. For example, if the file is used as a photo file for the item, a protocol client can interpret the photo file as an icon for the item, a picture of a person related to the item, or something implementation-specific. Exactly zero or one **File** element in **AttachProps** MUST have **File.Photo** set to "-1". "-1" indicates the file is the photo file for the item.

2.2.4.2 RecurrenceRule

The **RecurrenceRule** complex type defines when a recurrence takes place.

```
<s:complexType name="RecurrenceRule">
  <s:sequence>
    <s:element name="firstDayOfWeek" type="s1:DayOfWeekOrMonth" />
    <s:element name="repeat" type="s1:RepeatPattern" />
    <s:choice>
      <s:element name="windowEnd" type="s:dateTime" />
      <s:element name="repeatForever">
        <s:simpleType>
          <s:restriction base="s:string">
            <s:enumeration value="FALSE" />
          </s:restriction>
        </s:simpleType>
      </s:element>
      <s:element name="repeatInstances" type="s:integer" />
    </s:choice>
```

```
    </s:sequence>
  </s:complexType>
```

firstDayOfWeek: Indicates which day of the week is considered the first day of the week. For example, if **firstDayOfWeek** is Thursday, then the second day is Friday, and so on.

Repeat: Using **firstDayOfWeek** as the first day of the week, this element specifies the repeat pattern of the recurrence.

windowEnd: If this element is present then the recurrence ends at the date and time specified in windowEnd.

repeatForever: If this element is present then the recurrence has no end date and repeats forever.

repeatInstances: If this element is present then the recurrence ends after the number of instances specified by this element.

2.2.4.3 RecurrenceDefinition

```
The RecurrenceDefinition complex type contains a RecurrenceRule (section 2.2.4.2).
<s:complexType name="RecurrenceDefinition">
  <s:sequence>
    <s:element name="rule" type="s1:RecurrenceRule" />
  </s:sequence>
</s:complexType>
```

rule: Contains a recurrence rule that defines a recurrence.

2.2.4.4 RecurrenceXML

The **RecurrenceXML** complex type contains a **RecurrenceDefinition** (section 2.2.4.3).

```
<s:complexType name="RecurrenceXML">
  <s:sequence>
    <s:element name="recurrence" type="s1:RecurrenceDefinition" />
    <s:element name="deleteExceptions" type="s:string" fixed="true" minOccurs="0"
maxOccurs="1" />
  </s:sequence>
</s:complexType>
```

recurrence: Contains elements describing a recurrence.

deleteExceptions: This element MUST be present if and only if **RecurrenceXML** is written by a protocol client in **UpdateListItems** (section 3.1.4.10) and the protocol client requests that the protocol server delete all exception items for this recurrence. See Appointments (section 3.2.1.1) for details about exception items and recurrences.

The following is an example of how to specify **RecurrenceXML** for a daily recurrence that occurs every three days.

```
<recurrence>
  <rule>
    <firstDayOfWeek>su</firstDayOfWeek>
    <repeat>
```

```
        <daily dayFrequency="3" />
      </repeat>
      <repeatForever>FALSE</repeatForever>
    </rule>
  </recurrence>
```

The following is an example of how to specify **RecurrenceXML** for a monthly recurrence that occurs every five months on the second Tuesday of the month.

```
  <recurrence>
     <rule>
     <firstDayOfWeek>su</firstDayOfWeek>
     <repeat>
         <monthlyByDay tu='TRUE' weekdayOfMonth='second' monthFrequency='5' />
      </repeat>
      <repeatForever>FALSE</repeatForever>
    </rule>
  </recurrence>
```

2.2.4.5 RepeatPattern

The **RepeatPattern** complex type contains a choice of elements which describe what days a recurrence occurs on.

```
<s:complexType name="RepeatPattern">
  <s:choice>
    <s:element name="daily">
      <s:complexType>
        <s:simpleContent>
          <s:extension base="s:string">
            <s:attribute name="su" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="mo" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="tu" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="we" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="th" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="fr" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="sa" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="weekFrequency" type="s:integer" default="1" use="optional" />
          </s:extension>
        </s:simpleContent>
      </s:complexType>
    </s:element>
    <s:element name="monthlyByDay">
      <s:complexType>
        <s:simpleContent>
          <s:extension base="s:string">
            <s:attribute name="su" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="mo" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="tu" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="we" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="th" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="fr" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="sa" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="day" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="weekday" type="s1:TrueFalseDOW" default="FALSE" use="optional"
/>
```

```
            <s:attribute name="weekend_day" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
            <s:attribute name="monthFrequency" type="s:integer" default="1" use="optional" />
            <s:attribute name="weekdayOfMonth" type="s1:WeekdayOfMonth" default="first"
use="optional" />
          </s:extension>
        </s:simpleContent>
      </s:complexType>
    </s:element>
    <s:element name="monthly">
      <s:complexType>
        <s:simpleContent>
          <s:extension base="s:string">
            <s:attribute name="monthFrequency" type="s:integer" default="1" use="optional" />
            <s:attribute name="day" type="s:integer" default="1" use="optional" />
          </s:extension>
        </s:simpleContent>
      </s:complexType>
    </s:element>
    <s:element name="yearly">
      <s:complexType>
        <s:simpleContent>
          <s:extension base="s:string">
            <s:attribute name="yearFrequency" type="s:integer" default="1" use="optional" />
            <s:attribute name="month" type="s:integer" default="1" use="optional" />
            <s:attribute name="day" type="s:integer" default="1" use="optional" />
          </s:extension>
        </s:simpleContent>
      </s:complexType>
    </s:element>
    <s:element name="yearlyByDay">
      <s:complexType>
        <s:simpleContent>
          <s:extension base="s:string">
            <s:attribute name="su" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="mo" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="tu" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="we" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="th" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="fr" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="sa" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="day" type="s1:TrueFalseDOW" default="FALSE" use="optional" />
            <s:attribute name="weekday" type="s1:TrueFalseDOW" default="FALSE" use="optional"
/>
            <s:attribute name="weekend_day" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
            <s:attribute name="yearFrequency" type="s:integer" default="1" use="optional" />
            <s:attribute name="month" type="s:integer" default="1" use="optional" />
            <s:attribute name="weekdayOfMonth" type="s1:WeekdayOfMonth" default="first"
use="optional" />
          </s:extension>
        </s:simpleContent>
      </s:complexType>
    </s:element>
  </s:choice>
</s:complexType>
```

daily: If the recurrence is a daily recurrence, the **daily** element will be present.

daily.su: If TRUE, this attribute indicates the recurrence occurs on Sundays.

daily.mo: If TRUE, this attribute indicates the recurrence occurs on Mondays.

daily.tu: If TRUE, this attribute indicates the recurrence occurs on Tuesdays.

daily.we: If TRUE, this attribute indicates the recurrence occurs on Wednesdays.

daily.th: If TRUE, this attribute indicates the recurrence occurs on Thursdays.

daily.fr: If TRUE, this attribute indicates the recurrence occurs on Fridays.

daily.sa: If TRUE, this attribute indicates the recurrence occurs on Saturdays.

monthlyByDay: If the recurrence is a monthly recurrence and the day it occurs depends on which day of the week the month starts with, then the **monthlyByDay** element will be present.

monthlyByDay.su: If TRUE, this attribute indicates the recurrence occurs on Sunday specified by **monthlyByDay.weekdayOfMonth**.

monthlyByDay.mo: If TRUE, this attribute indicates the recurrence occurs on Monday specified by **monthlyByDay.weekdayOfMonth**.

monthlyByDay.tu: If TRUE, this attribute indicates the recurrence occurs on Tuesday specified by **monthlyByDay.weekdayOfMonth**.

monthlyByDay.we: If TRUE, this attribute indicates the recurrence occurs on Wednesday specified by **monthlyByDay.weekdayOfMonth**.

monthlyByDay.th: If TRUE, this attribute indicates the recurrence occurs on Thursday specified by **monthlyByDay.weekdayOfMonth**.

monthlyByDay.fr: If TRUE, this attribute indicates the recurrence occurs on Friday specified by **monthlyByDay.weekdayOfMonth**.

monthlyByDay.sa: If TRUE, this attribute indicates the recurrence occurs on Saturday specified by **monthlyByDay.weekdayOfMonth**.

monthlyByDay.day: If TRUE, this attribute indicates the recurrence occurs on the day specified by **monthlyByDay.weekdayOfMonth**.

monthlyByDay.weekday: If TRUE, this attribute indicates the recurrence occurs on the weekday (Monday, Tuesday, Wednesday, Thursday, or Friday) specified by **monthlyByDay.weekdayOfMonth**.

monthlyByDay.weekend_day: If TRUE, this attribute indicates the recurrence occurs on the weekend day (Saturday or Sunday) specified by **monthlyByDay.weekdayOfMonth**.

monthlyByDay.monthFrequency: An integer specifying how many months go by before the next month the recurrence occurs on. A 1 means "every month". A 2 means "every other month". A 3 would be "every third month". A 4 means "every fourth month", and so on.

monthlyByDay.weekdayOfMonth: Specifies how to interpret the other **monthlyByDay** attributes by adding a limitation to which days qualify. For example, if this attribute is "first", then the first day of the month that matches the other recurrence attributes is the day the recurrence occurs on. If **monthlyByDay.mo** is TRUE, and **monthlyByDay.weekdayOfMonth** is first, then the recurrence occurs on the first Monday.

monthly: If the recurrence is a monthly recurrence on the Nth day of a month, then this element will be present.

monthly.monthFrequency: Specifies the frequency of the recurrence in months. 1=every month, 2=every second month, 3=every third month, and so on.

monthly.day: Specifies the day of the month that the recurrence occurs on, counting up from 1="the first day of the month".

yearly: If the recurrence is a yearly recurrence on the Nth day of a month specified by a number, then this element will be present.

yearly.yearFrequency: Specifies the frequency in years that the recurrence occurs. 1=every year, 2=every second year, 3=every third year, and so on.

yearly.month: Specifies the month that the recurrence occurs on, counting up from 1="the first month of the year".

yearly.day: Specifies the day of the month that the recurrence occurs on, counting up from 1, where 1 means "the first day of the month".

yearlyByDay: If the recurrence is a yearly recurrence and the day it occurs depends on which day of the week the month starts with, then the **yearlyByDay** element will be present. For example, a yearly recurrence that occurs on the first Wednesday of June will use the **yearlyByDay** element. A yearly recurrence that occurs on the 5th of June will use the **yearly** element and not the **yearlyByDay** element.

yearlyByDay.su: Specifies that a Sunday is the day of the occurrence.

yearlyByDay.mo: Specifies that a Monday is the day of the occurrence.

yearlyByDay.tu: Specifies that a Tuesday is the day of the occurrence.

yearlyByDay.we: Specifies that a Wednesday is the day of the occurrence.

yearlyByDay.th: Specifies that a Thursday is the day of the occurrence.

yearlyByDay.fr: Specifies that a Friday is the day of the occurrence.

yearlyByDay.sa: Specifies that a Saturday is the day of the occurrence.

yearlyByDay.day: Specifies that any day can be the day of the recurrence.

yearlyByDay.weekday: Specifies that weekday (Monday, Tuesday, Wednesday, Thursday, or Friday) can be the day of the recurrence.

yearlyByDay.weekend_day: Specifies that weekend day (Saturday or Sunday) is the day of the recurrence.

yearlyByDay.yearFrequency: Specifies the frequency in years of the recurrence, counting up from 1="every year".

yearlyByDay.month: Specifies the month of the occurrence in the year, counting up from 1="the first month of the year".

yearlyByDay.weekdayOfMonth: Specifies which one of the days allowed by the other yearlyByDay attributes is the day the recurrence occurs on.

2.2.4.6 TimeZoneRule

The **TimeZoneRule** complex type contains DST biases and **TransitionDate** (section 2.2.4.8) elements.

```
<s:complexType name="TimeZoneRule">
  <s:sequence>
    <s:element name="standardBias" type="s:integer" />
    <s:element name="additionalDaylightBias" type="s:integer" minOccurs="0" />
    <s:element name="standardDate" type="s1:TransitionDate" minOccurs="0" />
    <s:element name="daylightDate" type="s1:TransitionDate" minOccurs="0" />
  </s:sequence>
</s:complexType>
```

standardBias: An integer that specifies the time difference, in minutes, from **Coordinated Universal Time (UTC)**.

additionalDaylightBias: An integer that specifies in minutes the time that is added to the **standardBias** while the time zone is between **daylightDate** and **standardDate**.

standardDate: The date and time after which only **standardBias** is applied.

daylightDate: The date and time after which **standardBias** plus **additionalDaylightBias** is applied.

2.2.4.7 TimeZoneXML

The **TimeZoneXML** complex type contains a **TimeZoneRule** (section 2.2.4.6) to define a time zone.

```
<s:complexType name="TimeZoneXML">
  <s:sequence>
    <s:element name="timeZoneRule" type="s1:TimeZoneRule" />
  </s:sequence>
</s:complexType>
```

timeZoneRule: A **TimeZoneRule** that describes the time zone.

2.2.4.8 TransitionDate

The **TransitionDate** complex type contains transition dates for DST.

```
<s:complexType name="TransitionDate">
  <s:sequence>
    <s:element name="transitionRule">
      <s:complexType>
        <s:simpleContent>
          <s:extension base="s:string">
            <s:attribute name="day" type="s1:DayOfWeek" default="su" use="optional" />
            <s:attribute name="month" type="s:integer" use="required" />
            <s:attribute name="dayOfMonth" type="s:integer" use="optional" />
            <s:attribute name="weekdayOfMonth" type="s1:WeekdayOfMonth" default="first"
use="optional" />
          </s:extension>
```

```
          </s:simpleContent>
        </s:complexType>
      </s:element>
      <s:element name="transitionTime" type="s:time" />
    </s:sequence>
  </s:complexType>
```

transitionRule: contains attributes defining the transition date and time.

transitionRule.day: The day of the transition.

transitionRule.month: The month of the transition, counting up from 1="the first month of the year".

transitionRule.dayOfMonth: The day of the transition, counting up from 1="the first day of the month". If this attribute is present then the transition date is the Nth day of the month where N is the value of **dayOfMonth**. If **dayOfMonth** is absent then the transition day is calculated from the **day** and **weekdayOfMonth** attributes.

transitionRule.weekdayOfMonth: This attribute restricts the days allowed by the **day** attribute so that only a single day is valid.

transitionTime: This is the time of day that the transition takes place. The clocks in the time zone officially change at this time.

2.2.5 Simple Types

The following table summarizes the set of common XML Schema simple type definitions defined by this specification. XML Schema simple type definitions that are specific to a particular operation are described with the operation.

Simple type	Description
BusyStatus	Busy status of a block of time.
booleanInteger	An integer used to represent a Boolean value.
DayOfWeek	Specifies a day of the week.
DayOfWeekOrMonth	Specifies a day of the week or a day of a month when combined with a WeekdayOfMonth value.
EventType	For items that can be recurring, this represents the type of recurrence.
FollowUp	An integer representing a color.
Gender	An integer representing a gender.
Importance	An integer representing importance.
Participants	A string delimited by ";#" that lists **user identifier** numbers.
Priority	An integer representing priority.
stringGUID	A **GUID** written as a string.
TrueFalseDOW	A Boolean value that specifies TRUE or FALSE.

Simple type	Description
WeekdayOfMonth	Specifies a day of the week or a day of a month when combined with a DayOfWeek or DayOfWeekOrMonth value.

2.2.5.1 BusyStatus

The **BusyStatus** simple type specifies the status of a block of time.

```
<s:simpleType name="BusyStatus">
  <s:restriction base="s:int">
    <s:enumeration value="-1"/>
    <s:enumeration value="0"/>
    <s:enumeration value="1"/>
    <s:enumeration value="2"/>
    <s:enumeration value="3" />
  </s:restriction>
</s:simpleType>
```

Value	Meaning
-1	Unspecified busy status. Protocol clients can choose to display one of the other values if **BusyStatus** is -1.
0	Free.
1	Tentative.
2	Busy.
3	Out of office.

2.2.5.2 booleanInteger

The **booleanInteger** simple type is an integer used to represent a Boolean value. If a nonzero value is received in a **booleanInteger** type field protocol clients and protocol servers MUST treat the nonzero value as "1".

```
<s:simpleType name="booleanInteger">
  <s:restriction base="s:int">
    <s:enumeration value="0" />
    <s:enumeration value="1" />
  </s:restriction>
</s:simpleType>
```

Value	Meaning
0	False
1	True

2.2.5.3 DayOfWeek

The **DayOfWeek** simple type specifies a day of the week.

```
<s:simpleType name="DayOfWeek">
  <s:restriction base="s:string">
    <s:enumeration value="su" />
    <s:enumeration value="mo" />
    <s:enumeration value="tu" />
    <s:enumeration value="we" />
    <s:enumeration value="th" />
    <s:enumeration value="fr" />
    <s:enumeration value="sa" />
  </s:restriction>
</s:simpleType>
```

Value	Meaning
su	Sunday
mo	Monday
tu	Tuesday
we	Wednesday
th	Thursday
fr	Friday
sa	Saturday

2.2.5.4 DayOfWeekOrMonth

The **DayOfWeekOrMonth** simple type specifies a day of the week or a day of the month. This is typically combined with a **WeekdayOfMonth** value to choose a particular day out of the ones allowed by **DayOfWeekOrMonth**. Example: **DayOfWeekOrMonth**="su" allows Sunday, but there are several Sundays in a typical month. Adding **WeekdayOfMonth**="first" would make it clear that only the "first Sunday of the month" is allowed.

```
<s:simpleType name="DayOfWeekOrMonth">
  <s:restriction base="s:string">
    <s:enumeration value="su" />
    <s:enumeration value="mo" />
    <s:enumeration value="tu" />
    <s:enumeration value="we" />
    <s:enumeration value="th" />
    <s:enumeration value="fr" />
    <s:enumeration value="sa" />
    <s:enumeration value="day" />
    <s:enumeration value="weekday" />
    <s:enumeration value="weekend_day" />
  </s:restriction>
</s:simpleType>
```

Value	Meaning
su	Sunday
mo	Monday

Value	Meaning
tu	Tuesday
we	Wednesday
th	Thursday
fr	Friday
sa	Saturday
day	Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday are allowed.
weekday	Monday, Tuesday, Wednesday, Thursday, and Friday are allowed.
weekend_day	Sunday and Saturday are allowed.

2.2.5.5 EventType

The **EventType** simple type describes whether the containing item is recurring.

```
<s:simpleType name="EventType">
  <s:restriction base="s:int" >
    <s:enumeration value="0" />
    <s:enumeration value="1" />
    <s:enumeration value="2" />
    <s:enumeration value="3" />
    <s:enumeration value="4" />
  </s:restriction>
</s:simpleType>
```

Value	Meaning
0	Single instance.
1	Recurring.
2	MUST be interpreted as 0.
3	Deleted instance of a recurrence.
4	Exception to a recurrence.

2.2.5.6 FollowUp

The **FollowUp** simple type indicates to the user that some action needs to be taken. **FollowUp** values SHOULD correspond to colors, but can represent something implementation-specific.

```
<s:simpleType name="FollowUp">
  <s:restriction base="s:int" >
    <s:enumeration value="0" />
    <s:enumeration value="1" />
    <s:enumeration value="2" />
    <s:enumeration value="3" />
    <s:enumeration value="4" />
    <s:enumeration value="5" />
```

```
  </s:restriction>
</s:simpleType>
```

Value	Meaning
0	None
1	Red
2	Orange
3	Green
4	Yellow
5	Blue

2.2.5.7 Importance

The **Importance** simple type indicates the level of importance.

```
<s:simpleType name="Importance">
  <s:restriction base="s:int" >
    <s:enumeration value="0" />
    <s:enumeration value="1" />
    <s:enumeration value="2" />
  </s:restriction>
</s:simpleType>
```

Value	Meaning
0	Low
1	Normal
2	High

2.2.5.8 Participants

The **Participants** simple type is a string that lists user identifiers. These user identifiers are delimited by ";#".

```
<s:simpleType name="Participants">
  <s:restriction base="s:string" >
    <s:maxLength value="255" />
    <s:pattern value="(;#[0-9]+)*" />
  </s:restriction>
</s:simpleType>
```

2.2.5.9 Priority

The **Priority** simple type indicates the level of precedence.

```
<s:simpleType name="Priority">
```

```
  <s:restriction base="s:int" >
    <s:enumeration value="-1" />
    <s:enumeration value="0" />
    <s:enumeration value="1" />
  </s:restriction>
</s:simpleType>
```

Value	Meaning
-1	Low priority
0	Normal
1	Urgent

2.2.5.10 stringGUID

The **stringGUID** simple type is a GUID written as a string using hexadecimal digits enclosed by {} and separated by '-', for example, "{B8760CFE-3A46-46c1-B4C3-D32FE4F294D2}".

```
<s:simpleType name="stringGUID">
  <s:restriction base="s:string">
    <s:maxLength value="38"/>
    <s:minLength value="38"/>
    <s:pattern value="\{[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-
F]{12}\}" />
  </s:restriction>
</s:simpleType>
```

2.2.5.11 TrueFalseDOW

The **TrueFalseDOW** simple type specifies TRUE or FALSE. This type is used with an attribute name that is the same as one of the **WeekdayOfMonth**, **DayOfWeek**, or **DayOfWeekOrMonth** enumerated types, as in the **RepeatPattern** complex type. For example, **DayOfWeek** has an enumeration value "mo" that represents Monday. A **TrueFalseDOW** type attribute named "mo" with the value "TRUE" would mean that Monday is TRUE. This document specifies the individual meaning and use of all such attributes with the type that contains them.

```
<s:simpleType name="TrueFalseDOW">
  <s:restriction base="s:string">
    <s:enumeration value="TRUE" />
    <s:enumeration value="FALSE" />
    <s:enumeration value="true" />
    <s:enumeration value="false" />
  </s:restriction>
</s:simpleType>
```

Value	Meaning
TRUE	True
FALSE	False
true	True

Value	Meaning
false	False

2.2.5.12 WeekdayOfMonth

When combined with a **DayOfWeek** or **DayOfWeekOrMonth** value, the **WeekdayOfMonth** simple type specifies a day of a week or month. For example, **DayOfWeekOrMonth**="su" **WeekdayOfMonth**="first" would be the "first Sunday of the month".

```
<s:simpleType name="WeekdayOfMonth">
  <s:restriction base="s:string">
    <s:enumeration value="first" />
    <s:enumeration value="second" />
    <s:enumeration value="third" />
    <s:enumeration value="fourth" />
    <s:enumeration value="last" />
  </s:restriction>
</s:simpleType>
```

Value	Meaning
first	First
second	Second
third	Third
fourth	Fourth
last	Last

2.2.6 Attributes

None.

2.2.7 Groups

None.

2.2.8 Attribute Groups

None.

3 Protocol Details

In the following sections, the schema definition might be less restrictive than the processing rules imposed by the protocol. The WSDL in this specification matches the WSDL that shipped with the product and provides a base description of the schema. The text that introduces the WSDL specifies additional restrictions that reflect protocol behavior. For example, the schema definition might allow for an element to be **empty**, **null**, or **not present** but the behavior of the protocol as specified restricts the same elements to being **non-empty**, **present**, and **not null**.

The Lists Web Service Protocol specifies how to transfer data between a client and a server. The Lists Client Protocol specifies individual WSDL operations of the Lists Web service to explain the behavior of the protocol client. [MS-LISTSWS] specifies the behavior of the protocol server for the Lists Web Service Protocol.

Except where specified, protocol clients SHOULD interpret HTTP Status Codes returned by the protocol server as specified in [RFC2616], section10, Status Code Definitions.

This protocol allows protocol servers to perform implementation-specific authorization checks and notify protocol clients of authorization faults either using HTTP Status Codes or using **SOAP faults** as specified previously in this section.

3.1 Server 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.

Protocol clients and protocol servers use the same Abstract Data Model (section 3.2.1).

3.1.2 Timers

This protocol uses the [MS-LISTSWS] protocol.

3.1.3 Initialization

See [MS-LISTSWS] section 3.1.3 for details about initialization of the protocol server.

3.1.4 Message Processing Events and Sequencing Rules

Operation	Description
AddAttachment	Adds an attachment to an item.
AddDiscussionBoardItem	Adds a new **discussion item** to a list.
DeleteAttachment	Deletes an attachment from an item on a list.
GetAttachmentCollection	Gets a list of the attachments on an item.
GetList	Gets information about a list.
GetListItemChanges	Gets item information from a list.

Operation	Description
GetListItemChangesSinceToken	Gets an item from and information about a list.
UpdateListItems	Creates or modifies items on a list.

3.1.4.1 AddAttachment

AddAttachment is used by protocol clients to create a new attachment on an item on the protocol server.

The details of **AddAttachment** are specified in [MS-LISTSWS]. This document only covers the usage of particular elements and attributes of **AddAttachment** for this protocol.

The protocol client fills in the **AddAttachment** elements specified in [MS-LISTSWS].

3.1.4.1.1 Messages

3.1.4.1.1.1 AddAttachmentResponse

Protocol clients process **AddAttachmentResponse** (see [MS-LISTSWS]) to confirm the successful upload of the attachment.

If an **AddAttachmentResponse** is received, then the upload was successful.

If a SOAP exception is received instead of an **AddAttachmentResponse**, the protocol client SHOULD behave as follows:

- If the exception **errorcode** (see [MS-LISTSWS] section 3.1.4.1) is 0x81020067, this indicates that the item already has an attachment with the same file name. The protocol client SHOULD use HTTP PUT to overwrite the attachment binary, as specified in section 3.1.4.9 of this document.
- If the **errorcode** is any other value, the protocol client SHOULD treat the operation as failed and move on to the next operation.

If anything other than an **AddAttachmentResponse** or a SOAP exception is received, then the protocol client SHOULD assume the upload failed.

If no response is received, then protocol clients SHOULD assume the protocol server did not receive the request and can try again at a time allowed by the Timers (section 3.2.2).

3.1.4.2 AddDiscussionBoardItem

Protocol clients SHOULD use this operation to add new discussion items to a list. Protocol clients SHOULD NOT use **UpdateListItems** to add new discussion items to a list. Protocol servers who implement **AddDiscussionBoardItem** MAY do additional processing on discussion items<5>. If protocol clients use **UpdateListItems** instead, protocol servers will not do any additional processing of the items.

The details of **AddDiscussionBoardItem** are specified in [MS-LISTSWS]. This document only covers the usage of particular elements and attributes of **AddDiscussionBoardItem** for this protocol.

3.1.4.2.1 Messages

3.1.4.2.1.1 AddDiscussionBoardItemResponse

Protocol clients SHOULD process **AddDiscussionBoardItemResponse** (see [MS-LISTSWS]) to retrieve the identifier of the item on the protocol server. The item identifier is found in the attribute **AddDiscussionBoardItemResponse.AddDiscussionBoardItemResult.listitems.data.row.ows_ID**. Protocol clients SHOULD remember the identifier and use it in an **UpdateListItems** operation to update the item with any data that **AddDiscussionBoardItem** was unable to upload.

If protocol clients receive a SOAP exception or connection error, they SHOULD treat the operation as failed and move on to the next item.

3.1.4.3 DeleteAttachment

Protocol clients use **DeleteAttachment** to delete attachments from an item on the protocol server.

The details of **DeleteAttachment** are specified in [MS-LISTSWS]. This document only covers the usage of particular elements and attributes of **DeleteAttachment** for this protocol.

3.1.4.3.1 Messages

3.1.4.3.1.1 DeleteAttachmentResponse

Protocol clients SHOULD process **DeleteAttachmentResponse** (see [MS-LISTSWS]) to confirm successful deletion of the attachment. If protocol clients receive a SOAP exception, they SHOULD assume that the attachment cannot be deleted. If protocol clients do not receive any response, they SHOULD assume the protocol server did not receive the request and can try again later.

If protocol clients receive a SOAP exception or connection error, they SHOULD treat the operation as failed and move on to the next attachment or item.

3.1.4.4 GetAttachmentCollection

Protocol clients use **GetAttachmentCollection** to get the list of all attachments on a single item in one list.

3.1.4.4.1 Messages

The details of **GetAttachmentCollection** are specified in [MS-LISTSWS]. This document only covers the usage of particular elements and attributes of **GetAttachmentCollection** for this protocol.

3.1.4.4.1.1 GetAttachmentCollectionResponse

Protocol clients SHOULD process **GetAttachmentCollectionResponse** to obtain the requested list of attachments. See [MS-LISTSWS] about the **GetAttachmentCollectionResponse.GetAttachmentCollectionResult.Attachments.Attachment** element. Protocol clients who request to download the attachments SHOULD use HTTP GET as specified in section 3.1.4.8 to download each attachment.

3.1.4.5 GetList

Protocol clients SHOULD use **GetList** as the first message sent as part of the sync process. The purpose of **GetList** is to obtain the schema and version of the protocol server. Additional useful information can be obtained, but the protocol client can choose to ignore all of it.

If the protocol server supports **GetListItemChangesSinceToken** and the protocol client is going to call that method, then protocol clients SHOULD remember the information from the first **GetList** and rely on **GetListItemChangesSinceToken** to inform them of any changes. Remembering information from **GetList** instead of using **GetList** again will help the protocol server perform better.

Schema for each item type is specified in Schema of Each Item Type (section 3.2.4.2).

The details of **GetList** are specified in [MS-LISTSWS]. This document only covers the usage of particular elements and attributes of **GetList** for this protocol.

3.1.4.5.1 Messages

3.1.4.5.1.1 GetListResponse

See [MS-LISTSWS] on each of the following parts of **GetListResponse**. Protocol clients SHOULD pay attention to these values and can use other information in **GetListResponse** as well. The elements and attributes listed here are listed along with sections of this document that reference them.

[MS-LISTSWS] [MS-LISTSWS] section 3.1.4.16.2.2 defines the **GetListResponse.GetListResult.List** element. This document uses the following elements and attributes from the **GetListResponse.GetListResult.List** element.

Element or attribute name	Purpose	Section
Description	A text description of the list.	None
EnableAttachments	"True" if items are allowed to have attachments. Protocol clients can ignore this and let the protocol server enforce it.	None
ExcludeFromOfflineClient	Protocol clients SHOULD not make the data available offline when this is "True".<6>	None
Fields.Field.ID	Identifier of a field.	Schema of Each Item Type (section 3.2.4.2)
Fields.Field.Name	Name of a field.	Schema of Each Item Type (section 3.2.4.2)
Fields.Field.Type	Type of a field.	Schema of Each Item Type (section 3.2.4.2)
Fields.Field.CHOICES	Suggested values for the field.	Tasks (section 3.2.1.5)
Fields.Field.MAPPINGS	Allows protocol clients to store some text values as numbers.	Tasks (section 3.2.1.5)
RootFolder	The root folder of a list.	Section 3.2.4.2.5
ServerSettings.ServerVersion	Server version.	Versioning and

Element or attribute name	Purpose	Section
		Capability Negotiation (section 1.7)

3.1.4.6 GetListItemChanges

Protocol clients SHOULD use **GetListItemChangesSinceToken** instead of **GetListItemChanges** on protocol servers that support **GetListItemChangesSinceToken**. See section 1.7 for information about how to determine this.

The details of **GetListItemChanges** are specified in [MS-LISTSWS]. This document only covers the usage of particular elements and attributes of **GetListItemChanges** for this protocol.

3.1.4.7 GetListItemChangesSinceToken

GetListItemChangesSinceToken is the method that the protocol client SHOULD use to download changes that have happened since the protocol client's last download on any protocol server that supports it. **GetListItemChanges** or **GetListItems** MAY be used instead.<7>

The details of **GetListItemChangesSinceToken** are specified in [MS-LISTSWS]. This document only covers the usage of particular elements and attributes of **GetListItemChangesSinceToken** for this protocol.

Protocol clients fill in the arguments for **GetListItemChangesSinceToken** as follows.

listName: Protocol clients can fill this with a list identifier. See [MS-LISTSWS].

viewName: Protocol clients can include this element or omit it. See [MS-LISTSWS].

query: Defines additional restrictions on the data the protocol client wants from the protocol server.

The following schema defines the values that protocol clients SHOULD use within the **query** element to get the protocol server behavior described in this document. [MS-LISTSWS] defines additional ways that clients can use the **query** element.

query.Query: This element is used to sort items. The following WSDL describes how the **query.Query** element is used by protocol clients.

```
<complexType name="OrderByQuery">
  <sequence>
    <element name="OrderBy" type="OrderByIDQuery" minOccurs="1" maxOccurs="1" />
  </sequence>
</complexType>
```

query.Query.OrderBy: This element is used to sort items by the ID field, in descending order. The ID field is specified in section 3.2.4.2.1. Because the ID field holds unique integers assigned in increasing order, the result of this sort is that items created earlier are sorted to the top of the list of results. The following WSDL describes how the **query.Query.OrderBy** element is used by protocol clients.

```
<complexType name="OrderByIDQuery">
  <sequence>
    <element name="FieldRef" type="string" fixed="" minOccurs="1" maxOccurs="1">
      <complexType>
```

```
          <simpleContent>
            <extension base="string">
              <attribute name="Name" type="string" fixed="ID" use="required" />
              <attribute name="ASCENDING" type="string" fixed="FALSE" use="required" />
            </extension>
          </simpleContent>
        </complexType>
      </element>
    </sequence>
  </complexType>
```

viewFields: Protocol clients use **viewFields** for the following purposes:

- An attribute **Properties** on **viewFields** SHOULD be set to TRUE as follows:

```
<viewFields Properties="TRUE">
```

This attribute tells the protocol server to include **property bag (1)** fields. If protocol clients are not interested in those fields, then they SHOULD NOT include the **Properties** attribute.

1. The **viewFields** element SHOULD contain a series of **FieldRef** elements, one for each field the protocol client requests to have in the results. See [MS-LISTSWS] for more details.
2. **viewFields.ViewFields** SHOULD specify the following:

```
<FieldRef Name="PermMask" />
```

This tells the protocol server to provide as many results as the protocol client has permission to get. Without it, the protocol server SHOULD reject the entire request if the protocol client does not have permission to one item.

- **viewFields.ViewFields** SHOULD specify the following:

```
<FieldRef Name="MetaInfo" />
```

if the protocol client requests to receive property bag (1) fields. See section 3.2.4.2 .

```
<complexType name="ViewFields">
  <attribute name="Properties" type="string" use="optional" />
  <sequence>
    <element name="FieldRef" type="FieldRef" fixed="" minOccurs="0"
             maxOccurs="unbounded">
      <complexType>
        <simpleContent>
          <extension base="string">
            <attribute name="Name" type="string" use="required" />
          </extension>
        </simpleContent>
      </complexType>
    </element>
  </sequence>
</complexType>
```

viewFields.Properties: Protocol clients who request to receive property bag (1) fields MUST set this attribute to TRUE as follows:

```
<viewFields Properties="TRUE">
```

viewFields.FieldRef: Specifies a field the protocol client requests to get in **GetListItemChangesSinceTokenResponse** via the **viewFields.FieldRef.Name** attribute.

viewFields.FieldRef.Name: The name of a field.

rowLimit: Protocol clients who request to receive all items in one response MUST omit this element. Protocol clients who request to handle receiving items in several batches MUST set this to a number. Protocol clients can set this to any number allowed by [MS-LISTSWS], but are encouraged to set this to either 100 or a number between 15 and 1000, exclusive<8>.

queryOptions: Protocol clients can use **queryOptions** with **GetListItemChangesSinceToken**, according to their needs (see [MS-LISTSWS] for details). The following table is a list of elements used by this document.

GetListItemChangesSinceToken.queryOptions.QueryOptions.
<ViewAttributes Scope="RecursiveAll" />
<DateInUtc>TRUE</DateInUtc>
<IncludePermissions>FALSE</IncludePermissions>
<ExpandUserField>TRUE</ExpandUserField>
<IncludeAttachmentUrls>TRUE</IncludeAttachmentUrls>
<IncludeAttachmentVersion>TRUE</IncludeAttachmentVersion>
<MeetingInstanceID>-1</MeetingInstanceID>
<OptimizeLookups>TRUE</OptimizeLookups>
<RecurrenceOrderBy>TRUE</ RecurrenceOrderBy>
<RecurrencePatternXMLVersion>v3</ RecurrencePatternXMLVersion >
<OptimizeFor>FolderUrls</OptimizeFor>

queryOptions.OptimizeLookups: Protocol clients can include this element to tell the protocol server that it is ok to use an alternate implementation that reduces server load. Protocol servers can ignore this element. Including this element MUST NOT change the contents of a successful protocol server response. Including this element can result in queries receiving a successful response instead of an error because of an overloaded protocol server.

queryOptions.RecurrenceOrderBy: See [MS-LISTSWS] for details on how the server sorts results when this is present.

queryOptions.RecurrencePatternXMLVersion: This tells the protocol server that the protocol client requests to receive recurrence **XML** for certain types of recurrences. These types include the following<9>:

- Daily recurrences whose definition uses the daily.weekday attribute in its **RepeatPattern** (section 2.2.4.5).
- Monthly recurrences whose definition uses the monthlyByDay.day, monthlyByDay.weekday, or monthlyByDay.weekend_day attributes in its **RepeatPattern** (section 2.2.4.5).
- Yearly recurrences whose definition uses the yearlyByDay element in its **RepeatPattern** (section 2.2.4.5).

queryOptions.OptimizeFor: Protocol clients SHOULD use **OptimizeFor** to help the protocol server perform better. Protocol servers SHOULD respect **OptimizeFor**, but can ignore it instead. The following line:

```
<OptimizeFor>FolderUrls</OptimizeFor>
```

tells the protocol server that it SHOULD optimize for a sort, query, or restriction that depends on the FileRef property or any URL properties it is based on.

See [MS-LISTSWS] for information about **queryOptions.OptimizeFor**.

changeToken: Protocol clients SHOULD supply a valid **change token** in this field. If a protocol client does not have a valid change token, then the protocol client SHOULD omit this element. Protocol clients receive a change token as part of **GetListItemChangesSinceTokenResponse**.

contains: This element is used when querying for items with data that contains certain values. See the **CamlContains** type in [MS-LISTSWS].

contains.Contains: This element is used when querying for items with data that contains certain values.

```
<complexType name="ContainsQuery">
  <choice>
    <element name="Participants" type="positiveInteger" minOccurs="1" maxOccurs="1" />
  </choice>
</complexType>
```

contains.Contains.Participants: This element is only used when a user identifier is known for the protocol client. That user identifier is the value in the **Participants** element.

BeginsWith: This element SHOULD be used with document type items to limit the results to a particular document or folder. Protocol clients can decide to limit the results this way or omit this and receive all items.

```
<complexType name="BeginsWithPathQuery">
  <sequence>
    <element name="FieldRef" type="string" fixed="" minOccurs="1" maxOccurs="1">
      <complexType>
        <simpleContent>
          <extension base="string">
            <attribute name="Name" type="string" fixed="FileRef" use="required" />
          </extension>
        </simpleContent>
      </complexType>
    </element>
    <element name="Value" type="string" minOccurs="1" maxOccurs="1">
```

```
        <complexType>
          <simpleContent>
            <extension base="string">
              <attribute name="Type" type="string" fixed="Note" use="required" />
            </extension>
          </simpleContent>
        </complexType>
      </element>
    </sequence>
  </complexType>
```

BeginsWith.FieldRef: This element specifies the name of the field that contains the value being compared.

BeginsWith.Value: This element specifies what the contents of the field MUST be to qualify as a match. As an example, if the following line is specified,

```
<Value Type="Note">example</Value>
```

then only fields whose value begins with "example" will match. Protocol clients SHOULD set the contents of **Value** to a subsite URL or the path to whatever file or folder the protocol client requests.

3.1.4.7.1 Messages

3.1.4.7.1.1 GetListItemChangesSinceTokenResponse

GetListItemChangesSinceTokenResponse (see [MS-LISTSWS]) includes a lot of information that is useful to protocol clients. If the schema is included, then protocol clients SHOULD process all of the information from the schema that they do from **GetList (**section 3.1.4.5**)**. For information about when and how the schema is included in **GetListItemChangesSinceToken**, see [MS-LISTSWS] on the element **GetListItemChangesSinceTokenResponse.GetListItemChangesSinceTokenResult.listitems.Changes.List**.

In addition to the schema and other information already noted, the following attributes and elements are referenced in other parts of this document. Each of these is contained within **GetListItemChangesSinceTokenResponse.GetListItemChangesSinceTokenResult.listitems**, as specified by [MS-LISTSWS].

Name	Purpose and Section
AlternateUrls	Alternate URLs. See section 3.2.4.1.
Changes.Id.ChangeType	See section 3.2.6.2.
Changes.LastChangeToken	See section 3.2.4.1.
Changes.MoreChanges	See section 3.2.4.1.
data.ListItemCollectionPositionNext	See section 3.2.4.1.
EffectivePermMask	Specifies permissions. See section 3.2.6.3.
MaxBulkDocumentSyncSize	See section 3.2.4.2.6.

Name	Purpose and Section
MinTimeBetweenSyncs	See section 3.2.4.1 and section 3.2.5.
RecommendedTimeBetweenSyncs	See section 3.2.4.1 and section 3.2.5.

3.1.4.8 HTTP GET

Protocol clients SHOULD send the following header with their HTTP GET:

```
Translate: f
```

Some protocol servers support a feature that displays the requested file as HTML in response to HTTP GET. The "Translate: f" header tells the protocol server that the protocol client requests the actual file and not an HTML view of it<10>.

The status code header is used for error detection. The last-modified header is useful if protocol clients request to make use of the **CheckOutFile** WSDL operation specified in [MS-LISTSWS]. Protocol clients can tell if the file has been updated by comparing the modified times.

If the protocol server includes an ETag header in the response, then protocol clients can use its value with the IF-Match header in section 3.1.4.9.

3.1.4.9 HTTP PUT

When overwriting files that are already on the protocol server, protocol clients SHOULD add the following header:

```
IF-Match: <a string>
```

Protocol clients MUST replace "<a string>" with data for comparison with the file's current value. The required data is the file version from the "Attachments" item property listed in section 3.2.4.2.1 in **GetListItemChangesSinceTokenResponse**. The **Attachments** property MUST contain a file version if protocol clients have included the **IncludeAttachmentUrls** and **IncludeAttachmentVersion** elements specified in [MS-LISTSWS].

Here is an example of the **Attachments** value:

```
ows_Attachments=";#http://example/Lists/Calendar/Attachments/14/MiddleMan.log;#{03b04af0-
69a4-4fea-990f-7ee608068931},2;#"
```

The data that belongs in the If-Match header is the text after the attachment URL and between the ";#" separators: "{03b04af0-69a4-4fea-990f-7ee608068931},2". Protocol clients SHOULD treat this data as opaque and not parse it. See [MS-LISTSWS] for the format of **Attachments**. The final header becomes:

```
IF-Match: "{03b04af0-69a4-4fea-990f-7ee608068931},2"
```

The purpose of the IF-Match header is to prevent files from being overwritten if they have changed since the time that they were downloaded. Protocol servers MUST respond with an HTTP status code 409 (which indicates a conflict) if the header data does not match the current version and some other issue does not require a different error code.

3.1.4.10 UpdateListItems

This operation is used to add items that are not discussion items and to update items.

The details of **UpdateListItems** are specified in [MS-LISTSWS]. This document only covers the usage of particular elements and attributes of **UpdateListItems** for this protocol.

3.1.5 Timer Events

This protocol uses the same timer behavior for protocol servers as [MS-LISTSWS].

3.1.6 Other Local Events

This protocol has the same behavior as [MS-LISTSWS] for this section.

3.2 Client Details

3.2.1 Abstract Data Model

Lists can be one of five types. Each type shares a common schema. The schema for each type is defined in this section.

List type	Item type name
Calendar	Events and appointments.
Contact	Contacts.
Discussion	Discussion items and posts.
Document	Documents and folders.
Tasks	Tasks.

3.2.1.1 Appointments

Appointment items and event items are synonymous and both terms are interchangeable.

At a minimum, an appointment MUST have a starting date and time. Appointments SHOULD have a duration value and an ending date and time. If an appointment lacks either a duration value or an ending date, or both, then protocol clients MUST calculate the missing value from the other. If both are missing, then the protocol client SHOULD assume the duration to be zero, but can assume any duration. Duration MUST be positive or zero. The ending date and time MUST NOT precede the starting date and time.

Appointments MUST be one of four types:

- Single.
- Recurring.
- An exception to a recurrence.
- A deleted instance of a recurrence.

Appointments MUST be either all-day or not. All-day appointments are all-day in every time zone and are 24 hours long on every day they occur. All-day appointments MUST begin at midnight, 0 hours UTC. Two users in different time zones looking at an all-day appointment SHOULD both see the appointment beginning at midnight in their time zone even though midnight is at a different UTC time in each time zone. Because of this, all-day appointments do not have time zones and it is up to clients to choose a time zone for displaying all-day appointments to users.

3.2.1.1.1 Single Appointments

A single appointment is not recurring or any type of exception.

3.2.1.1.2 Recurring Appointments

In addition to a starting date and time, recurring appointments have a recurrence definition and a time zone definition that specify the type of recurrence and when it occurs.

The last instance of recurring appointments is defined by one of the following:

- A date-time.
- A count of instances.
- The recurrence does not end.

Each instance of the recurrence MUST begin and end at the same time as the starting date and time and ending date and time. When showing appointments in the user interface, protocol clients can adjust these times to conform to the time zone definition of the recurrence.

Any instance of a recurrence SHOULD have zero or one total exceptions and deleted instances<11>. If an instance has more than one then the protocol client SHOULD choose one and ignore the other(s). Protocol clients can choose one and display the other(s) as single appointments.

3.2.1.1.3 Exceptions to a Recurrence

Exceptions are a single appointment that overrides one instance of a recurring appointment. The starting date and time of the overridden instance is the replacement date-time of the exception. Exceptions are in the time zone of the recurrence they belong to.

Any property of a recurrence that is missing from an exception is assumed to have the same value as other instances of the recurrence. This means an exception with no location to a recurrence with location="xyz" has a location "xyz". All properties of an exception override properties of a recurrence. This means if an exception has a different starting date and time than the instance it replaces, the protocol client MUST use the exception's starting date and time for that instance.

When a recurrence is deleted, all exceptions to that recurrence SHOULD also be deleted. Protocol clients MAY instead choose to convert or display them as other appointment types.<12>.

Exception items SHOULD only be converted to deleted instance items. Exception items MAY be deleted to restore the original recurring instance.<13> Exception items can also be converted to single appointment items, in which case the protocol client decides whether the replaced instance can be restored, an exception, or deleted by uploading a replacement exception item.

3.2.1.1.4 Deleted Instances of a Recurrence

Deleted instances are exceptions to a recurrence (section 3.2.1.1.3). Protocol clients SHOULD NOT display the deleted instance items<14> or the instance of the recurrence it replaces.

Users might not understand that an item still exists for a deleted instance. Protocol clients can help users by doing the following when deleting an instance of a recurrence:

When creating a deleted instance from an exception, protocol clients SHOULD delete the exception and create a new deleted instance to replace it. If protocol clients change the exception's event type to make it a deleted instance, they SHOULD remove as much information as possible from the exception. Protocol clients MAY leave information about the exception or add more information.<15>

When creating a deleted instance from a recurrence instance, protocol clients MUST create a new deleted-instance item for that instance of the recurrence. The deleted instance can have any information the protocol client requests, but SHOULD have the minimum amount necessary for protocol clients to process the item.

3.2.1.2 Contacts

Contacts SHOULD have some information about a person, group, business, or other entity.

3.2.1.3 Discussions

Conceptually, discussion items often represent posts to a newsgroup, e-mail messages, bulletin boards, or e-mail archives. The exact context is up to the protocol client.

Discussion items can be one of two types:

- Root items chosen by the protocol server.
- All other items.

Root items SHOULD be the first item the protocol server received on an e-mail thread, but can be arbitrarily chosen. Root items SHOULD be replies to earlier posts on a tread. Every thread MUST have exactly one root item. When a root item is deleted, all posts on that thread MUST be deleted too.

All non-root items MUST belong to a root item. Deleting a non-root item SHOULD only delete that item, but can do whatever a protocol client requests.

Root items are items where the **ContentTypeId** indicates the item is a folder. Section 3.2.4.2.1 specifies the values of **ContentTypeId**.

3.2.1.4 Documents

Conceptually, a document item is similar to a file in a file system. The document item MUST include information such as the file name, size, and how or where to get the file itself.

There are three things that specify a document:

- A document item exists in a folder item. If the folder item is deleted, then all documents and folders in it MUST be deleted too.
- Document items SHOULD have exactly one file. Document items MUST have at least one file.
- Document items SHOULD have information describing the attachment. What information is included is specified in section 3.2.4.2.6.

3.2.1.5 Tasks

Conceptually, tasks are a way to track work, activities, or steps in a process. Tasks might have properties to describe how much work has been done, who is supposed to be working on the task, or when the task needs to be finished.

Tasks SHOULD have a title, due date, and something to indicate if the task is complete or not. What data is included with a task is an implementation detail. Section 3.2.4.2.8 specifies several task properties that protocol clients can use or ignore.

3.2.2 Timers

Timer	Description
TimeElapsedSinceLastSync	The period of time that has passed since the last time this protocol was performed. This is ignored if the protocol client has not performed the protocol before. Each list gets a copy of this timer.

3.2.3 Initialization

At the beginning of the protocol, the timers are in the following states:

TimeElapsedSinceLastSync is ignored if protocol client has not performed this protocol on the current list, so protocol clients are free to choose an initial value in that case. If the protocol client has performed this protocol with the current list, then **TimeElapsedSinceLastSync** is the time elapsed since the last time the protocol finished work on the current list.

3.2.4 Message Processing Events and Sequencing Rules

3.2.4.1 State Diagram

The following state diagram illustrates how a protocol client SHOULD download items from a protocol server using this protocol. The first decision the protocol client makes is whether **TimeElapsedSinceLastSync** is large enough. **TimeElapsedSinceLastSync** is large enough if it is greater than both the **MinTimeBetweenSyncs** and **RecommendedTimeBetweenSyncs** attributes of the **GetListItemChangesSinceTokenResponse.GetListItemChangesSinceTokenResult.listitems** element specified in [MS-LISTSWS]. Protocol clients SHOULD NOT ignore **MinTimeBetweenSyncs**, but protocol clients can ignore **RecommendedTimeBetweenSyncs**<16>.

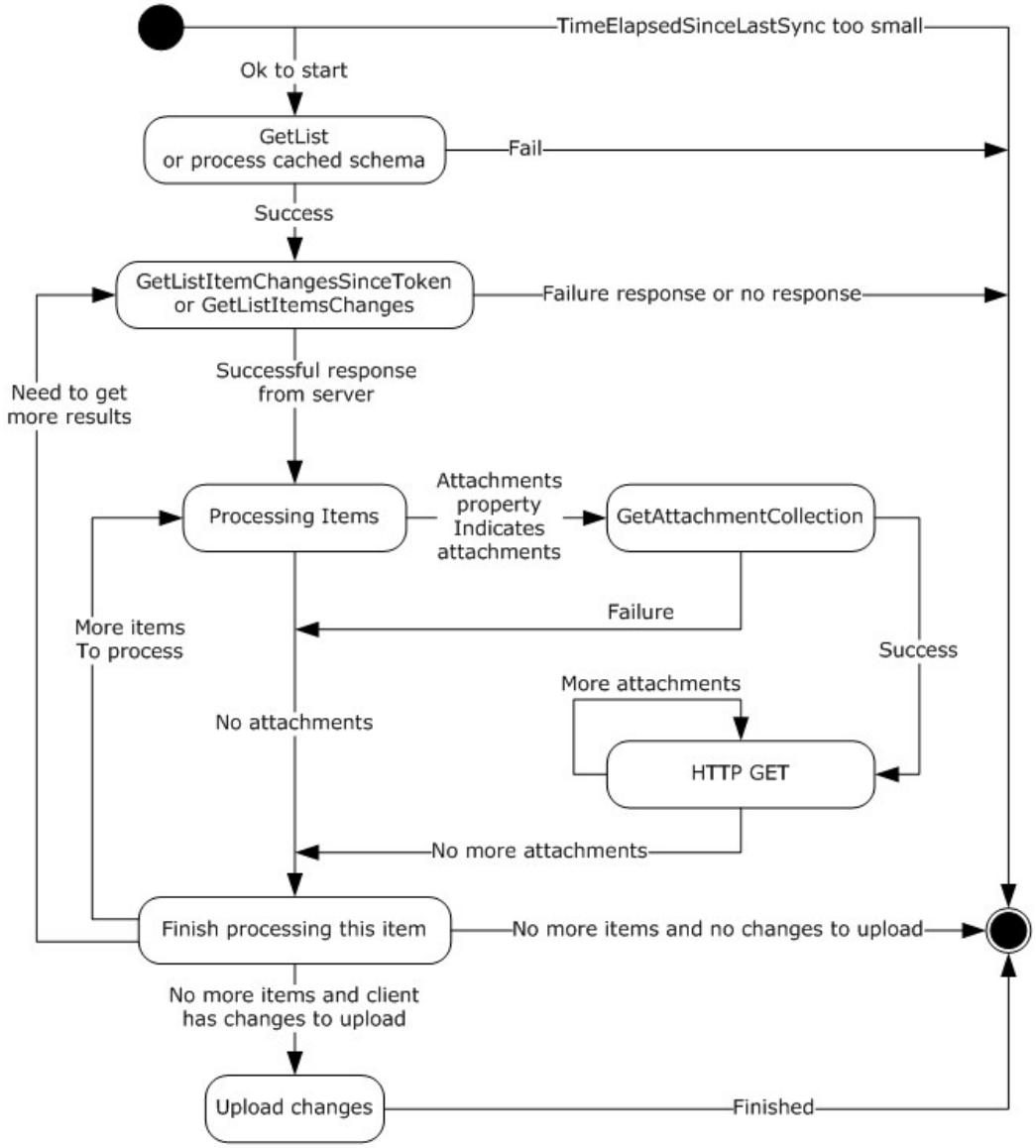

Figure 2: State diagram showing the sequence of decisions and operations performed by a protocol client

The "Need to get more results" transition in the diagram refers to when protocol clients download items in batches. See [MS-LISTSWS] on **GetListItemChangesSinceToken.rowLimit** and the following contents of **GetListItemChangesSinceTokenResponse.GetListItemChangesSinceTokenResult.listitems**.

Name	Purpose and section
Changes.data.ListItemCollectionPositionNext	See section 3.2.4.1
Changes.LastChangeToken	See section 3.2.4.1

Name	Purpose and section
Changes.MoreChanges	See section 3.2.4.1

The following state diagram specifies the details of the "Upload changes" state shown in the previous diagram showing the sequence of decisions and operations performed by a protocol client.

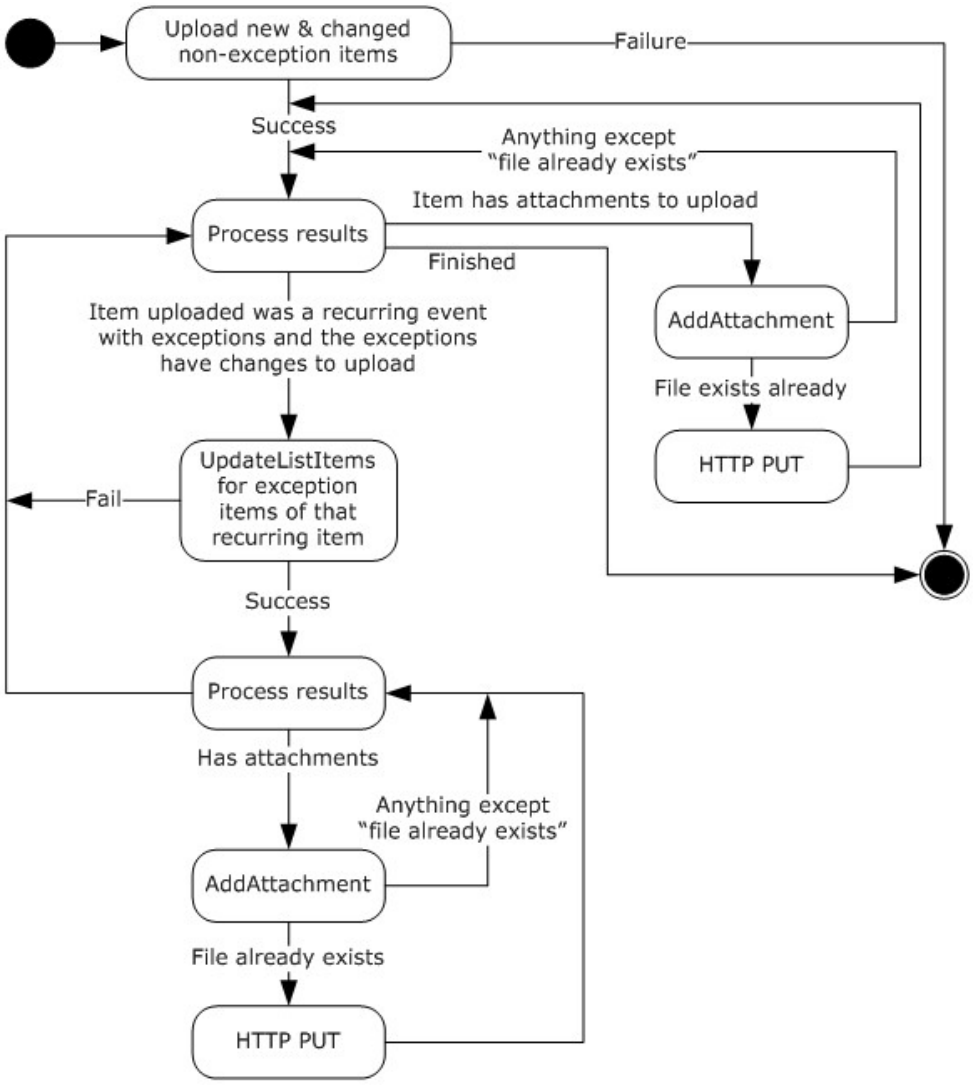

Figure 3: State diagram detailing the sequence of decisions and operations performed by a protocol client in the "Upload changes" state

The following diagram specifies the details of the "Upload new and changed non-exception items" state shown in the previous diagram detailing the sequence of decisions and operations performed by a protocol client in the "Upload changes" state. Non exception items are every item type that is not an exception to a recurrence (section 3.2.1.1).

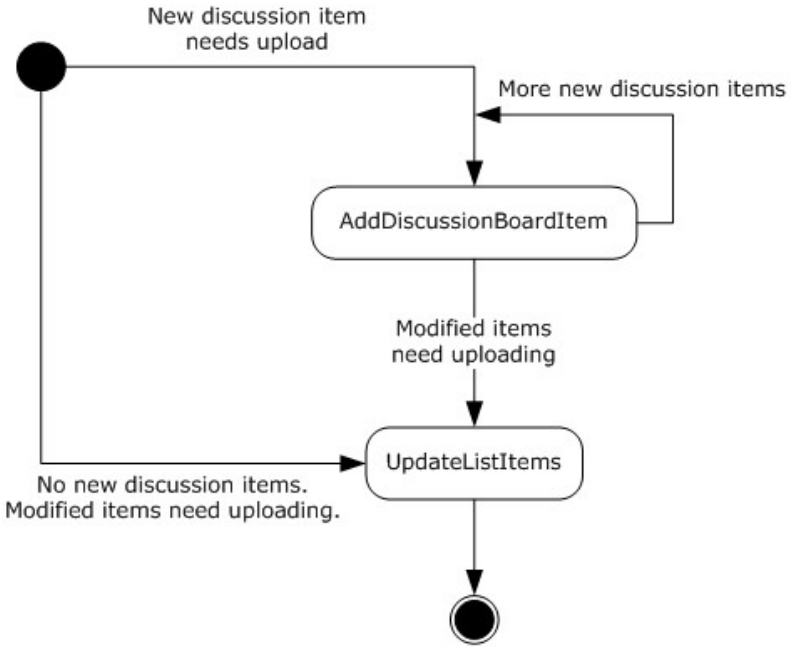

Figure 4: State diagram detailing the sequence of decisions and operations performed by a protocol client in the "Upload new and changed non-exception items" state

In addition, if a protocol client receives an error from a remote operation that indicates a connection loss, the protocol client SHOULD select an alternate URL from the **GetListItemChangesSinceTokenResponse.GetListItemChangesSinecTokenResult.listitems.AlternateUrls** attribute (see [MS-LISTSWS]). The protocol client SHOULD then restart the sync process using the new URL. If an operation on that URL has failed recently, then protocol clients SHOULD NOT try to use that URL again. The definition of "failed recently" is up to each individual protocol client and is not specified by this protocol.

3.2.4.2 Schema of Each Item Type

Each item has a schema that is associated with it. This schema is the union of two sets of fields:

- Common schema.
- Type-specific schema.

In the sections that follow, each row in a table identifies a field by a group of attributes that appear in **GetListResponse.GetListResult.List.Fields.Field** (see [MS-LISTSWS]). These attributes are:

Field.ID: The ID column of the table corresponds to this attribute. If a field does not have a defined identifier, then the ID will be listed as "None defined".

Field.Name: The Name column of the table corresponds to this attribute.

Field.Type: The Type column of the table corresponds to this attribute. If a field does not have a defined type, the type will be listed as "None defined" and an "interpret as" type will be given. Clients SHOULD interpret the value as the type given in this document if the server does not specify a type.

After each table is a section that lists each field, a description of the field's purpose, and additional information about how to interpret the field's data.

When trying to determine if any particular field is present in the current schema, protocol clients SHOULD follow this step by step rule while parsing for schema:

1. If the **Field.ID** matches, the field is present in the current schema. If it does not match, proceed to step 2.
2. If the **Field.Name** matches, the field is present in the current schema. If it does not, the field is not in the current schema.

In addition, protocol clients SHOULD validate that the type of the field is something they can handle. The following table gives an example of a set of rules for compatibility. In the table, anything stored as text is compatible with any type because no conversion is required to or from XML. Entries enclosed in quotes are valid values for the **Field.Type** attribute. When the server type is not a recognized type, this can indicate a customized scenario on the protocol server. Protocol clients SHOULD assume unknown types are compatible if the field is in the schema.

Client type	Server type	Compatibility
"Text"	Any type.	Yes, protocol server chooses format.
Any type	"Text"	Yes, protocol client chooses format.
Any type	Unknown type.	Protocol clients SHOULD say Yes.
Same as server	Same as client.	Yes, same format used on both protocol client & protocol server.
"Integer"	"Boolean"	Yes, {nonzero, 0}={true,false}.
"Boolean"	"Integer"	Yes, {true,false}={nonzero, 0}.

In this example, any combination not listed in the table would not be compatible. For example, binary data and "Integer" data would not be compatible.

Protocol clients can implement a different table of compatible types if the way the client stores data requires it. Protocol clients can have data types that protocol servers do not. Such types would require a conversion to a type the protocol server has and protocol clients can choose not to support converting types.

When the protocol server does not list a field in the current server schema, protocol clients SHOULD get the field value from the property bag (1) fields<17><18>, but protocol clients can ignore the property bag (1) fields. Getting and setting an item's values for these fields works the same as fields in the server schema, except that protocol clients need to make the following changes. This example uses a field named "Example" with value "value". "MetaInfo" is a constant string. See [MS-LISTSWS] for details.

Operation	In server schema	Not in server schema
UpdateListItems	`<Field Name="Example">value</Field>`	`<Field Name="MetaInfo" Property="Example">value</Field>`
GetListItemChangesSinceToken	`ows_Example`	`ows_MetaInfo_Example`

When a property has values in both the server schema and the property bag (1) fields, the server schema is used and the property bag (1) field value is ignored.

3.2.4.2.1 Common Schema

Common fields are fields that appear on all item types and whose interpretation does not change depending on the type of the item. Unless stated otherwise, all fields in this section MUST be present on all item types<19> and contain valid data. The following table identifies a field by a group of attributes that appear in **GetListResponse.GetListResult.List.Fields.Field** (see [MS-LISTSWS] and section 3.2.4.2.

Field.Name	Field.ID	Field.Type (interpret as)
Attachments	{67df98f4-9dec-48ff-a553-29bece9c5bf4}	Attachments
Categories	{9EBCD900-9D05-46c8-8F4D-E46E87328844}	Text
ContentTypeId	{03e45e84-1992-4d42-9116-26f756012634}	ContentTypeId
Created	{8c06beca-0777-48f7-91c7-6da68bc07b69}	DateTime
ID	{1d22ea11-1e32-424e-89ab-9fedbadb6ce1}	Counter
Modified	{28cf69c5-fa48-462a-b5cd-27b6f9d2bd5f}	DateTime
owshiddenversion	{d4e44a66-ee3a-4d02-88c9-4ec5ff3f4cd5}	Integer
ReplicationID<20>	None defined	None defined (see **stringGUID** (section 2.2.5.10))
vti_versionhistory<21>	None defined	None defined (see description following the table)

Attachments: See **IncludeAttachmentUrls** in [MS-LISTSWS].

Categories: This is a delimited list of strings stored as a single string. The delimiter used is an implementation detail and can vary depending on the language settings of the user. Suggested delimiters include the semicolon ';' and comma ',' characters. **Categories** can be absent from items and can be absent from the schema.

ContentTypeId: This property identifies the item type of each item. The format of **ContentTypeId** is defined in [MS-LISTSWS]. The following table shows how to identify an item type by the value in the **ContentTypeId** field.

ContentTypeId begins with	Content or item type name
0x0102	Appointment
0x0106	Contact
0x0116	Contact

ContentTypeId begins with	Content or item type name
0x0107	Discussion item
0x0101	Document
0x0120	Folder
0x0108	Task
0x010801	Workflow Task (a subtype of Task)

Protocol clients MAY choose to ignore **ContentTypeId** and assume all items in a list are a particular type.<22>

Created: The date and time the item was created. **Created** can be absent from items and can be absent from the schema.

ID: An integer that uniquely identifies this item from all other items in the list. Protocol clients are not allowed to change the identifier. IDs are assigned by the protocol server, so protocol clients also cannot set an identifier on a new item.

Modified: Date and time that the item was last modified. For document items, protocol clients MUST NOT assume that this time is the same as the last modified time of the document file. **Modified** can be absent from items and can be absent from the schema.

owshiddenversion: This is an integer that increases by 1 every time the item is modified on the protocol server. See [MS-LISTSWS].

ReplicationID: This property is of the simple type stringGUID. Protocol clients SHOULD generate a GUID when creating new items on the protocol server and fill the **ReplicationID** property with it. This allows a protocol client to correctly identify the item later if the creation succeeds but the protocol client receives no response from the protocol server. Protocol servers can choose to set **ReplicationID** on new items created by the protocol server. **ReplicationID** can be absent from items and the **ReplicationID** field can be absent from the schema.

vti_versionhistory: Version history is a list of GUIDs and integers in the following format:

GUID:Integer,GUID:Integer,GUID:Integer

There can be any nonzero number of GUID-integer pairs in the list. Each unique GUID MUST appear at most once in the list. One integer MUST be greater than all other integers. The list is not sorted, so any GUID can be the one paired with the highest integer. Protocol clients and protocol servers MAY remove the pairs with the smallest integers to save space.<23> GUIDs are written as a hexadecimal string with no non-hexadecimal characters.

Version history is built in the following way:

1. Every protocol client and protocol server that edits items generates a GUID. This GUID is reused and not regenerated.

2. Each time an item is updated, the highest integer is found among the GUID-integer pairs. This is the version number. That integer is incremented by one to get the integer to use in step 3.

3. The editing protocol client or protocol server searches for its GUID in the GUID-integer pairs and removes the pair if it is found. Then the editing protocol client or protocol server inserts a new GUID-integer pair using its GUID and the integer from step 2.

The following is an example of version history:

```
de97ebfb2cab394bb8907df29fbd9a71:4,c35a52c8cb6a9c40bcee84c0865d4622:7,508a32fb9f9f4ccd8e5663b
4172b4546:9
```

Protocol clients and protocol servers can use the version history to perform conflict detection and resolution, as explained in the following table.

Highest-integer comparison between two values of version history	Result
Different GUIDs	Conflict: items are different
Different integers	Conflict: items are different
Same integer and same GUID	Same item

In these conflict cases, if the GUID-integer pair with the lower integer is found in the version history with the higher integer, then the lower integer item is a previous state of the higher integer item. Protocol clients SHOULD use this to do more sophisticated conflict resolution than a simple equality comparison. Protocol clients SHOULD choose to overwrite the older version instead of reporting a conflict.

3.2.4.2.2 Appointment-Specific Schema

The appointment schema implements the appointments (section 3.1.1) abstract data model. All appointment properties can be empty or missing unless this section states otherwise.<24> The following table identifies a field by a group of attributes that appear in **GetListResponse.GetListResult.List.Fields.Field** (see [MS-LISTSWS] and section 3.2.4.2).

Field.Name	Field.ID	Field.Type (interpret as)
BusyStatus	None defined	None defined (Busy Status)
Description	{9da97a8a-1da5-4a77-98d3-4bc10456e700}	Note
Duration	{4d54445d-1c84-4a6d-b8db-a51ded4e1acc}	Integer
Editor	{d31655d1-1d5b-4511-95a1-7a09e9b75bf2}	User
EndDate	{2684f9f2-54be-429f-ba06-76754fc056bf}	DateTime
EventDate	{64cd368d-2f95-4bfc-a1f9-8d4324ecb007}	DateTime
EventType	{5d1d4e76-091a-4e03-ae83-6a59847731c0}	Integer
fAllDayEvent	{7d95d1f4-f5fd-4a70-90cd-b35abc9b5bc8}	AllDayEvent.
FollowUp	None defined	None defined (FollowUp).
FooterInfo	None defined	None defined (Text).
fRecurrence	{f2e63656-135e-4f1c-8fc2-ccbe74071901}	Recurrence.
HeaderInfo	None defined	None defined (Text).
IntendedBusyStatus	None defined	None defined (Busy Status).

Field.Name	Field.ID	Field.Type (interpret as)
Location	{288f5f32-8462-4175-8f09-dd7ba29359a9}	Text.
MasterSeriesItemID	{9b2bed84-7769-40e3-9b1d-7954a4053834}	Integer.
Participants	None defined.	None defined (Participants).
Priority	None defined.	None defined (Priority).
RecurrenceData	{d12572d0-0a1e-4438-89b5-4d0430be7603}	Note.
RecurrenceID	{dfcc8fff-7c4c-45d6-94ed-14ce0719efef}	DateTime.
TimeZone	{6cc1c612-748a-48d8-88f2-944f477f301b}	Integer.
Title	{fa564e0f-0c70-4ab9-b863-0177e6ddd247}	Text.
UID	{63055d04-01b5-48f3-9e1e-e564e7c6b23b}	GUID.
XMLTZone	{c4b72ed6-45aa-4422-bff1-2b6750d30819}	Note.

BusyStatus: A **BusyStatus** (section 2.2.5.1) type. This busy status specifies what busy status the last **Editor** of the event wanted.

Description: Any user entered text about the appointment.

Duration: The duration of the appointment, in seconds. **Duration** can be empty or missing.

Duration can be calculated from the **EventDate** and **EndDate**. The following algorithm MUST be used to do this:

1. If the **fAllDayEvent** property is 1:

   - The **Duration** MUST be 86340 seconds (1 minute less than 24 hours).

2. The **fAllDayEvent** property is 0:

   1. Change both the **EventDate** and **EndDate** into a time zone with no DST transitions, such as UTC.

   2. Remove the date portion of the results from step 2.1 so only the time portion remains.

   3. Subtract the **EventDate** value from step 2.2 from the **EndDate** value from step 2.2. The result is the duration.

Editor: The name of the last person who made changes to the item.

EndDate: The ending date and time of the appointment. **EndDate** MUST be equal to or later than **EventDate**. **EndDate** can be empty or missing. If **EndDate** is not provided, then protocol clients SHOULD follow the guidelines in section 3.2.1.1.

EventDate: The starting date and time of the appointment. **EventDate** MUST be equal to or earlier than **EndDate**. **EventDate** MUST NOT be empty or missing. If the **fAllDayEvent** property is 1 then the time portion of the **EventDate** MUST be 0 hours UTC as in this example: "2009-05-19T00:00:00Z".

EventType: An **EventType** (section 2.2.5.5) integer. **EventType** MUST NOT be empty or missing. If the **EventType** is one that the protocol client does not support, then the protocol client SHOULD treat it as a single instance event.

If the **EventType** indicates a recurring event, then **fRecurrence** MUST be 1. Otherwise **fRecurrence** MUST be "0". If **EventType** indicates a recurring event and **fRecurrence** is FALSE, then the item is not a recurring event.

fAllDayEvent: A **booleanInteger** value that specifies whether the appointment is an all-day appointment, as specified in the appointments (section 3.2.1.1) abstract data model. **fAllDayEvent** SHOULD NOT be empty or missing<25>. If it is, then the default value is 0, which means FALSE. 1 means TRUE.

FollowUp: A **FollowUp** (section 2.2.5.6) integer.

FooterInfo: Any user entered text that is meant to appear after the **Description** text.

fRecurrence: A **booleanInteger** value that specifies whether the **EventType** value indicates a recurring event or an exception. 1 means it is recurring, 0 means it is not. **fRecurrence** MUST NOT be empty or missing.

HeaderInfo: Any user-entered text that is meant to appear before the **Description** text.

IntendedBusyStatus: A **BusyStatus** (section 2.2.5.1**)** type. This property specifies the busy status the creator of the event wanted.

Location: Any text specifying where the event is supposed to take place.

MasterSeriesItemID: This exists only for exception items. This is the item identifier (see the ID property in section 3.2.4.2.1 of the recurring item that the exception belongs to).

Participants: A **Participants** (section 2.2.5.8) string.

Priority: A **Priority** (section 2.2.5.9) integer .

RecurrenceData: If **EventType** is 1, this property MUST contain a valid **RecurrenceXML** (section 2.2.4.4). If **fRecurrence** is FALSE, then this property MUST be ignored and can be empty or missing.

RecurrenceID: RecurrenceID MUST be equal to the starting date and time of one instance of a recurrence when the **EventType** indicates an exception or deleted instance. If the **EventType** is something else, then **RecurrenceID** can be empty or missing.

TimeZone: If **fRecurrence** is TRUE, this property SHOULD contain an integer index into a list of time zones. Where this list exists and how to access it is an implementation detail of the protocol server. Protocol servers SHOULD set a number in this value, but can leave it empty. Protocol clients SHOULD remember whatever number the protocol server provides here. Protocol clients SHOULD set the recurrence **TimeZone** integer on exceptions to a recurrence (section 3.2.1.1.3) when protocol clients update exception items.

Protocol clients SHOULD NOT use **TimeZone** for any other purpose. Instead, protocol clients SHOULD use **XMLTZone** because this approach is more extensible and flexible.

If **fRecurrence** is FALSE, then protocol clients SHOULD ignore **TimeZone**.

Title: Any text describing the event. This text is typically shorter than **Description**, but might be longer.

UID: If **fRecurrence** is TRUE, this property MUST contain a valid **stringGUID** (section 2.2.5.10). **UID** is one of the fields used to indicate recurrence changes (section 3.2.4.2.3). **UID** MUST be changed if, and only if, the recurrence has been changed or added. **UID** SHOULD be unique among all other recurring events on this list. The **queryOptions.RecurrenceOrderBy** (section 3.1.4.7) sort order relies on a unique **UID** to provide unambiguous results. If **fRecurrence** is FALSE, then this property MUST be ignored.

XMLTZone: If **EventType** is 1, then this property MUST contain a valid **TimeZoneXML** (section 2.2.4.7). The **TimeZoneXML** defines the time zone the event uses. If **fRecurrence** is FALSE, then this property SHOULD be ignored and can be empty.

If EventType is 1 and fAllDayEvent is 1 then XMLTZone MUST indicate a time zone with no bias or offset:
`<timeZoneRule><standardBias>0</standardBias><additionalDaylightBias>0</additionalDaylightBias></timeZoneRule>`.

3.2.4.2.3 Updating Recurring Appointments

Protocol clients need to be careful when updating a recurring item. Updating properties that describe the recurrence of an item MAY result in the deletion of exception items (section 3.2.1.1.3) and deleted instance items (section 3.2.1.1.4). These properties are specified in section 3.2.4.2.2 and include those in the following list:

- EndDate
- EventDate
- RecurrenceData
- UID
- XMLTZone

Protocol clients MUST include the **deleteExceptions** element of **RecurrenceXML** (section 2.2.4.4) when changing one or more of those properties to cause the protocol server trigger exception deletion. Protocol servers SHOULD trigger exception deletion when one of these properties is updated, so protocol clients SHOULD NOT send any updates to these properties unless the recurrence has changed.<26>

3.2.4.2.4 Contact-Specific Schema

The contact schema implements the contact item abstract data model (section 3.2.1.2). All contact properties can be empty or missing unless this section states otherwise.<27> The following table identifies a field by a group of attributes that appear in **GetListResponse.GetListResult.List.Fields.Field** (see [MS-LISTSWS] and section 3.2.4.2.

Field.Name	Field.ID	Field.Type (interpret as)
Account	None defined.	None defined (Text).
Anniversary<28>	{9D76802C-13C4-484a-9872-D7F9641C4672}	DateTime.
AssistantNumber	{F55DE332-074E-4e71-A71A-B90ABFAD51AE}	Text.

Field.Name	Field.ID	Field.Type (interpret as)
AssistantsName	{2AEA194D-E399-4f05-95AF-94F87B1F2687}	Text.
AttachProps	None defined.	None defined (AttachProps).
BCPicture	None defined.	None defined (base64Binary).
BillingInformation	{4F03F66B-FB1E-4ed2-AB8E-F6ED3FE14844}	Text.
Birthday	{C4C7D925-BC1B-4f37-826D-AC49B4FB1BC1}	DateTime.
BizCard	None defined.	None defined (hexBinary).
Business2Number	{6547D03A-76D3-4d74-9D34-F51B837C0879}	Text.
CallbackNumber	{344E9657-B17F-4344-A834-FF7C056BCC5E}	Text.
CarNumber	{92A011A9-FD1B-42e0-B6FA-AFCFEE1928FA}	Text.
CellPhone	{2a464df1-44c1-4851-949d-fcd270f0ccf2}	Text.
Certificate	None defined.	None defined (base64Binary).
CertificateStr	None defined.	None defined (Text).
CertificatesX509	None defined.	None defined (base64Binary).
ChildrensNames	{6440B402-8EC5-4d7a-83F4-AFCCB556B5CC}	Text.
Comments	{9da97a8a-1da5-4a77-98d3-4bc10456e700}	Note.
Company	{038d1503-4629-40f6-adaf-b47d1ab2d4fe}	Text.
CompanyNumber	{27CB1283-BDA2-4ae8-BCFF-71725B674DBB}	Text.
CompanyPhonetic	{034aae88-6e9a-4e41-bc8a-09b6c15fcdf4}	Text.
ComputerNetworkName	{86A78395-C8AD-429e-ABFF-BE09417B523E}	Text.
ConfDefServerIndex	None defined.	None defined (Text).
ConfServerNames	None defined.	None defined (Text).

Field.Name	Field.ID	Field.Type (interpret as)
ContactLinkName	None defined.	None defined (Text).
CustomerID	{81368791-7CBC-4230-981A-A7669ADE9801}	Text.
Description	None defined.	None defined (Text).
Editor	{d31655d1-1d5b-4511-95a1-7a09e9b75bf2}	User.
Email	{fce16b4c-fe53-4793-aaab-b4892e736d15}	Text.
Email2	{E232D6C8-9F49-4be2-BB28-B90570BCF167}	Text.
Email3	{8BD27DBD-29A0-4ccd-BCB4-03FE70C538B1}	Text.
EmailDisplayAs1	None defined.	None defined (Text).
EmailDisplayAs2	None defined.	None defined (Text).
EmailDisplayAs3	None defined.	None defined (Text).
Fax1	None defined.	None defined (Text).
Fax2	None defined.	None defined (Text).
Fax3	None defined.	None defined (Text).
FaxDisplayAs1	None defined.	None defined (Text).
FaxDisplayAs2	None defined.	None defined (Text).
FaxDisplayAs3	None defined.	None defined (Text).
FileAs	None defined.	None defined (Text).
FirstName	{4a722dd4-d406-4356-93f9-2550b8f50dd0}	Text.
FirstNamePhonetic	{ea8f7ca9-2a0e-4a89-b8bf-c51a6af62c73}	Text.
FreeBusyURL	None defined.	None defined (Text).
FTPSite	{D733736E-4204-4812-9565-191567B27E33}	URL.
FullName	{475c2610-c157-4b91-9e2d-6855031b3538}	Text.
Gender	{23550288-91B5-4e7f-81F9-1A92661C4838}	Choice or Integer (Gender)<29>.
GovernmentIDNumber	{DA31D3C9-F9DA-4c35-88D4-60AAFA4C3F19}	Text.

Field.Name	Field.ID	Field.Type (interpret as)
Hobbies	{203FA378-6EB8-4ed9-A4F9-221A4C1FBF46}	Text.
Home2Number	{8C5A385D-2FFF-42da-A4C5-F6A904F2E491}	Text.
HomeAddress	{8C66E340-0985-4d68-AF03-3050ECE4862B}	Text.
HomeAddressCity<30>	{5AEABC56-57C6-4861-BC12-BD72C30FC6BD}	Text.
HomeAddressCountry<31>	{897ECFD7-4293-4782-B463-BD68440A5FED}	Text.
HomeAddressPostalCode<32>	{C0E4B4C6-6245-4846-8561-B8C6C01FEFC1}	Text.
HomeAddressStateOrProvince<33>	{F5B36006-69B0-418c-BD4A-F25CA7E096BB}	Text.
HomeFaxNumber	{C189A857-E6B0-488f-83A0-F4EE0A3AD01E}	Text.
HomeFreeForm	None defined	None defined (Text).
HomePhone	{2ab923eb-9880-4b47-9965-ebf93ae15487}	Text.
HomePObox	None defined.	None defined (Text).
IMAddress	{4CBD96F7-09C6-4b5e-AD42-1CBE123DE63A}	Text.
Initials	{7A282F86-69D9-40ff-AE1C-C746CF21256B}	Text.
ISDNNumber	{A579062A-6C1D-4ad3-9D5E-035F9F2C1882}	Text.
JobTitle	{c4e0f350-52cc-4ede-904c-dd71a3d11f7d}	Text.
Language	{D81529E8-384C-4ca6-9C43-C86A256E6A44}	Choice.
LastNamePhonetic	{fdc8216d-dabf-441d-8ac0-f6c626fbdc24}	Text.
Location	{288F5F32-8462-4175-8F09-DD7BA29359A9}	Text.
ManagersName	{BA934502-D68D-4960-A54B-51E15FEF5FD3}	Text.
MiddleName	{418C8D29-6F2E-44c3-8955-2CD7EC3E2151}	Text.

Field.Name	Field.ID	Field.Type (interpret as)
Mileage	{3126C2F1-063E-4892-828F-0696EC6E105F}	Text.
NameTitle	None defined.	None defined (Text).
Nickname	{6B0A2CD7-A7F9-41ca-B932-F3BEBB603793}	Text.
Office<34>	{26169AB2-4BD2-4870-B077-10F49C8A5822}	Text.
ol_Department	{C814B2CF-84C6-4f56-B4A4-C766938A97C5}	Text.
OrganizationalIDNumber	{0850AE15-19DD-431f-9C2F-3AFF3AE292CE}	Text.
OtherAddressCity	{90FA9A8E-AAC0-4828-9CB4-78F98416AFFA}	Text.
OtherAddressCountry	{3C0E9E00-8FCC-479f-9D8D-3447CDA34C5B}	Text.
OtherAddressPostalCode	{0557C3F8-60C4-4dfb-B5BA-BF3C4E4386B1}	Text.
OtherAddressPObox	None defined.	None defined (Text).
OtherAddressStateOrProvince	{F45883BC-8733-4b77-AB5D-43613986AA12}	Text.
OtherAddressStreet	{DFF5DFC2-E2B7-4a19-BDE7-76DABC90A3D2}	Text.
OtherFaxNumber	{AAD15EB6-D7FD-47b8-ABD4-ADC0FE33A6BA}	Text.
OtherFreeForm	None defined.	None defined (Text).
OtherNumber	{96E02495-F428-48bc-9F13-06D98BA58C34}	Text.
PagerNumber	{F79BF074-DAF7-4c06-A314-15B287FDF4C9}	Text.
PersonalWebsite	{5AA071D9-3254-40fb-82DF-5CEDEFF0C41E}	URL.
Photo	{1020C8A0-837A-4f1b-BAA1-E35AFF6DA169}	URL.
PrimaryNumber	{D69BCC0E-57C3-4f3b-BBC5-B090EDF21F0F}	Text.
PostAddrID	None defined.	None defined (Text).
Profession	{F0753A13-44B1-4269-82AF-5C34C57B0C67}	Text.

Field.Name	Field.ID	Field.Type (interpret as)
RadioNumber	{D1AEDE4F-1352-48d9-81E2-B10097C359C1}	Text.
ReferredBy	{9B4CC5A9-1119-43e4-B2A8-412C4031F92B}	Text.
SpouseName	{F590B1DE-8E28-4c17-91BC-BF4096024B7E}	Text.
Suffix	{D886EBA3-D018-4103-A322-D5780127EF8A}	Text.
TelexNumber	{E7BE7F3C-C436-481d-8865-669E5146F53C}	Text.
Title	{fa564e0f-0c70-4ab9-b863-0177e6ddd247}	Text.
TTYTDDNumber	{F54697F1-0357-4c5a-A711-0CB654BC73E4}	Text.
UserField1	{566656F5-17B3-4291-98A5-5074AADF77B3}	Text.
UserField2	{182D1B9E-1718-4e11-B279-38F7ED0A20D6}	Text.
UserField3	{A03EB53E-F123-4af9-9355-F92BD75C00B3}	Text.
UserField4	{ADEFA4CA-14C3-4694-B531-F51B706EFE9D}	Text.
WebPage	{a71affd2-dcc7-4529-81bc-2fe593154a5f}	URL.
WorkAddress	{fc2e188e-ba91-48c9-9dd3-16431afddd50}	Note.
WorkCity	{6ca7bd7f-b490-402e-af1b-2813cf087b1e}	Text.
WorkCountry	{3f3a5c85-9d5a-4663-b925-8b68a678ea3a}	Text.
WorkFax	{9d1cacc8-f452-4bc1-a751-050595ad96e1}	Text.
WorkFreeForm	None defined.	None defined (Text).
WorkPhone	{fd630629-c165-4513-b43c-fdb16b86a14d}	Text.
WorkPObox	None defined.	None defined (Text).
WorkState	{ceac61d3-dda9-468b-b276-f4a6bb93f14f}	Text.

Field.Name	Field.ID	Field.Type (interpret as)
WorkZip	{9a631556-3dac-49db-8d2f-fb033b0fdc24}	Text.

Account: Any kind of text string that is about an account.

Anniversary: The date of the contact's wedding or anniversary.

AssistantNumber: Assistant’s phone number.

AssistantsName: Assistant’s name.

AttachProps: Information about an attachment that is specified as the **AttachProps** (section 2.2.4.1) complex type.

BCPicture: This helps a protocol client to display information about a contact. See PidLidBusinessCardCardPicture in [MS-OXOCNTC], section 2.2.1.7.2. Clients can pass this through as text instead of processing it.

BillingInformation: A string in any kind of format that describes billing information.

Birthday: Date of birth.

BizCard: This helps a protocol client to display information about a contact. See PidLidBusinessCardDisplayDefinition in [MS-OXOCNTC], section 2.2.1.7.1. Clients can pass this through as text instead of processing it.

Business2Number: Second work phone number.

CallbackNumber: Phone number used to return calls.

CarNumber: Car phone number.

CellPhone: Cellular phone number.

Certificate: This helps a protocol client to send secure e-mail. See PidTagUserX509Certificate in [MS-OXOCNTC], section 2.2.1.9.24. Clients can pass this through as text instead of processing it.

CertificateStr: The name of, or a string describing, the contents of either **Certificate** or **CertificatesX509**.

CertificatesX509: This helps a protocol client to send secure e-mail. See PidLidEmsAddressBookX509Certificate in [MS-OXOCNTC], section 2.2.1.9.21. Clients can pass this through as text instead of processing it.

ChildrensNames: A delimited string containing the contact's children's names.

Comments: Any text the user wants to enter about the item.

Company: Company or business name.

CompanyNumber: Company or business phone number.

CompanyPhonetic: **yomigana** for the company name.

ComputerNetworkName: Computer network name.

ConfDefServerIndex: The index of the default protocol server to use from the list contained in the **ConfServerNames** property. The first entry is index 0. -1 or an item without this property indicates no default protocol server has been set.

ConfServerNames: Delimited string of server names. These server names are usually for servers that handle meetings between groups of people over a network.

ContactLinkName: Delimited string of names of contacts that are relevant to this contact.

CustomerID: Any string that identifies the contact as a customer.

Editor: The last person to change this item.

Email: Primary e-mail address.

Email2: Second e-mail address.

Email3: Third e-mail address.

EmailDisplayAs1: The string that the user interface SHOULD display instead of the e-mail address in the **Email** field when users send mail to this contact.

EmailDisplayAs2: The string that the user interface SHOULD display instead of the e-mail address in the **Email2** field when users send mail to this contact.

EmailDisplayAs3: The string that the user interface SHOULD display instead of the e-mail address in the **Email3** field when users send mail to this contact.

Fax1: Primary fax number.

Fax2: Second fax number.

Fax3: Third fax number.

FaxDisplayAs1: The string that the user interface SHOULD display instead of the fax number in the **Fax1** field when a user sends a fax to this contact.

FaxDisplayAs2: The string that the user interface SHOULD display instead of the fax number in the **Fax2** field when a user sends a fax to this contact.

FaxDisplayAs3: The string that the user interface SHOULD display instead of the fax number in the **Fax3** field when a user sends a fax to this contact.

FileAs: The string that the user interface SHOULD display as the name of this contact.

FirstName: The first name of the contact.

FirstNamePhonetic: Yomigana for the first name.

FreeBusyURL: A URL to a location where clients can retrieve information about when the contact is busy. See **PidLidFreeBusyLocation** in [MS-OXOCNTC], section 2.2.1.9.10.

FTPSite: A URL to a File Transfer Protocol (FTP) **site**.

FullName: The full name of the contact.

Gender: The contact's gender. If this field is an integer type then clients SHOULD interpret it as a Gender. If this field is a **Choice** type then the **MAPPINGS** element SHOULD be present<35> and clients MUST use the **MAPPINGS** element to convert to and from the Gender type. See [MS-

LISTSWS] on fields that are of the **Choice** type. Each **MAPPING** element within the **MAPPINGS** element holds a number and a string.

```
<complexType name="CHOICES">
   <sequence>
      <element name="CHOICE" type="string" minOccurs="0" maxOccurs="unbounded" />
   </sequence>
</complexType>
```

CHOICES: Defines a list of valid values for the field that contains the CHOICE element.

CHOICES.CHOICE: A valid string for this property.

```
<complexType name="MAPPINGS">
   <sequence>
      <element name="MAPPING" type="string" minOccurs="0" maxOccurs="unbounded">
         <complexType>
            <simpleContent>
               <extension base="string">
                  <attribute name="Value" type="integer" use="required" />
               </extension>
            </simpleContent>
         </complexType>
      </element>
   </sequence>
</complexType>
```

MAPPINGS.MAPPING and MAPPINGS.Value: If a string in a **CHOICE** element exactly matches a string in a <MAPPING> element, then the **MAPPING.Value** attribute associated with it can be used to represent the string. This allows protocol clients to go between protocol servers running different languages and have the protocol servers tell protocol clients how to translate.

The following is an example of a field definition for the **Gender** field:

```
<Field ID="{a8eb573e-9e11-481a-a8c9-1104a54b2fbd}" Type="Choice" Name="Gender">
   <CHOICES>
      <CHOICE>Unknown</CHOICE>
      <CHOICE>Female</CHOICE>
      <CHOICE>Male</CHOICE>
   </CHOICES>
   <MAPPINGS>
      <MAPPING Value="0">Unknown</MAPPING>
      <MAPPING Value="1">Female</MAPPING>
      <MAPPING Value="2">Male</MAPPING>
   </MAPPINGS>
   <Default>Unknown</Default>
</Field>
```

In this example, the integer "2" is equivalent to the text string "Male". Protocol clients can store the **Gender** as a string, but SHOULD store it as an integer if the string matches a **MAPPING** value.

GovernmentIDNumber: Any string that describes the contact's identification number with the government.

Hobbies: A string describing the contact's hobbies.

Home2Number: Second home phone number.

HomeAddress: Home address.

HomeAddressCity: Home city name.

HomeAddressCountry: Home country or region name.

HomeAddressPostalCode: Post office box number in the home address.

HomeAddressStateOrProvince: The name of the state in the home address.

HomeFaxNumber: Home fax number.

HomeFreeForm: Any string that describes the contact's home address.

HomePhone: Home phone number.

HomeZip: The zip code or postal code in the home address.

IMAddress: Instant messaging address or name.

Initials: The initials of the contact's name.

ISDNNumber: A phone number used for Internet services.

JobTitle: The contact's job title.

Language: Any text that describes the contact's language.

LastNamePhonetic: Yomigana for the last name.

Location: Any string describing a location.

ManagersName: The name of the contact's manager.

MiddleName: The contact's middle name.

Mileage: Any user-entered text representing some mileage information associated with the contact for purposes of reimbursement.

NameTitle: A title associated with a name. This title comes before the name. Examples include "Mr.", "Mrs.", "Captain", "President", and so on. Use Suffix for titles that come after names.

Nickname: The contact's nickname.

OfficeLocation: Any string describing the location of the contact's office.

ol_Department: Any string describing the contact's work or business department.

OrganizationalIDNumber: Any string describing the contact's organizational identifier number.

OtherAddressCity: A city name for the contact's general purpose address.

OtherAddressCountry: A country or region name for the contact's general purpose address.

OtherAddressPostalCode: A postal code for the contact's general purpose address.

OtherAddressPObox: A post office box number for the contact's general purpose address.

OtherAddressStateOrProvince: A state or province name for the contact's general purpose address.

OtherAddressStreet: A street name for the contact's general purpose address.

OtherFaxNumber: General purpose fax number.

OtherFreeForm: Any text that describes the contact's general purpose address.

OtherNumber: Any phone number associated with the contact.

PagerNumber: A pager number.

PersonalWebsite: The contact's personal Web site.

Photo: A URL to the contact's picture. Pictures SHOULD be stored as some kind of image file. There is no guarantee that this URL points to safe data. The value of **Photo** SHOULD be a URL to one of the item's attachments, but can be any URL. Protocol clients can use **Photo** to find the contact's picture, but **AttachProps** SHOULD be used instead.

PrimaryNumber: The contact's primary phone number.

PostAddrID: Any string describing a postal address identifier.

Profession: Any string describing a profession.

RadioNumber: A phone number for a radio phone.

ReferredBy: Any string describing who or what referred this contact.

SpouseName: The name of the contact's spouse or domestic partner.

Suffix: A suffix or any other text. Some examples include "Jr.", "Sr.", "III", as in "Guy Example Jr."

TelexNumber: The phone number of a telex printer.

Title: Any string describing the contact's title.

TTYTDDNumber: The phone number of a telephone device for the deaf (TDD).

UserField1: This can be used for any text data about the contact.

UserField2: This can be used for any text data about the contact.

UserField3: This can be used for any text data about the contact.

UserField4: This can be used for any text data about the contact.

WebPage: A URL to a Web page for the contact.

WorkAddress: The street address where the contact works.

WorkCity: The name of the city where the contact works.

WorkCountry: The name of the country or region where the contact works.

WorkFax: The fax number at the contact's workplace.

WorkFreeForm: Any string describing the location of the contact's workplace.

WorkPhone: The phone number of the contact's work phone.

WorkPObox: The post office box number in the contact's work address.

WorkState: The name of the state where the contact works.

WorkZip: The zip code or postal code of the contact's workplace.

3.2.4.2.5 Discussion-Specific Schema

The discussion item schema implements the discussion item abstract data model (section 3.2.1.3). All discussion item properties can be empty or missing unless this section states otherwise. The following table identifies a field by a group of attributes that appear in **GetListResponse.GetListResult.List.Fields.Field** (see [MS-LISTSWS] and section 3.2.4.2.

Field.Name	Field.ID	Field.Type (interpret as)
Author	{1df5e554-ec7e-46a6-901d-d85a3881cb18}	User.
Body	{7662cd2c-f069-4dba-9e35-082cf976e170}	Note.
DiscussionTitle	{c5abfdc7-3435-4183-9207-3d1146895cf8}	Computed.
Editor	{d31655d1-1d5b-4511-95a1-7a09e9b75bf2}	User.
Importance	None defined.	None defined (Importance).
ThreadID	None defined.	None defined (stringGUID).
ThreadIndex	{cef73bf1-edf6-4dd9-9098-a07d83984700}	ThreadIndex.
Title	{fa564e0f-0c70-4ab9-b863-0177e6ddd247}	Text.

Author: The person who created the item.

Body: Text entered by a user.

DiscussionTitle: The original title or subject of the discussion item.

Editor: The last person who made changes to this item.

Importance: The level of **Importance** (section 2.2.5.7).

ThreadID: A **stringGUID**. This value can be ignored and SHOULD NOT be present.

ThreadIndex: A thread index string that uniquely identifies each discussion thread in a list. See [MS-LISTSWS] for the format and use of this field.

Title: The current title or subject of the discussion item.

3.2.4.2.6 Document-Specific Schema

The document schema specifies document items defined by the documents (section 3.2.1.4) abstract data model. All document properties can be empty or missing unless this section states otherwise. The following table identifies a field by a group of attributes that appear in **GetListResponse.GetListResult.List.Fields.Field** (see [MS-LISTSWS] and section 3.2.4.2).

Field.Name	Field.ID	Field.Type
Author	{1df5e554-ec7e-46a6-901d-d85a3881cb18}	User
Editor	{d31655d1-1d5b-4511-95a1-7a09e9b75bf2}	User
EncodedAbsUrl	{7177cfc7-f399-4d4d-905d-37dd51bc90bf}	Computed
FileDirRef	{56605df6-8fa1-47e4-a04c-5b384d59609f}	Lookup
FileSizeDisplay	{78a07ba4-bda8-4357-9e0f-580d64487583}	Computed
LinkCheckedOutTitle	{e2a15dfd-6ab8-4aec-91ab-02f6b64045b0}	Computed
LinkFilename	{5cc6dc79-3710-4374-b433-61cb4a686c12}	Computed

Author: The author of the document.

Editor: The last person who made changes to the document item.

EncodedAbsUrl: The URL to the document file on the protocol server. This MUST be present and valid.

FileDirRef: Protocol clients can parse the FileDirRef and RootFolder values to create a representation of the folders for the user. The URL text that is between '\' characters can be used as folder names. The value of the RootFolder attribute listed in section 3.1.4.5.1.1 SHOULD appear at the beginning of the FileDirRef value.

FileSizeDisplay: The size of the file in bytes. This MUST be present and valid. Protocol clients SHOULD add up the total size of all document items they intend to download and compare this to the **MaxBulkDocumentSyncSize** listed in section 3.1.4.7. Protocol clients SHOULD NOT download the document files if the total size is greater than **MaxBulkDocumentSyncSize**. Instead, protocol clients SHOULD only download the document items. Protocol clients can indicate to users whether each document item's file has or has not been downloaded.

LinkCheckedOutTitle: The name of the person who currently has this file **checked out**.

LinkFilename: The file name of the document. This MUST be present and valid.

3.2.4.2.7 Folder-Specific Schema

Folder items are items with the folder ContentTypeId listed in section 3.2.4.2.1. Folder items can appear in lists containing document and discussion items and use the same schema as the items they appear with. A folder item in a document list is covered in section 3.2.1.4. A folder item in list with discussion items is a root item as described in section 3.2.1.3.

3.2.4.2.8 Task-Specific Schema

The task schema specifies task items defined in the tasks (section 3.2.1.5) abstract data model. All task properties can be empty or missing unless this section states otherwise. The following table identifies a field by a group of attributes that appear in **GetListResponse.GetListResult.List.Fields.Field** (see [MS-LISTSWS] and section 3.2.4.2.

Field.Name	Field.ID	Field.Type (interpret as)
ActualWork	{B0B3407E-1C33-40ed-A37C-2430B7A5D081}	Number.

Field.Name	Field.ID	Field.Type (interpret as)
AssignedTo	{53101f38-dd2e-458c-b245-0c236cc13d1a}	User.
Body	{7662cd2c-f069-4dba-9e35-082cf976e170}	Note.
BillingInformation	{4F03F66B-FB1E-4ed2-AB8E-F6ED3FE14844}	Text.
ContactLinkName	None defined.	None defined (Text).
DateComplete	{24BFA3C2-E6A0-4651-80E9-3DB44BF52147}	DateTime.
DueDate	{cd21b4c2-6841-4f9e-a23a-738a65f99889}	DateTime.
Editor	{d31655d1-1d5b-4511-95a1-7a09e9b75bf2}	User.
EmailBody	None defined.	None defined (Text).
FormURN	None defined.	None defined (Text).
LastUpdate	None defined.	None defined (DateTime).
Mileage	{3126C2F1-063E-4892-828F-0696EC6E105F}	Text.
PercentComplete	{d2311440-1ed6-46ea-b46d-daa643dc3886}	Number.
Priority	{a8eb573e-9e11-481a-a8c9-1104a54b2fbd}	Choice.
Role	{EEAEAAF1-4110-465b-905E-DF1073A7E0E6}	Text.
StartDate	{64cd368d-2f95-4bfc-a1f9-8d4324ecb007}	DateTime.
Status	{c15b34c3-ce7d-490a-b133-3f4de8801b76}	Choice.
TaskCompanies	{3914F98E-6D99-4218-9BA3-AF7370B9E7BC}	Text.
Title	{fa564e0f-0c70-4ab9-b863-0177e6ddd247}	Text.
TotalWork	{F3C4A259-19A2-44b8-AB3D-E9145D07D538}	Number.

ActualWork: Actual work time spent on the task expressed in number of minutes.

AssignedTo: The person to whom the task is currently assigned.

Body: Any user-entered text about the task. This SHOULD be the primary place to enter text.

BillingInformation: Billing information for the task.

ContactLinkName: Names of people, businesses, or other names that serve as contacts. This is typically a semicolon delimited string, but can be any text.

DateComplete: The date that the task was finished. If the task is not complete, then protocol clients SHOULD ignore this.

DueDate: The date that the task is due.

Editor: The last person who changed this item.

EmailBody: This text overrides the text in the Body property when the item's ContentTypeId indicates a workflow task. See section 3.2.4.2.1.

FormURN: This property SHOULD be ignored by protocol clients<36> and protocol servers and SHOULD NOT be present. Protocol clients can set this to any text string.

LastUpdate: The date and time when the task was last updated.

Mileage: Any user entered text that contains mileage information associated with the task for purposes of reimbursement.

PercentComplete: A percentage describing how close the task is to being done. 0% means nothing is done. 100% means everything is done.

Priority: A choice between several strings that describe how important the task is. These strings are provided by the protocol server as part of the schema. See [MS-LISTSWS] on fields that are of the **Choice** type. In addition to the <CHOICE> element, the protocol server SHOULD provide a <MAPPINGS> element. Each <MAPPINGS> element holds a number and a string.

```
<complexType name="CHOICES">
   <sequence>
      <element name="CHOICE" type="string" minOccurs="0" maxOccurs="unbounded" />
   </sequence>
</complexType>
```

CHOICES: Defines a list of valid values for the field that contains the CHOICE element.

CHOICES.CHOICE: A valid string for this property.

```
<complexType name="MAPPINGS">
   <sequence>
      <element name="MAPPING" type="string" minOccurs="0" maxOccurs="unbounded">
         <complexType>
            <simpleContent>
               <extension base="string">
                  <attribute name="Value" type="integer" use="required" />
               </extension>
            </simpleContent>
         </complexType>
      </element>
   </sequence>
<complexType>
```

MAPPINGS.MAPPING and MAPPINGS.Value: If a string in a **CHOICE** element exactly matches a string in a <MAPPING> element, then the **MAPPING.Value** attribute associated with it can be used to represent the string. This allows protocol clients to go between protocol servers running different languages and have the protocol servers tell protocol clients how to translate.

The following is an example of a field definition for the **Priority** field:

```
<Field ID="{a8eb573e-9e11-481a-a8c9-1104a54b2fbd}" Type="Choice" Name="Priority">
   <CHOICES>
      <CHOICE>High</CHOICE>
      <CHOICE>Normal</CHOICE>
      <CHOICE>Low</CHOICE>
      <CHOICE>Priority X</CHOICE>
   </CHOICES>
   <MAPPINGS>
```

```
      <MAPPING Value="3">Low</MAPPING>
      <MAPPING Value="1">High</MAPPING>
      <MAPPING Value="2">Normal</MAPPING>
   </MAPPINGS>
   <Default>(2) Normal</Default>
</Field>
```

In this example, the integer 2 is equivalent to the text string "Normal". Protocol clients can store the **Priority** as a string, but SHOULD store it as an integer if the string matches a MAPPING value. When uploading items, protocol clients MUST change integer priorities to strings. Protocol clients MUST store the **Priority** value as text if the CHOICE value has no MAPPING, as "Priority X" does in this example.

Role: Any user entered text about the role the task plays.

StartDate: The date that work on the task is supposed to begin.

Status: This is the status of the task. This property works exactly the same way as the **Priority** task property also specified in this section (see **Priority** in section 3.2.4.2.8. Here is an example of the **Status** field definition:

```
<Field Type="Choice" ID="{c15b34c3-ce7d-490a-b133-3f4de8801b76}" Name="Status">
   <CHOICES>
      <CHOICE>Not Started</CHOICE>
      <CHOICE>In Progress</CHOICE>
      <CHOICE>Completed</CHOICE>
      <CHOICE>Status X</CHOICE>
   </CHOICES>
   <MAPPINGS>
      <MAPPING Value="3">Completed</MAPPING>
      <MAPPING Value="1">Not Started</MAPPING>
      <MAPPING Value="2">In Progress</MAPPING>
   </MAPPINGS>
</Field>
```

In this example, the integer 1 is equivalent to the text string "Not Started". Protocol clients can store the **Status** as a string, but SHOULD store it as an integer if the string matches a MAPPING value. When uploading items, protocol clients MUST change integer status to strings. Protocol clients MUST store the **Status** value as text if the CHOICE value has no MAPPING, as "Status X" does in this example.

TaskCompanies: Any text describing companies that the task is for, companies doing the task, or any other companies, businesses, or people.

Title: The title of the task.

TotalWork: The total amount of work time spent on the task so far, in minutes.

3.2.5 Timer Events

TimeElapsedSinceLastSync: Protocol clients MUST NOT initiate the Lists Client Sync Protocol if this timer is less than the value of **MinTimeBetweenSyncs**, which is listed in section 3.1.4.7.1.1. Protocol clients SHOULD NOT initiate the Lists Client Sync Protocol if this timer is less than **RecommendedTimeBetweenSyncs**, which is listed in section 3.1.4.7.1.1. This timer is reset to 0 when the Lists Client Sync Protocol completes.

3.2.6 Other Local Events

Users can initiate the Lists Client Sync Protocol or cancel it at any time, except where timers prevent this. If the protocol is cancelled, then all remote operations and all processing of results SHOULD be aborted at the protocol client's earliest opportunity.

3.2.6.1 Lost, Interrupted, or Failed Connections

Protocol clients SHOULD either restart this protocol using an alternate URL (see section 3.2.4.1) or abort the protocol.

3.2.6.2 Server or List Restoration

If a protocol server restores data from backups then protocol clients might not be able to determine the correct state of the items. Protocol clients SHOULD discard their local data because the protocol server might have been restored to undo changes that protocol clients still have. Protocol clients can keep a backup of changes that were not uploaded to the protocol server. A protocol server or list restoration event is indicated by `Changes.Id.ChangeType="restore"` and no data in the **Changes.Id** listed in section 3.1.4.7.1.1.

3.2.6.3 Permission Changes

Protocol clients know permissions have changed by comparing **GetListItemChangesSinceTokenResponse.GetListItemChangesSinceTokenResult.listitems.EffectivePermMask** (see [MS-LISTSWS]) to the last value they received from that attribute. If the values differ then permissions changed. Clients SHOULD discard their change token and restart this protocol when permissions change. Any items not received upon completion of this protocol SHOULD be deleted from the protocol client. Protocol clients SHOULD do this because protocol servers will not give protocol clients information about items they do not have access to. If the protocol client does not do this, a permission change can result in items remaining on the protocol client after the protocol client loses access to them.

3.2.6.4 Corrupt or Invalid Data

If a protocol client uploads an item and the protocol server rejects the item data given in **UpdateListItems** (section 3.1.4.10), then the protocol client SHOULD NOT attempt to upload that item again. Protocol clients SHOULD keep a copy of the failed item. Protocol clients SHOULD discard the change token and restart this protocol to get the correct current state of the list. Protocol clients can use a WSDL operation from [MS-LISTSWS] to download only the failed item instead.

3.2.6.5 Restoring Items

Protocol clients can follow the server list restoration (section 3.2.6.2) logic when Changes.Id.ChangeType="restore" and an integer is given in Changes.Id (see section 3.1.4.7.1.1. This indicates an item was restored from backups. Protocol clients can use a WSDL operation from [MS-LISTSWS] to download only the restored item.

4 Protocol Examples

In each example, protocol clients follow the state diagram (section 3.2.4.1) given earlier in this document.

4.1 Client Downloading a Group of Items from a Server

In this example, a protocol client provides an offline copy of server data for users to view. The protocol client does not permit users to edit the data. The list type for this example is documents.

The protocol client would begin with an initialization (section 3.2.3) of data required to use [MS-LISTSWS], such as the list identifier, and other data.

Once initialized, the protocol client would follow the state diagram (section 3.2.4.1) by calling **GetList**. In this example, the protocol client has not used the Lists Client Sync Protocol with the list before, so **TimeElapsedSinceLastSync** is ignored.

The call to **GetList** succeeds, so the protocol client processes **GetListResponse.** The protocol client requires all the fields specified in the document-specific schema (section 3.2.4.2.6) and lists each of those field names in **GetListItemChangesSinceToken.viewFields.ViewFields**. The following is the result for **ViewFields**:

```
<viewFields>
   <ViewFields>
      <FieldRef Name="ID" />
      <FieldRef Name="ReplicationID" />
      <FieldRef Name="Attachments" />
      <FieldRef Name="owshiddenversion" />
      <FieldRef Name="Created" />
      <FieldRef Name="Modified" />
      <FieldRef Name="vti_versionhistory" />
      <FieldRef Name="ContentTypeId" />
      <FieldRef Name="FileDirRef" />
      <FieldRef Name="EncodedAbsUrl" />
      <FieldRef Name="Author" />
      <FieldRef Name="LinkFilename" />
      <FieldRef Name="Editor" />
      <FieldRef Name="LinkCheckedOutTitle" />
      <FieldRef Name="FileSizeDisplay" />
      <FieldRef Name="Categories" />
      <FieldRef Name="MetaInfo" />
      <FieldRef Name="PermMask" />
   </ViewFields>
</viewFields>
```

The protocol client also requests to receive items in batches of 100, so it sets **GetListItemChangesSinceToken.rowLimit** to 100:

```
<rowLimit>100</rowLimit>
```

For **GetListItemChangesSinceToken.queryOptions**, the protocol client looks over the list in section 3.1.4.7 and does the following:

```
<queryOptions>
```

```
    <QueryOptions>
       <ViewAttributes Scope="RecursiveAll" />
        <DateInUtc>TRUE</DateInUtc>
        <IncludePermissions>FALSE</IncludePermissions>
        <IncludeAttachmentUrls>TRUE</IncludeAttachmentUrls>
        <IncludeAttachmentVersion>TRUE</IncludeAttachmentVersion>
        <ExpandUserField>TRUE</ExpandUserField>
        <MeetingInstanceID>-1</MeetingInstanceID>
        <OptimizeLookups>TRUE</OptimizeLookups>
     </QueryOptions>
  </queryOptions>
```

The protocol client is now set to send a **GetListItemChangesSinceToken** operation call to download items.

While processing each item, the protocol client sees each item has an attachment, which is expected because document items are supposed to have exactly one attachment. The protocol client requests these attachments and uses **GetAttachmentCollection** to get the attachment list for each item. The protocol client knows that the attachments is downloaded last just in case the total size of all the documents is larger than **MaxBulkDocumentSyncSize** (see section 3.1.4.7.1.1). Instead of downloading the files now, the protocol client remembers the attachment file URLs for later.

There are 150 documents in the list, so the protocol client makes a second **GetListItemChangesSinceToken** call with the same **viewFields**, **queryOptions**, and **rowLimit**, but uses the change tokens as specified in [MS-LISTSWS] to get the next batch of results.

Once all documents are processed and their sizes added up, the protocol client sees that the total size is less than **MaxBulkDocumentSyncSize**. The protocol client then revisits each document item and uses the URL to the attachment file to make HTTP GET calls for each document item's attachment.

The protocol client requests to be read only and avoid uploading anything, so the "upload changes" state does nothing.

The network traffic for this example is as follows.

Operations	Sender and Purpose
GetList	Protocol client (gets schema)
GetListResponse	Protocol server (response)
GetListItemChangesSinceToken	Protocol client (download items)
GetListItemChangesSinceTokenResponse	Protocol server (response)
GetAttachmentCollection (100 times)	Protocol client (get attach URLs)
GetAttachmentCollectionResponse (100 times)	Protocol server (response)
GetListItemChangesSinceToken	Protocol client (second batch of items)
GetListItemChangesSinceTokenResponse	Protocol server (response)
GetAttachmentCollection (50 times)	Protocol client (get attach URLs)
GetAttachmentCollectionResponse (50 times)	Protocol server (response)

Operations	Sender and Purpose
HTTP GET (150 times)	Protocol client (download file)
HTTP OK (150 times)	Protocol server (response)

4.2 Uploading a New Recurring Appointment with Exceptions

In this example, the protocol client has already downloaded appointment items. The protocol client now has a new recurring appointment (section 3.2.1.1.2) with exceptions to a recurrence (section 3.2.1.1.3) to upload. First the protocol client follows the first figure in section 3.2.4.1 State Diagram up to the "Upload changes" state. There are no changes and no new items, so the network traffic is as follows.

Operations	Sender and purpose
GetListItemChangesSinceToken	Protocol client (check for new or changed items).
GetListItemChangesSinceTokenResponse	Protocol server (response).

Note that the protocol client did not use the **GetList** operation. This is because the protocol client remembered the results from an earlier **GetList** and used them instead.

The protocol client is up to date with recent changes, so the next step is to upload. The protocol client uses **UpdateListItems** operation as specified in [MS-LISTSWS] to create a new recurring appointment (section 3.2.1.1.2) with the appointment-specific schema (section 3.2.4.2.2). Because the protocol server assigns identifiers and the protocol client needs to set the MasterSeriesItemID field, the protocol client is not able to create valid exception items until it receives **UpdateListItemsResponse**. While processing **UpdateListItemsResponse**, the protocol client uses a second **UpdateListItems** operation to handle the exceptions.

The network traffic is as follows:

Operations	Sender and purpose
UpdateListItems	Protocol client (create new recurring appointment).
UpdateListItemsResponse	Protocol server (response).
UpdateListItems	Protocol client (create new exception).
UpdateListItemsResponse	Protocol server (response).

4.3 Updating an Item with an Attachment

In this example, the protocol client has already downloaded contacts. The protocol client has also modified an existing contact that and modified that contact's attachment.

First, the protocol client downloads new and changed items, as described in the preceding examples.

Next the protocol client makes an **UpdateListItems** operation call (see [MS-LISTSWS]) to update the contact's properties on the server. The server response indicates a successful update, so the contact makes an **AddAttachment** (section 3.1.4.1) operation call to set the attachment. The server responds to **AddAttachment** (section 3.1.4.1) with a SOAP exception and error code that

indicates the file name is in use. Because of this, the protocol client tries HTTP PUT to update the attachment, which succeeds. The network traffic is as follows.

Operations	Sender and purpose
GetListItemChangesSinceToken	Protocol client (check for new/changed items).
GetListItemChangesSinceTokenResponse	Protocol server (response).
UpdateListItems	Protocol client (update contact).
UpdateListItemsResponse	Protocol server (response).
AddAttachment	Protocol client (upload attachment).
SOAP fault	Protocol server (file name in use).
HTTP PUT	Protocol client (overwrite attachment).
HTTP OK	Protocol server (response).

4.4 Uploading a New Recurrence with an Attachment and Exceptions that Have Attachments

This example expands on the previous recurring item upload by giving all items one attachment. It is an upload of a new recurring appointment with two exceptions (3 items total). Because the item attachment cannot be created on the protocol server until the item is created on the protocol server, all attachment uploads are finished after the protocol client receives an update confirmation via **UpdateListItemsResponse**. The network traffic from the example, uploading a new recurring appointment with exceptions (section 4.2), is as follows.

Operations	Sender and purpose
UpdateListItems	Protocol client (create new recurring appointment).
UpdateListItemsResponse	Protocol server (response).
UpdateListItems	Protocol client (create new exception).
UpdateListItemsResponse	Protocol server (response).

After the first **UpdateListItemsResponse**, the protocol client has confirmed that the protocol server has the recurring item. Attachment upload, as described in section 4.3, happens next. The exception items do not exist on the protocol server yet, so this attachment upload is for the recurrence only. The network traffic is as follows.

Operations	Sent by (why sent)
UpdateListItems	Protocol client (create new recurring appointment).
UpdateListItemsResponse	Protocol server (response).
AddAttachment	Protocol client (set recurring item attachment).
AddAttachmentResponse	Protocol server (response).
UpdateListItems	Protocol client (create all new exceptions).

Operations	Sent by (why sent)
UpdateListItemsResponse	Protocol server (response).
AddAttachment	Protocol client (set exception item attachment).
AddAttachmentResponse	Protocol server (response).
AddAttachment	Protocol client (set exception item attachment).
AddAttachmentResponse	Protocol server (response).

Each of the three **AddAttachment** operation calls listed are for different attachments and different items. Because the items are new, so are the attachments. That is why the protocol server did not respond with a file name in use error, unlike in section 4.3.

4.5 Uploading New Discussion Items

Discussion items use a different WSDL operation call to upload. Other item types use **UpdateListItems**, but discussion items use **AddDiscussionBoardItem** for new items.

This example starts after the protocol client has downloaded new and updated discussion items. The protocol client is now beginning the "Upload changes" state depicted in the second figure in section 3.2.4.1. The protocol client will now upload two discussion items.

The protocol client makes one **AddDiscussionBoardItem** operation call for each new discussion item. Both calls happen before **UpdateListItems**. Next, the protocol client requests to set the Categories property on the discussion items that were just uploaded. **AddDiscussionBoardItem** did not let the protocol client do this, so an **UpdateListItems** is required. The network traffic is as follows.

Operations	Sender and purpose
AddDiscussionBoardItem	Protocol client (create new discussion item).
AddDiscussionBoardItemResponse	Protocol server (response).
AddDiscussionBoardItem	Protocol client (create new discussion item).
AddDiscussionBoardItemResponse	Protocol server (response).
UpdateListItems	Protocol client (set Categories on both items).
UpdateListItemsResponse	Protocol server (response).

Note that the **UpdateListItems** operation call does not create the discussion items; it only updates existing ones.

5 Security

5.1 Security Considerations for Implementers

Security considerations for this protocol are covered in the preceding sections.

5.2 Index of Security Parameters

See [MS-LISTSWS] section 5.2.

6 Appendix A: Full WSDL

For ease of implementation, the full WSDL is provided:

```
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://schemas.microsoft.com/sharepoint/soap/"
xmlns:s1="http://microsoft.com/wsdl/types/" xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
targetNamespace="http://schemas.microsoft.com/sharepoint/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://schemas.microsoft.com/sharepoint/soap/">
      <s:import namespace="http://www.w3.org/2001/XMLSchema" />
      <s:import namespace="http://microsoft.com/wsdl/types/" />
      <s:element name="GetList">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="listName" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetListResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="GetListResult">
              <s:complexType mixed="true">
                <s:sequence>
                  <s:any />
                </s:sequence>
              </s:complexType>
            </s:element>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetListItemChanges">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="listName" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="viewFields">
              <s:complexType mixed="true">
                <s:sequence>
                  <s:any />
                </s:sequence>
              </s:complexType>
            </s:element>
            <s:element minOccurs="0" maxOccurs="1" name="since" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="contains">
              <s:complexType mixed="true">
                <s:sequence>
                  <s:any />
                </s:sequence>
              </s:complexType>
            </s:element>
          </s:sequence>
```

```
  </s:complexType>
</s:element>
<s:element name="GetListItemChangesResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="GetListItemChangesResult">
        <s:complexType mixed="true">
          <s:sequence>
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="GetListItemChangesSinceToken">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="listName" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="viewName" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="query">
        <s:complexType mixed="true">
          <s:sequence>
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element minOccurs="0" maxOccurs="1" name="viewFields">
        <s:complexType mixed="true">
          <s:sequence>
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element minOccurs="0" maxOccurs="1" name="rowLimit" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="queryOptions">
        <s:complexType mixed="true">
          <s:sequence>
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element minOccurs="0" maxOccurs="1" name="changeToken" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="contains">
        <s:complexType mixed="true">
          <s:sequence>
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="GetListItemChangesSinceTokenResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="GetListItemChangesSinceTokenResult">
        <s:complexType mixed="true">
          <s:sequence>
```

```
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="UpdateListItems">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="listName" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="updates">
        <s:complexType mixed="true">
          <s:sequence>
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="UpdateListItemsResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="UpdateListItemsResult">
        <s:complexType mixed="true">
          <s:sequence>
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="AddDiscussionBoardItem">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="listName" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="message" type="s:base64Binary" />
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="AddDiscussionBoardItemResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="AddDiscussionBoardItemResult">
        <s:complexType mixed="true">
          <s:sequence>
            <s:any />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="AddAttachment">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="listName" type="s:string" />
```

```
            <s:element minOccurs="0" maxOccurs="1" name="listItemID" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="fileName" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="attachment" type="s:base64Binary" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="AddAttachmentResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="AddAttachmentResult" type="s:string"
/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetAttachmentCollection">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="listName" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="listItemID" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetAttachmentCollectionResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="GetAttachmentCollectionResult">
              <s:complexType mixed="true">
                <s:sequence>
                  <s:any />
                </s:sequence>
              </s:complexType>
            </s:element>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="DeleteAttachment">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="listName" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="listItemID" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="url" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="DeleteAttachmentResponse">
        <s:complexType />
      </s:element>
    </s:schema>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://microsoft.com/wsdl/types/">
      <s:simpleType name="guid">
        <s:restriction base="s:string">
          <s:pattern value="[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-
9a-fA-F]{12}" />
        </s:restriction>
      </s:simpleType>
      <s:simpleType name="BusyStatus">
        <s:restriction base="s:int">
          <s:enumeration value="0"/>
```

```
      <s:enumeration value="1"/>
      <s:enumeration value="2"/>
      <s:enumeration value="3" />
    </s:restriction>
  </s:simpleType>
  <s:simpleType name="booleanInteger">
    <s:restriction base="s:int">
      <s:enumeration value="0" />
      <s:enumeration value="-1" />
    </s:restriction>
  </s:simpleType>
  <s:simpleType name="DayOfWeek">
    <s:restriction base="s:string">
      <s:enumeration value="su" />
      <s:enumeration value="mo" />
      <s:enumeration value="tu" />
      <s:enumeration value="we" />
      <s:enumeration value="th" />
      <s:enumeration value="fr" />
      <s:enumeration value="sa" />
    </s:restriction>
  </s:simpleType>
  <s:simpleType name="EventType">
    <s:restriction base="s:int" >
      <s:enumeration value="0" />
      <s:enumeration value="1" />
      <s:enumeration value="2" />
      <s:enumeration value="3" />
      <s:enumeration value="4" />
    </s:restriction>
  </s:simpleType>
  <s:simpleType name="FollowUp">
    <s:restriction base="s:int" >
      <s:enumeration value="0" />
      <s:enumeration value="1" />
      <s:enumeration value="2" />
      <s:enumeration value="3" />
      <s:enumeration value="4" />
      <s:enumeration value="5" />
    </s:restriction>
  </s:simpleType>
  <s:simpleType name="Importance">
    <s:restriction base="s:int" >
      <s:enumeration value="0" />
      <s:enumeration value="1" />
      <s:enumeration value="2" />
    </s:restriction>
  </s:simpleType>
  <s:simpleType name="Participants">
    <s:restriction base="s:string" >
      <s:maxLength value="255" />
      <s:pattern value="(;#[0-9]+)*" />
    </s:restriction>
  </s:simpleType>
  <s:simpleType name="Priority">
    <s:restriction base="s:int" >
      <s:enumeration value="-1" />
      <s:enumeration value="0" />
      <s:enumeration value="1" />
```

```
        </s:restriction>
      </s:simpleType>
      <s:simpleType name="stringGUID">
        <s:restriction base="s:string">
          <s:maxLength value="38"/>
          <s:minLength value="38"/>
          <s:pattern value="\{[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-
9a-fA-F]{12}\}" />
        </s:restriction>
      </s:simpleType>
      <s:simpleType name="TrueFalseDOW">
        <s:restriction base="s:string">
          <s:enumeration value="TRUE" />
          <s:enumeration value="FALSE" />
          <s:enumeration value="true" />
          <s:enumeration value="false" />
        </s:restriction>
      </s:simpleType>
      <s:simpleType name="WeekdayOfMonth">
        <s:restriction base="s:string">
          <s:enumeration value="first" />
          <s:enumeration value="second" />
          <s:enumeration value="third" />
          <s:enumeration value="fourth" />
          <s:enumeration value="last" />
        </s:restriction>
      </s:simpleType>
      <s:simpleType name="DayOfWeekOrMonth">
        <s:restriction base="s:string">
          <s:enumeration value="su" />
          <s:enumeration value="mo" />
          <s:enumeration value="tu" />
          <s:enumeration value="we" />
          <s:enumeration value="th" />
          <s:enumeration value="fr" />
          <s:enumeration value="sa" />
          <s:enumeration value="day" />
          <s:enumeration value="weekday" />
          <s:enumeration value="weekend_day" />
        </s:restriction>
      </s:simpleType>
      <s:complexType name="AttachProps">
        <s:sequence>
          <s:element name="File">
            <s:complexType>
              <s:simpleContent>
                <s:extension base="s:string">
                  <s:attribute name="Photo" use="required">
                    <s:simpleType>
                      <s:restriction base="s:string">
                        <s:enumeration value="0" />
                        <s:enumeration value="-1" />
                      </s:restriction>
                    </s:simpleType>
                  </s:attribute>
                </s:extension>
              </s:simpleContent>
            </s:complexType>
          </s:element>
```

```
        </s:sequence>
      </s:complexType>
        <s:complexType name="RecurrenceRule">
        <s:sequence>
          <s:element name="firstDayOfWeek" type="s1:DayOfWeekOrMonth" />
          <s:element name="repeat" type="s1:RepeatPattern" />
          <s:choice>
            <s:element name="windowEnd" type="s:dateTime" />
            <s:element name="repeatForever">
              <s:simpleType>
                <s:restriction base="s:string">
                  <s:enumeration value="FALSE" />
                </s:restriction>
              </s:simpleType>
            </s:element>
            <s:element name="repeatInstances" type="s:integer" />
          </s:choice>
        </s:sequence>
      </s:complexType>
      <s:complexType name="RecurrenceDefinition">
        <s:sequence>
          <s:element name="rule" type="s1:RecurrenceRule" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="RecurrenceXML">
        <s:sequence>
          <s:element name="recurrence" type="s1:RecurrenceDefinition" />
          <s:element name="deleteExceptions" type="s:string" fixed="true" minOccurs="0"
maxOccurs="1" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="RepeatPattern">
        <s:choice>
          <s:element name="daily">
            <s:complexType>
              <s:simpleContent>
                <s:extension base="s:string">
                  <s:attribute name="su" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="mo" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="tu" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="we" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="th" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="fr" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="sa" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="weekFrequency" type="s:integer" default="1"
use="optional" />
                </s:extension>
              </s:simpleContent>
            </s:complexType>
          </s:element>
          <s:element name="monthlyByDay">
            <s:complexType>
              <s:simpleContent>
```

```
                <s:extension base="s:string">
                  <s:attribute name="su" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="mo" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="tu" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="we" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="th" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="fr" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="sa" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="day" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="weekday" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="weekend_day" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="monthFrequency" type="s:integer" default="1"
use="optional" />
                  <s:attribute name="weekdayOfMonth" type="s1:WeekdayOfMonth" default="first"
use="optional" />
                </s:extension>
              </s:simpleContent>
            </s:complexType>
          </s:element>
          <s:element name="monthly">
            <s:complexType>
              <s:simpleContent>
                <s:extension base="s:string">
                  <s:attribute name="monthFrequency" type="s:integer" default="1"
use="optional" />
                  <s:attribute name="day" type="s:integer" default="1" use="optional" />
                </s:extension>
              </s:simpleContent>
            </s:complexType>
          </s:element>
          <s:element name="yearly">
            <s:complexType>
              <s:simpleContent>
                <s:extension base="s:string">
                  <s:attribute name="yearFrequency" type="s:integer" default="1"
use="optional" />
                  <s:attribute name="month" type="s:integer" default="1" use="optional" />
                  <s:attribute name="day" type="s:integer" default="1" use="optional" />
                </s:extension>
              </s:simpleContent>
            </s:complexType>
          </s:element>
          <s:element name="yearlyByDay">
            <s:complexType>
              <s:simpleContent>
                <s:extension base="s:string">
                  <s:attribute name="su" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="mo" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
```

```
                  <s:attribute name="tu" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="we" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="th" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="fr" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="sa" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="day" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="weekday" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="weekend_day" type="s1:TrueFalseDOW" default="FALSE"
use="optional" />
                  <s:attribute name="yearFrequency" type="s:integer" default="1"
use="optional" />
                  <s:attribute name="month" type="s:integer" default="1" use="optional" />
                  <s:attribute name="weekdayOfMonth" type="s1:WeekdayOfMonth" default="first"
use="optional" />
                </s:extension>
              </s:simpleContent>
            </s:complexType>
          </s:element>
        </s:choice>
      </s:complexType>
      <s:complexType name="TimeZoneRule">
        <s:sequence>
          <s:element name="standardBias" type="s:integer" />
          <s:element name="additionalDaylightBias" type="s:integer" minOccurs="0" />
          <s:element name="standardDate" type="s1:TransitionDate" minOccurs="0" />
          <s:element name="daylightDate" type="s1:TransitionDate" minOccurs="0" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="TimeZoneXML">
        <s:sequence>
          <s:element name="timeZoneRule" type="s1:TimeZoneRule" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="TransitionDate">
        <s:sequence>
          <s:element name="transitionRule">
            <s:complexType>
              <s:simpleContent>
                <s:extension base="s:string">
                  <s:attribute name="day" type="s1:DayOfWeek" default="su" use="optional" />
                  <s:attribute name="month" type="s:integer" use="required" />
                  <s:attribute name="dayOfMonth" type="s:integer" use="optional" />
                  <s:attribute name="weekdayOfMonth" type="s1:WeekdayOfMonth" default="first"
use="optional" />
                </s:extension>
              </s:simpleContent>
            </s:complexType>
          </s:element>
          <s:element name="transitionTime" type="s:time" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="CHOICES">
         <s:sequence>
```

```
            <s:element name="CHOICE" type="string" minOccurs="0" maxOccurs="unbounded" />
         </s:sequence>
      </s:complexType>
      <s:complexType name="MAPPINGS">
         <s:sequence>
            <s:element name="MAPPING" type="string" minOccurs="0" maxOccurs="unbounded">
               <s:complexType>
                  <s:simpleContent>
                     <s:extension base="string">
                        <s:attribute name="Value" type="integer" use="required" />
                     </s:extension>
                  </s:simpleContent>
               </s:complexType>
            </s:element>
         </s:sequence>
      </s:complexType>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="GetListSoapIn">
    <wsdl:part name="parameters" element="tns:GetList" />
  </wsdl:message>
  <wsdl:message name="GetListSoapOut">
    <wsdl:part name="parameters" element="tns:GetListResponse" />
  </wsdl:message>
  <wsdl:message name="GetListItemChangesSoapIn">
    <wsdl:part name="parameters" element="tns:GetListItemChanges" />
  </wsdl:message>
  <wsdl:message name="GetListItemChangesSoapOut">
    <wsdl:part name="parameters" element="tns:GetListItemChangesResponse" />
  </wsdl:message>
  <wsdl:message name="GetListItemChangesSinceTokenSoapIn">
    <wsdl:part name="parameters" element="tns:GetListItemChangesSinceToken" />
  </wsdl:message>
  <wsdl:message name="GetListItemChangesSinceTokenSoapOut">
    <wsdl:part name="parameters" element="tns:GetListItemChangesSinceTokenResponse" />
  </wsdl:message>
  <wsdl:message name="UpdateListItemsSoapIn">
    <wsdl:part name="parameters" element="tns:UpdateListItems" />
  </wsdl:message>
  <wsdl:message name="UpdateListItemsSoapOut">
    <wsdl:part name="parameters" element="tns:UpdateListItemsResponse" />
  </wsdl:message>
  <wsdl:message name="AddDiscussionBoardItemSoapIn">
    <wsdl:part name="parameters" element="tns:AddDiscussionBoardItem" />
  </wsdl:message>
  <wsdl:message name="AddDiscussionBoardItemSoapOut">
    <wsdl:part name="parameters" element="tns:AddDiscussionBoardItemResponse" />
  </wsdl:message>
  <wsdl:message name="AddAttachmentSoapIn">
    <wsdl:part name="parameters" element="tns:AddAttachment" />
  </wsdl:message>
  <wsdl:message name="AddAttachmentSoapOut">
    <wsdl:part name="parameters" element="tns:AddAttachmentResponse" />
  </wsdl:message>
  <wsdl:message name="GetAttachmentCollectionSoapIn">
    <wsdl:part name="parameters" element="tns:GetAttachmentCollection" />
  </wsdl:message>
  <wsdl:message name="GetAttachmentCollectionSoapOut">
    <wsdl:part name="parameters" element="tns:GetAttachmentCollectionResponse" />
```

```
  </wsdl:message>
  <wsdl:message name="DeleteAttachmentSoapIn">
    <wsdl:part name="parameters" element="tns:DeleteAttachment" />
  </wsdl:message>
  <wsdl:message name="DeleteAttachmentSoapOut">
    <wsdl:part name="parameters" element="tns:DeleteAttachmentResponse" />
  </wsdl:message>
  <wsdl:portType name="ListsSoap">
    <wsdl:operation name="GetList">
      <wsdl:input message="tns:GetListSoapIn" />
      <wsdl:output message="tns:GetListSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="GetListItemChanges">
      <wsdl:input message="tns:GetListItemChangesSoapIn" />
      <wsdl:output message="tns:GetListItemChangesSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="GetListItemChangesSinceToken">
      <wsdl:input message="tns:GetListItemChangesSinceTokenSoapIn" />
      <wsdl:output message="tns:GetListItemChangesSinceTokenSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="UpdateListItems">
      <wsdl:input message="tns:UpdateListItemsSoapIn" />
      <wsdl:output message="tns:UpdateListItemsSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="AddDiscussionBoardItem">
      <wsdl:input message="tns:AddDiscussionBoardItemSoapIn" />
      <wsdl:output message="tns:AddDiscussionBoardItemSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="AddAttachment">
      <wsdl:input message="tns:AddAttachmentSoapIn" />
      <wsdl:output message="tns:AddAttachmentSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="GetAttachmentCollection">
      <wsdl:input message="tns:GetAttachmentCollectionSoapIn" />
      <wsdl:output message="tns:GetAttachmentCollectionSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="DeleteAttachment">
      <wsdl:input message="tns:DeleteAttachmentSoapIn" />
      <wsdl:output message="tns:DeleteAttachmentSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="ListsSoap" type="tns:ListsSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetList">
      <soap:operation soapAction="http://schemas.microsoft.com/sharepoint/soap/GetList"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetListItemChanges">
      <soap:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/GetListItemChanges" style="document"
/>
      <wsdl:input>
        <soap:body use="literal" />
```

```
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetListItemChangesSinceToken">
      <soap:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/GetListItemChangesSinceToken"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="UpdateListItems">
      <soap:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/UpdateListItems" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="AddDiscussionBoardItem">
      <soap:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/AddDiscussionBoardItem"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="AddAttachment">
      <soap:operation soapAction="http://schemas.microsoft.com/sharepoint/soap/AddAttachment"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetAttachmentCollection">
      <soap:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/GetAttachmentCollection"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="DeleteAttachment">
```

```
      <soap:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/DeleteAttachment" style="document"
/>
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="ListsSoap12" type="tns:ListsSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetList">
      <soap12:operation soapAction="http://schemas.microsoft.com/sharepoint/soap/GetList"
style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetListItemChanges">
      <soap12:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/GetListItemChanges" style="document"
/>
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetListItemChangesSinceToken">
      <soap12:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/GetListItemChangesSinceToken"
style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="UpdateListItems">
      <soap12:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/UpdateListItems" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="AddDiscussionBoardItem">
      <soap12:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/AddDiscussionBoardItem"
style="document" />
      <wsdl:input>
```

```
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="AddAttachment">
      <soap12:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/AddAttachment" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetAttachmentCollection">
      <soap12:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/GetAttachmentCollection"
style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="DeleteAttachment">
      <soap12:operation
soapAction="http://schemas.microsoft.com/sharepoint/soap/DeleteAttachment" style="document"
/>
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
</wsdl:definitions>
```

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:

- Microsoft® Office Outlook® 2003
- Microsoft® Office Outlook® 2007
- Microsoft® Outlook® 2010
- Microsoft® SharePoint® Foundation 2010
- Microsoft® Office SharePoint® Server 2007
- Windows® SharePoint® Services 2.0
- Windows® SharePoint® Services 3.0

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 1: Office Outlook 2003 and Office Outlook 2007 use the Lists Client Sync Protocol to provide an offline scenario for users of Windows SharePoint Services 2.0 and Windows SharePoint Services 3.0.

<2> Section 1.7: Windows SharePoint Services 2.0 does not support **GetListItemChangesSinceToken**.

<3> Section 1.7: Office Outlook 2003 supports only **GetListItemChanges** and not **GetListItemChangesSinceToken**.

<4> Section 1.7: Office Outlook 2007 uses the server version element in **GetListResponse** (see section 3.1.4.5 and [MS-LISTSWS]) to determine whether the server supports **GetListItemChangesSinceToken**. A value of "12.0.0.4326" or greater indicates the server supports **GetListItemChangesSinceToken**.

<5> Section 3.1.4.2: Windows SharePoint Services 3.0 uses **AddDiscussionBoardItem** to apply the same processing rules that incoming e-mail receives, which includes assigning one e-mail to be the root item described in section 3.2.1.3.

<6> Section 3.1.4.5.1.1: Office Outlook 2003 and Office Outlook 2007 ignore this attribute.

<7> Section 3.1.4.7: Office Outlook 2003 uses **GetListItems** because Office Outlook 2003 does not support **GetListItemChangesSinceToken**.

<8> Section 3.1.4.7: Office Outlook 2007 sets **rowLimit** to 100 by **default.rowLimit** can be configured to use a number between 15 and 1000. Office Outlook 2007 omits **rowLimit** when using this protocol to download appointments.

<9> Section 3.1.4.7: Office Outlook 2003 and Windows SharePoint Services 2.0 do not support the "v3" recurrence patterns listed here.

<10> Section 3.1.4.8: The **Translate** header is a Microsoft extension to the HTTP specification used in conjunction with WebDAV functionality.

<11> Section 3.2.1.1.2: WSS2, WSS3, and WSS4 allow this. Out2003, Out2007, and Out2010 do not allow this.

<12> Section 3.2.1.1.3: Windows SharePoint Services 3.0 does not delete exception items when the recurrence is deleted. Office Outlook 2003 and Office Outlook 2007 treat exception items that have no recurrence as single appointments (section 3.2.1.1.1).

<13> Section 3.2.1.1.3: Office Outlook 2003 and Office Outlook 2007 do not support restoring instances of a recurrence once they have become exceptions. Windows SharePoint Services 2.0 and Windows SharePoint Services 3.0 allow this.

<14> Section 3.2.1.1.4: Windows SharePoint Services 2.0 and Windows SharePoint Services 3.0 can be configured to display deleted instance items.

<15> Section 3.2.1.1.4: Windows SharePoint Services 2.0 and Windows SharePoint Services 3.0 convert exception items to deleted instance items and remove some of the data on the items. Office Outlook 2007 deletes the exception item and uploads a deleted instance item to replace it.

<16> Section 3.2.4.1: Office Outlook 2003 ignores **MinTimeBetweenSyncs**. Office Outlook 2007 can be configured to ignore **MinTimeBetweenSyncs**.

<17> Section 3.2.4.2: Office Outlook 2003 reads only the following fields, ignores all other appointment-specific fields, and reads these only if they are in the server's current schema:

- Description
- Duration
- EndDate
- EventDate
- EventType
- FooterInfo
- fRecurrence
- HeaderInfo
- LinkTitle
- Location
- RecurrenceData
- RecurrenceID
- UID
- XMLTZone

<18> Section 3.2.4.2: Office Outlook 2003 reads only the following field, ignores all other contact-specific fields, and reads these only if they are in the server's current schema:

- Birthday
- CellPhone
- Comments
- Company
- CompanyPhonetic
- Email
- FileAs
- FirstName
- FirstNamePhonetic
- HomeAddress
- HomeCity
- HomeCountry
- HomePhone
- HomeState
- HomeZip
- JobTitle
- LastNamePhonetic
- MiddleName
- NameTitle
- OfficeLocation
- Suffix
- Title
- WebPage
- Wedding
- WorkAddress
- WorkCity
- WorkCountry
- WorkFax
- WorkPhone

- WorkState
- WorkZip

<19> Section 3.2.4.2.1: Office Outlook 2003 reads only the following fields and ignores all other common properties:

ID

Attachments

<20> Section 3.2.4.2.1: Office Outlook 2007 will use **ReplicationID** only if it is a property bag (1) field.

<21> Section 3.2.4.2.1: Office Outlook 2007 will use **vti_versionhistory** only if it is a property bag (1) field.

<22> Section 3.2.4.2.1: Office Outlook 2003 ignores **ContentTypeId**. Office Outlook 2007 ignores **ContentTypeId** on calendar lists.

<23> Section 3.2.4.2.1: Office Outlook 2007 and Windows SharePoint Services 3.0 can remove old GUID-integer pairs if the version history becomes large.

<24> Section 3.2.4.2.2: Office Outlook 2003 reads only the following fields, ignores all other appointment-specific fields, and reads these only if they are in the server's current schema:

- Description
- Duration
- EndDate
- EventDate
- EventType
- FooterInfo
- fRecurrence
- HeaderInfo
- LinkTitle
- Location
- RecurrenceData
- RecurrenceID
- UID
- XMLTZone

<25> Section 3.2.4.2.2: Windows SharePoint Services 2.0 does not have the **fAllDayEvent** field. Office Outlook 2003 ignores the **fAllDayEvent** field.

<26> Section 3.2.4.2.3: Windows SharePoint Services 2.0 does not delete exception items when these properties are updated.

<27> Section 3.2.4.2.4: Office Outlook 2003 reads only the following fields, ignores all other contact-specific fields, and reads these only if they are in the server's current schema:

- Birthday
- CellPhone
- Comments
- Company
- CompanyPhonetic
- Email
- FileAs
- FirstName
- FirstNamePhonetic
- HomeAddress
- HomeCity
- HomeCountry
- HomePhone
- HomeState
- HomeZip
- JobTitle
- LastNamePhonetic
- MiddleName
- NameTitle
- OfficeLocation
- Suffix
- Title
- WebPage
- Wedding
- WorkAddress
- WorkCity
- WorkCountry
- WorkFax
- WorkPhone

- WorkState
- WorkZip

<28> Section 3.2.4.2.4: Office Outlook 2003 and Office Outlook 2007 use **Wedding** instead of **Anniversary** for this field.

<29> Section 3.2.4.2.4: Office Outlook 2007 interprets this field as an integer Gender only. Clients SHOULD interpret it as a choice type when the server includes CHOICE and MAPPING elements.

<30> Section 3.2.4.2.4: Office Outlook 2003 and Office Outlook 2007 use **HomeCity** instead of **HomeAddressCity** for this field.

<31> Section 3.2.4.2.4: Office Outlook 2003 and Office Outlook 2007 use **HomeCountry** instead of **HomeAddressCountry** for this field.

<32> Section 3.2.4.2.4: Office Outlook 2003 and Office Outlook 2007 use **HomeZip** instead of **HomeAddressPostalCode** for this field.

<33> Section 3.2.4.2.4: Office Outlook 2003 and Office Outlook 2007 use **HomeState** instead of **HomeAddressStateOrProvince** for this field.

<34> Section 3.2.4.2.4: Office Outlook 2003 and Office Outlook 2007 use **OfficeLocation** instead of **Office** for this field.

<35> Section 3.2.4.2.4: Office SharePoint Server 2007 does not include the MAPPINGS element for the Gender field.

<36> Section 3.2.4.2.8: Office Outlook 2007 serves as a pass-through for this data.

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

M

N

O

P

R

S

T

U

V

W