[MS-ASDTYPE]: Exchange ActiveSync Data Types

Intellectual Property Rights Notice for Protocol Documentation

- **Copyrights.** This protocol 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 protocols, and may distribute portions of it in your implementations of the protocols 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 protocol documentation.
- **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 protocols. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, the protocols may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp). If you would prefer a written license, or if the protocols are not covered by the OSP, patent licenses are available by contacting protocol@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.

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. This protocol documentation is 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. A protocol specification does 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.

Revision Summary			
Author	Date	Version	Comments
Microsoft Corporation	December 3, 2008	1.0	Initial Release.
Microsoft Corporation	March 4, 2009	1.01	Revised and edited technical content.

Table of Contents

1 Introduction

This document specifies the required format of each data type used by the Exchange ActiveSync **XML schema definitions (XSDs)**.

The Exchange ActiveSync protocol sends and receives data in **Wide Area Protocol (WAP) Binary XML (WBXML)** format. In order to ensure that both the **client** and the **server** have the same expectations about the format of the element data, the Exchange ActiveSync commands and classes use XSDs to define the data type of each element.

1.1 Glossary

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

class
client
Coordinated Universal Time (UTC)
server
Unicode
WAP Binary XML (WBXML)
XML
XML schema definition (XSD)

The following terms are specific to this document:

Base64: A binary-to-text encoding scheme whereby an arbitrary sequence of bytes is converted to a sequence of printable ASCII characters.

organizer: The owner of an event.

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

[ISO-8601] International Organization for Standardization, "Data Elements and Interchange Formats - Information Interchange - Representation of Dates and Times", ISO 8601:2004, December 2004, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=40874.

[MS-ASAIRS] Microsoft Corporation, "ActiveSync AirSyncBase Namespace Protocol Specification", December 2008.

[MS-ASCAL] Microsoft Corporation, "ActiveSync Calendar Class Protocol Specification", December 2008.

[MS-ASCMD] Microsoft Corporation, "ActiveSync Command Reference Protocol Specification", December 2008.

[MS-ASCNTC] Microsoft Corporation, "ActiveSync Contact Class Protocol Specification", December 2008.

[MS-ASDOC] Microsoft Corporation, "ActiveSync Document Class Protocol Specification", December 2008.

[MS-ASEMAIL] Microsoft Corporation, "ActiveSync E-Mail Class Protocol Specification", December 2008.

[MS-ASTASK] Microsoft Corporation, "ActiveSync Tasks Class Protocol Specification", December 2008.

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

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

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

[W3C-XSD2] World Wide Web Consortium, "XML Schema Part 2: Datatypes Second Edition", October 2004, http://www.w3.org/TR/xmlschema-2/.

1.2.2 Informative References

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

1.3 Relationship to Protocols and Other Structures

The data types in this document are used by the Exchange ActiveSync commands, as specified in [MS-ASCMD]. The data types are also used by each of the content classes, as specified in [MS-ASCAL], [MS-ASCNTC], [MS-ASDOC], [MS-ASEMAIL], and [MS-ASTASK], and by the AirSyncBase namespace, as specified in [MS-ASAIRS].

1.4 Applicability Statement

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

1.5 Versioning and Localization

None.

1.6 Vendor-Extensible Fields

None.

2 Data Types

The following sections describe data types used by the Exchange ActiveSync protocols. All data sent by the Exchange 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 String

A **string** is a chunk of **Unicode** text, as specified in [W3C-XSD2] section 3.2.1.

In most cases, the protocol does not limit the length of the text in a **string**, and the **client** SHOULD store all of the text that the **server** sends. If the client is unable to store all of the text sent from the server, then the client MAY store a truncated version of the text. If truncation is required, the client MAY make the truncated data read-only, or it MAY provide a warning to the user that data can be lost if the item is edited and synchronized with the server.

In certain contexts, however, the server limits the length of the string. For example, server IDs have a maximum size that the client MUST be able to store.

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

2.2 Integer

An **integer** is a numeric value that can be provided in the XML body of a command or in a **POST** command header, as specified in [W3C-XSD2] section 3.3.13.

Some integer values are constrained to a fixed set (**enumeration**), which is specified in the description of the element. For more details about **enumerations**, see section 2.9.

2.3 Boolean

A **boolean** is an integer whose only valid values are 1 (**TRUE**) or 0 (**FALSE**), as specified in [W3C-XSD2] section 3.2.2. If the integer value is missing, then it is assumed to be 1 (**TRUE**). For examples, see section 3.3.

2.4 Telephone Numbers

Telephone numbers are unconstrained **string** values that MAY include an area code and a country code.

2.5 E-Mail Addresses

E-mail addresses are unconstrained **string** values.

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 Date/Time

Date/time values are as specified in [ISO8601]. 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

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.

Note: Dates and times in calendar items MUST NOT include punctuation separators. For example: `<A:StartTime>20021126T160000Z</A:StartTime>`

2.6.1 Time Zones and Daylight Saving Time

Dates and times can be very simple in calendars that are not shared. All times MAY be in device-local time, and there is no need for time zones or Daylight Savings 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 Annette in New York, the meeting will appear at 1:00 P.M. on Annette's calendar. If Jeff in Arizona is also on the call, he should see 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 MUST store the following information. For a one-time meeting, the **UTC** time alone MAY be stored, and each device MAY 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 MAY change depending on whether daylight saving time 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 MUST be stored. In addition, the originator's time zone MUST be stored. To display a meeting time, the time MUST be converted to UTC by using the originator's time zone, and then it MUST be converted to local time by using the device's local time zone.

Note: The UTC time MAY be stored instead of the originator's local time. But the originator's time zone MUST still be stored. This feature allows for the daylight saving time 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 MUST be used to calculate the time of the

exception. Because the originator's time zone is stored with the recurrence, the time zone MAY NOT be stored again for each exception.

2.6.2 Calculating Dates and Times

The ActiveSync Exchange 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 Daylight Savings Time (DST). However, for recurring meetings, the device MUST consider 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 may have different corresponding UTC times. Therefore, to display the correct time, the device MUST perform 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.

	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)

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)	1800+(-7)+(0)-(+1) =	1800+(-5)+(+1)-(+1)

Date	Seattle	Arizona	New York
	= 1000	1000	= 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

2.7 TimeZone

The **TimeZone** type is a **Base64**-encoded time zone structure with the following definition:

```
typedef struct {
  LONG Bias;
  WCHAR StandardName[32];
  SYSTEMTIME StandardDate;
  LONG StandardBias;
  WCHAR DaylightName[32];
  SYSTEMTIME DaylightDate;
  LONG DaylightBias;
};
```

The required values are **Bias**, which is the offset from **UTC**, in minutes, and the **StandardDate** and **DaylightDate**, which are needed when the biases take effect. For example, the bias for Pacific Time is 480.

2.8 Container

A **container** is an XML element that encloses other elements but has no value of its own.

2.9 Enumeration

An **enumeration** is an **integer** value that has a fixed set of values. For more details about **integer** values, see section 2.2.

3 Structure Examples

3.1 String Examples

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

3.2 Integer Examples

```
<BodySize>456</BodySize>
<FilterType>3</FilterType>
<Status>1</Status>
```

3.3 Boolean Examples

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

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

3.4 Telephone Number Examples

```
<Home>3605551212</Home>
<Overseas>+011(73)5551212</Overseas>
```

3.5 E-Mail Address Examples

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

3.6 Date/Time Examples

```
<A:DateReceived>20091112T004506000Z</A:DateReceived>
<A:StartTime>20091121T193000000Z</A:StartTime>
```

3.7 Time Zone Example

```
<TimeZone>
```

```
4AEAACgARwBNAFQALQAwADgAOgAwADAAKQAgAFAAYQBjAGkAZgBpAGMAIABUAGkAbQBlACA
AKABVAFMAIAAmACAAQwAAAAsAAAABAAIAAAAAAAAAAAAAAACgARwBNAFQALQAwADgAOgAwAD
AAKQAgAFAAYQBjAGkAZgBpAGMAIABUAGkAbQBlACAAKABVAFMAIAAmACAAQwAAAAMAAAACA
AIAAAAAAAAAxP///w==
</TimeZone>
```

3.8 Container Example

In the following example, **<AddOne>** is a container.

```
<AddOne>
   <ServerId>1</ServerId>
   <ParentId>0</ParentId>
   <DisplayName>Calendar</DisplayName>
   <Type>8</Type>
</AddOne>
```

3.9 Enumeration Example

The allowed enumeration values are defined in the schema.

```
<EnumExample>3</EnumExample>
```

4 Security Considerations

In most cases, all communication between the **client** and **server** happens across a HTTP connection secured by the Secure Sockets Layer (SSL) protocol, as specified in [RFC2068]. 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: Office/Exchange Behavior

The information in this specification is applicable to the following versions of Office/Exchange:

- Microsoft Office 2003
- Microsoft Exchange 2003
- Microsoft Office 2007
- Microsoft Exchange 2007

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies

Office/Exchange behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies Office/Exchange does not follow the prescription.

Index