[MS-ASDTYPE]: Exchange ActiveSync: Data Types

Revision Summary

Date	Revision History	Revision Class	Comments
12/03/2008	1.0.0	Major	Initial Release.
03/04/2009	1.0.1	Editorial	Revised and edited technical content.
04/10/2009	2.0.0	Major	Updated technical content and applicable product releases.
07/15/2009	3.0.0	Major	Revised and edited for technical content.
11/04/2009	4.0.0	Major	Updated and revised the technical content.
02/10/2010	5.0.0	Major	Updated and revised the technical content.
05/05/2010	6.0.0	Major	Updated and revised the technical content.
08/04/2010	7.0	Major	Significantly changed the technical content.
11/03/2010	7.1	Minor	Clarified the meaning of the technical content.
03/18/2011	7.2	Minor	Clarified the meaning of the technical content.
08/05/2011	8.0	Major	Significantly changed the technical content.
10/07/2011	9.0	Major	Significantly changed the technical content.
01/20/2012	10.0	Major	Significantly changed the technical content.
04/27/2012	10.1	Minor	Clarified the meaning of the technical content.
07/16/2012	11.0	Major	Significantly changed the technical content.
10/08/2012	11.1	Minor	Clarified the meaning of the technical content.
02/11/2013	11.1	No change	No changes to the meaning, language, or formatting of the technical content.
07/26/2013	12.0	Major	Significantly changed the technical content.
11/18/2013	12.0	No change	No changes to the meaning, language, or formatting of the technical content.
02/10/2014	12.0	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

The Exchange ActiveSync: Data Types describes the required format of each data type used by the ActiveSync **XML schema definitions (XSDs)**.

This protocol sends and receives data in **Wireless Application Protocol (WAP) Binary XML (WBXML)** format. To ensure that both the client and the server have the same expectations about the format of the element data, the ActiveSync commands and classes use XSDs to define the data type of each element.

Sections 1.7 and 2 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. All other sections and examples in this specification are informative.

1.1 Glossary

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

Augmented Backus-Naur Form (ABNF)
Coordinated Universal Time (UTC)
Hypertext Transfer Protocol (HTTP)
Secure Sockets Layer (SSL)
Unicode
XML

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

base64 encoding
meeting
Meeting object
organizer
Wireless Application Protocol (WAP) Binary XML (WBXML)
XML schema
XML schema definition (XSD)

The following terms are specific to this document:

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

1.2 References

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

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[ISO-8601] International Organization for Standardization, "Data Elements and Interchange Formats - Information Interchange - Representation of Dates and Times", ISO/IEC 8601:2004, December

2004, http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=40874&ICS1=1&ICS2=140&ICS3=30

Note There is a charge to download the specification.

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

[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

[RFC822] Crocker, D.H., "Standard for ARPA Internet Text Messages", STD 11, RFC 822, August 1982, http://www.ietf.org/rfc/rfc0822.txt

[WBXML1.2] Martin, B., and Jano, B., Eds., "WAP Binary XML Content Format", W3C Note, June 1999, http://www.w3.org/1999/06/NOTE-wbxml-19990624

[XMLSCHEMA1/2] Thompson, H.S., Beech, D., Maloney, M., and Mendelsohn, N., Eds., "XML Schema Part 1: Structures Second Edition", W3C Recommendation, October 2004, http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

[XMLSCHEMA2/2] Biron, P.V., and Malhotra, A., Eds., "XML Schema Part 2: Datatypes Second Edition", W3C Recommendation, October 2004, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

1.2.2 Informative References

[MS-ASAIRS] Microsoft Corporation, "Exchange ActiveSync: AirSyncBase Namespace Protocol".

[MS-ASCAL] Microsoft Corporation, "Exchange ActiveSync: Calendar Class Protocol".

[MS-ASCMD] Microsoft Corporation, "Exchange ActiveSync: Command Reference Protocol".

[MS-ASCNTC] Microsoft Corporation, "Exchange ActiveSync: Contact Class Protocol".

[MS-ASCON] Microsoft Corporation, "Exchange ActiveSync: Conversations Protocol".

[MS-ASDOC] Microsoft Corporation, "Exchange ActiveSync: Document Class Protocol".

[MS-ASEMAIL] Microsoft Corporation, "Exchange ActiveSync: Email Class Protocol".

[MS-ASMS] Microsoft Corporation, "Exchange ActiveSync: Short Message Service (SMS) Protocol".

[MS-ASNOTE] Microsoft Corporation, "Exchange ActiveSync: Notes Class Protocol".

[MS-ASPROV] Microsoft Corporation, "Exchange ActiveSync: Provisioning Protocol".

[MS-ASRM] Microsoft Corporation, "Exchange ActiveSync: Rights Management Protocol".

[MS-ASTASK] Microsoft Corporation, "Exchange ActiveSync: Tasks Class Protocol".

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

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

[MS-OXPROTO] Microsoft Corporation, "Exchange Server Protocols System Overview".

[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

1.3 Overview

This protocol describes a set of data types that are used by the ActiveSync protocols to format data that is transferred between clients and servers. This protocol uses types defined by the **XML schema** data types definition, as described in [XMLSCHEMA2/2], and describes structured string types. Structured string types extend the **string** data type, as described in [XMLSCHEMA2/2], to contain more complex data.

1.4 Relationship to Protocols and Other Structures

This protocol depends on the XML schema data types definition, as described in [XMLSCHEMA2/2]. The following protocols depend on this protocol:

- The Exchange ActiveSync: AirSyncBase Namespace Protocol, as described in [MS-ASAIRS]
- The Exchange ActiveSync: Calendar Class Protocol, as described in [MS-ASCAL]
- The Exchange ActiveSync: Command Reference Protocol, as described in [MS-ASCMD]
- The Exchange ActiveSync: Contact Class Protocol, as described in [MS-ASCNTC]
- The Exchange ActiveSync: Conversations Protocol, as described in [MS-ASCON]
- The Exchange ActiveSync: Document Class Protocol, as described in [MS-ASDOC]
- The Exchange ActiveSync: Email Class Protocol, as described in [MS-ASEMAIL]
- The Exchange ActiveSync: Short Message Service (SMS) Protocol, as described in [MS-ASMS]
- The Exchange ActiveSync: Notes Class Protocol, as described in [MS-ASNOTE]
- The Exchange ActiveSync: Provisioning Protocol, as described in [MS-ASPROV]
- The Exchange ActiveSync: Rights Management Protocol, as described in [MS-ASRM]
- The Exchange ActiveSync: Tasks Class Protocol, as described in [MS-ASTASK]

For conceptual background information and overviews of the relationships and interactions between this and other protocols, see [MS-OXPROTO].

1.5 Applicability Statement

The data types specified in this document are applicable to all ActiveSync schemas.

1.6 Versioning and Capability Negotiation

None.

1.7 Vendor-Extensible Fields

None.

2 Structures

The following sections describe data types used by the ActiveSync protocols. All data sent by the ActiveSync protocol is text, but some of the text values adhere to the following text style data types, as specified by the schemas.

2.1 boolean Data Type

A **boolean** is an XML schema primitive data type, as specified in [XMLSCHEMA2/2] section 3.2.2. It is declared as an **element** with a **type** attribute of "boolean".

The value of a **boolean** element is an integer whose only valid values are 1 (TRUE) or 0 (FALSE). If the integer value is missing, then it is assumed to be 1 (TRUE). For examples, see section 3.1. Elements with a **boolean** data type MUST be encoded and transmitted as [WBXML1.2] inline strings.

2.2 container Data Type

A **container** is an **XML** element that encloses other elements but has no value of its own. It is a complex type with complex content, as specified in [XMLSCHEMA1/2] section 3.4.2. It is defined using a **complexType** element that specifies the allowable children for that element using the **element** tag.

2.3 dateTime Data Type

A **dateTime** is a primitive XML schema data type, as specified in [XMLSCHEMA2/2] section 3.2.7. It is declared as an **element** whose **type** attribute is set to "dateTime".

dateTime values are as specified in [ISO-8601].

All dates are given in **Coordinated Universal Time (UTC)** and are represented as a string in the following format.

YYYY-MM-DDTHH:MM:SS.MSSZ where

YYYY = Year (Gregorian calendar year)

MM = Month (01 - 12)

DD = Day (01 - 31)

HH = Number of complete hours since midnight (00 - 24)

MM = Number of complete minutes since start of hour (00 - 59)

SS = Number of seconds since start of minute (00 - 59)

MSS = Number of milliseconds. This portion of the string is optional.

The T serves as a separator, and the Z indicates that this time is in UTC.

For example, 8:35 A.M. on December 25, 2000 would be represented as 2000-12-25T08:35:00.000Z.

Elements with a **dateTime** data type MUST be encoded and transmitted as [WBXML1.2] inline strings.

2.3.1 Time Zones and Daylight Saving Time

Dates and times can be very simple in calendars that are not shared. All times can be in device-local time, and there is no need for time zones or Daylight Saving Time (DST). If a **meeting** is scheduled for 10:00 A.M., it is in device time and, if the user of the device travels to another time zone, he or she adjusts the device time, but the meeting time remains at 10:00 A.M. If DST begins, the device time is adjusted again, but the meeting time remains at 10:00 A.M.

Dates and times become more complex when calendar events are shared by people who are in different time zones and are not all on DST. If Sean in Seattle schedules a 10:00 A.M. conference call with Nick in New York, the meeting will appear at 1:00 P.M. on Nick's calendar. If Jeff in Arizona is also on the call, he sees the meeting in his local time on his calendar. Because Arizona does not observe DST, the meeting is shown at 11:00 A.M. if it is the winter, but at 10:00 A.M. if it is the summer. If the meeting is recurring, then the dates and times are more complex during the transitions between DST and standard time. The following table lists the local and UTC times for a 10:00 A.M. meeting the weeks before and after the transition to DST.

Date	Seattle	Arizona	New York	UTC
4/4/03	10:00 Pacific Time (PT)	11:00 MST (Mountain Standard Time)	13:00 Eastern Standard Time (EST)	18:00 UTC
4/11/03	10:00 Pacific Daylight Time (PDT)	10:00 MST	13:00 Eastern Daylight Time (EDT)	17:00 UTC

The Seattle time remains the same before and after the transition to DST because the meeting **organizer** is in Seattle. If the organizer was Jeff in Arizona, then the meeting times before and after the DST transition would be different, as shown in the following table.

Date	Seattle	Arizona	New York	UTC
4/4/03	10:00 PT	11:00 MST	13:00 EST	18:00 UTC
4/11/03	11:00 PDT	11:00 MST	14:00 EDT	18:00 UTC

The shared **Meeting object** in the calendar application stores the following information. For a one-time meeting, the UTC time alone can be stored, and each device can translate to its local time by using its local time zone information. The time zone information includes a permanent time zone offset and, if appropriate, DST start and end dates, and time bias.

If the meeting is recurring, however, the UTC time can change depending on whether DST is in effect at the originator's location for each occurrence. The constant is the time in the originator's time zone, which is the time that is stored. In addition, the originator's time zone is stored. To display a meeting time, the time is converted to UTC by using the originator's time zone, and then it is converted to local time by using the device's local time zone.

Note: The UTC time can be stored instead of the originator's local time. But the originator's time zone is also stored. This feature allows for the DST adjustment, although the calculation is somewhat less intuitive.

If this recurring meeting has an exception, then the exception contains the date and time of the series instance that is different. As with the series itself, the UTC of the exception varies based on DST. Therefore, the originator's time zone is used to calculate the time of the exception. Because the originator's time zone is stored with the recurrence, it is not necessary to store the time zone again for each exception.

2.3.2 Calculating Dates and Times

The ActiveSync protocols use the UTC time and the originator's time zone for all meetings. For single occurrences, the device converts the time to the local time zone. The originator's time zone is not important because the original conversion to UTC accounts for time zone and DST. However, for recurring meetings, there is the possibility of a transition into or out of DST during the series. The stored UTC corresponds to the first occurrence of the series, but later meetings can have different corresponding UTC times. Therefore, to display the correct time, the device performs one calculation that accounts for the originator's time zone, in addition to the device's local time zone.

The following table shows the time zone information for the earlier examples.

Time zone information	Pacific Time	Mountain Time (Arizona)	Eastern Time
Time zone offset	UTC-8	UTC-7	UTC-5
Daylight start	4/6/03 02:00	None	4/6/03 02:00
Daylight end	10/26/03 02:00	None	10/26/03 02:00
Daylight bias	+1	0	+1

The calculation to display the local time of a meeting instance is as follows:

(Meeting time in UTC) + (local time zone offset) + (local daylight bias) – (original daylight bias)

Note: Daylight bias is a time zone's offset during DST. The local daylight bias comes from the local time zone information, and the original daylight bias comes from the originator's time zone information.

The weekly conference call repeats every Friday beginning 4/4/03. The start time of the first instance is 10:00 A.M. PT, or 18:00 UTC. Therefore, the stored time is 18:00 and the time zone is Pacific Time.

Date	Seattle	Arizona	New York
4/4/03	1800+(-8)+(0)-(0) = 1000	1800+(-7)+(0)-(0) = 1100	1800+(-5)+(0)-(0) = 1300
4/11/03	1800+(-8)+(+1)-(+1) = 1000	1800+(-7)+(0)-(+1) = 1000	1800+(-5)+(+1)-(+1) = 1300

Notice that both the local and original DST biases are the ones in effect on the date/time of the meeting instance.

The weekly conference call repeats every Friday beginning on 4/4/03. The originator was in Arizona, so the start time of the first instance is 11:00 MST (Arizona), or 18:00 UTC. The stored time is 18:00 and the time zone is MST (Arizona).

Date	Seattle	Arizona	New York
4/4/03	1800+(-8)+(0)-(0) = 1000	1800+(-7)+(0)-(0) = 1100	1800+(-5)+(0)-(0) = 1300
4/11/03	1800+(-8)+(+1)-(0) = 1100	1800+(-7)+(0)-(0) = 1100	1800+(-5)+(+1)-(0) = 1400

2.4 enumeration Data Type

An **enumeration** specifies a fixed set of values for an element or attribute. In accordance with [XMLSCHEMA2/2] section 4.3.5, it is specified using the **restriction** element to declare the enumeration, and the **enumeration** element to define one or more allowed values.

2.5 integer Data Type

An **integer** is a numeric value that can be provided in the XML body of a command. It is an XML schema primitive data type, as specified in [XMLSCHEMA2/2] section 3.3.13. Elements with an **integer** data type MUST be encoded and transmitted as [WBXML1.2] inline strings.

2.6 string Data Type

A **string** is a chunk of **Unicode** text. It is an XML schema primitive data type as specified in [XMLSCHEMA2/2] section 3.2.1. An element of this type is declared as an **element** with a **type** attribute of "string".

Elements with a **string** data type MUST be encoded and transmitted as [WBXML1.2] inline strings.

Some **string** values are constrained to a particular set of values, which is included in the description of the element.

ActiveSync defines several conventions for strings that adhere to commonly used formats:

- **Byte Array** (section 2.6.1)
- **E-mail Address** (section 2.6.2)
- **Telephone Number** (section 2.6.3)
- **TimeZone** (section 2.6.4)
- **Compact DateTime** (section 2.6.5)

Elements of these types are defined as **string** types in XML schemas, but commands that process such elements can return an error if the value of the element does not adhere to the expected format.

2.6.1 Byte Array

A **byte array** is a structure inside of an element of the **string** type (section 2.6). The structure is comprised of a length, which is expressed as a multi-byte integer, as specified in [WBXML1.2], followed by that many bytes of data. Elements with a **byte array** structure MUST be encoded and transmitted as [WBXML1.2] opaque data.

2.6.2 E-Mail Address

An e-mail address is an unconstrained value of an element of the **string** type (section 2.6).

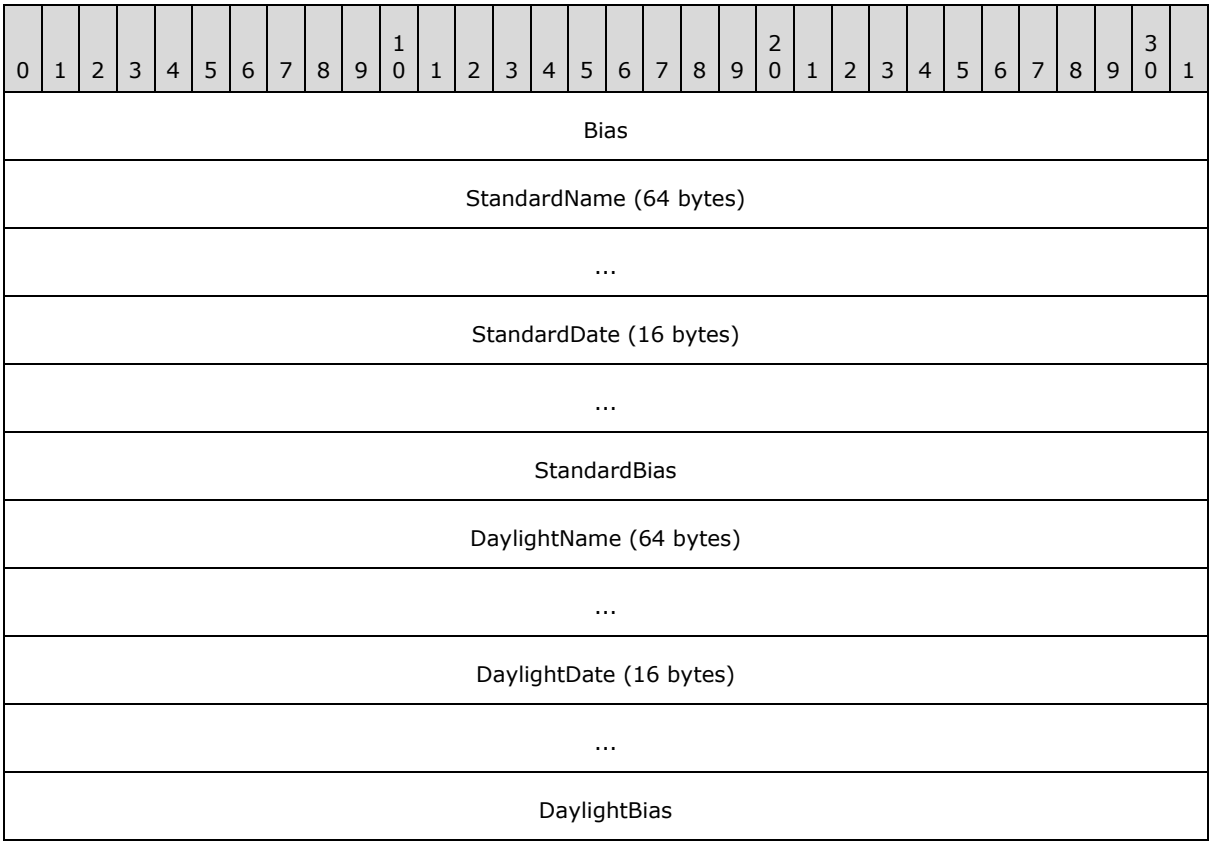
However, a valid individual e-mail address MUST have the following format: "local-part@domain". For more information about e-mail address syntax, see [RFC822] section 6.

2.6.3 Telephone Number

A telephone number is an unconstrained value of elements of the **string** type (section 2.6) that can include an area code and a country code.

2.6.4 TimeZone

The **TimeZone** structure is a structure inside of an element of the **string** type (section 2.6).



Bias (4 bytes): The value of this field is a **LONG,** as specified in [MS-DTYP]. The offset from UTC, in minutes. For example, the bias for Pacific Time (UTC-8) is 480.

StandardName (64 bytes): The value of this field is an array of 32 **WCHARs**, as specified in [MS-DTYP]. It contains an optional description for standard time. Any unused **WCHARs** in the array MUST be set to 0x0000.

StandardDate (16 bytes): The value of this field is a **SYSTEMTIME** structure, as specified in [MS-DTYP]. It contains the date and time when the transition from DST to standard time occurs.

StandardBias (4 bytes): The value of this field is a **LONG**. It contains the number of minutes to add to the value of the **Bias** field during standard time.

DaylightName (64 bytes): The value of this field is an array of 32 **WCHARs**. It contains an optional description for DST. Any unused **WCHARs** in the array MUST be set to 0x0000.

DaylightDate (16 bytes): The value of this field is a **SYSTEMTIME** structure. It contains the date and time when the transition from standard time to DST occurs.

DaylightBias (4 bytes): The value of this field is a **LONG**. It contains the number of minutes to add to the value of the **Bias** field during DST.

The **TimeZone** structure is encoded using **base64 encoding** prior to being inserted in an XML element. Elements with a **TimeZone** structure MUST be encoded and transmitted as [WBXML1.2] inline strings.

2.6.5 Compact DateTime

A **Compact DateTime** value is a representation of a **UTC** date and time within an element of type **xs:string**, as specified in [XMLSCHEMA2/2] section 3.2.1. The format of a **Compact DateTime** value is specified by the following **Augmented Backus-Naur Form (ABNF)** notation.

```
date_string   = year month day "T" hour minute seconds [milliseconds] "Z"
year          = 4*DIGIT
month         = ("0" DIGIT) / "10" / "11" / "12"
day           = ("0" DIGIT) / ("1" DIGIT) / ("2" DIGIT) / "30" / "31"
hour          = ("0" DIGIT) / ("1" DIGIT) / "20" / "21" / "22" / "23"
minute        = ("0" DIGIT) / ("1" DIGIT) / ("2" DIGIT) / ("3" DIGIT) / ("4" DIGIT) / ("5"
DIGIT)
seconds       = ("0" DIGIT) / ("1" DIGIT) / ("2" DIGIT) / ("3" DIGIT) / ("4" DIGIT) / ("5"
DIGIT)
milliseconds  = 1*3DIGIT
```

2.7 unsignedByte Data Type

The **unsignedByte** data type is an integer value between 0 and 255, inclusive. It is an XML schema primitive data type as specified in [XMLSCHEMA2/2] section 3.3.24. Elements of this type are declared with an **element** whose **type** attribute is set to "unsignedByte".

3 Data Type Examples

3.1 boolean Example

Note in the following example that the short form "<Tag />" is equivalent to "<Tag>1</Tag>".

```
<email:Read>0</email:Read>
<email:AllDayEvent>1</email:AllDayEvent>
<email:AllDayEvent />
```

3.2 container Example

In the following example, **FolderCreate** is a container.

```
<?xml version="1.0" encoding="utf-8"?>
<FolderCreate xmlns="FolderHierarchy:">
<FolderCreate>
   <ServerId>1</ServerId>
   <ParentId>0</ParentId>
   <DisplayName>Calendar</DisplayName>
   <Type>8</Type>
</FolderCreate>
```

3.3 dateTime Examples

The following example demonstrates the **dateTime** format as used by the Email class, as described in [MS-ASEMAIL].

```
<?xml version="1.0" encoding="utf-8"?>
<Sync xmlns:email="Email:" xmlns:airsyncbase="AirSyncBase:" xmlns:email2="Email2:"
xmlns="AirSync:">

<A:DateReceived>2009-11-12T00:45:06.000Z</A:DateReceived>
```

The following example demonstrates the **dateTime** format used by the Calendar class, as described in [MS-ASCAL].

```
<?xml version="1.0" encoding="utf-8"?>
<Sync xmlns="AirSync:" xmlns:calendar="Calendar:" xmlns:airsyncbase="AirSyncBase:">

...

<airsyncbase:StartTime>20091212T000000Z</airsyncbase:StartTime>

...
```

3.4 enumeration Example

The allowed **enumeration** values are defined in the schema.

```
<xs:element name="UserResponse">
    <xs:simpleType>
        <xs:restriction base="xs:unsignedByte">
            <xs:enumeration value="3"/>
            <xs:enumeration value="1"/>
            <xs:enumeration value="2"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

3.5 integer Example

```
<airsyncbase:TruncationSize>456</airsyncbase:TruncationSize>
<airsync:FilterType>3</ airsync:FilterType>
<airsync:Status>1</airsync:Status>
```

3.6 string Example

```
<contact:CompanyName>Adventure Works</contact:CompanyName>
<contact:BusinessPhoneNumber>(800) 555-0100</contact:BusinessPhoneNumber>
<email:MessageClass>IPM.NOTE</email:MessageClass>
```

3.6.1 Byte Array Example

In this example, the continuation flag (as described in [WBXML1.2]) is not set, indicating that the length is only one byte long. This results in a length of 4 bytes. The following 4 bytes compromise the data.

```
04 00 01 02 03
```

3.6.2 E-Mail Address Example

```
<resolverecipients:Recipient>amy@nowhere.com</resolverecipients:Recipient>
<email2:Sender>j.smith@nowhere.com</email2:Sender>
```

3.6.3 Telephone Number Example

```
<contacts:HomePhoneNumber>3605551212</contacts:HomePhoneNumber>
<contacts:BusinessPhoneNumber>+011(73)5551212</contacts:BusinessPhoneNumber>
```

3.6.4 TimeZone Example

```
<email:TimeZone>
4AEAACgARwBNAFQALQAwADgAOgAwADAAKQAgAFAAYQBjAGkAZgBpAGMAIABUAGkAbQBlACAAKABVA
FMAIAAmACAAQwAAAAsAAAABAAIAAAAAAAAAAAAAAACgARwBNAFQALQAwADgAOgAwADAAKQAgAFAAYQ
BjAGkAZgBpAGMAIABUAGkAbQBlACAAKABVAFMAIAAmACAAQwAAAAMAAAACAAIAAAAAAAAAxP///w=
=
</email:TimeZone>
```

3.6.5 Compact DateTime Example

In the following example, 9:00 A.M. UTC on July 22, 2013, is represented as a **Compact DateTime** value.

```
20130722T090000Z
```

3.7 unsignedByte Example

```
<calendar:BusyStatus>3</calendar:BusyStatus>
```

4 Security Considerations

In most cases, all communication between the client and server happens across an **HTTP** connection secured by the **Secure Sockets Layer (SSL)** protocol, as described in [RFC2616]. The SSL connection is assumed to be secure enough to transmit confidential data, such as user credentials and sensitive e-mail. The SSL certificate on the server is assumed to be trusted by the client application.

5 Appendix A: 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 Exchange Server 2007 Service Pack 1 (SP1)
- Microsoft Exchange Server 2010
- Microsoft Exchange Server 2013
- Windows 8.1
- Windows Communication Apps

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.

6 Change Tracking

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

7 Index