[MS-OXOAB]: Offline Address Book (OAB) Format and Schema Protocol Specification

Revision Summary			
Author	Date	Version	Comments
Microsoft Corporation	April 4, 2008	0.1	Initial Availability.
Microsoft Corporation	April 25, 2008	0.2	Revised and updated property names and other technical content.
Microsoft Corporation	June 27, 2008	1.0	Initial Release.
Microsoft Corporation	August 6, 2008	1.01	Revised and edited technical content.
Microsoft Corporation	September 3, 2008	1.02	Revised and edited technical content.

Table of Contents

1 Introduction

This document specifies the offline address book (OAB) version 2 and OAB version 4 file formats. OABs are files that store address list information on the client, so that the client can access the information when it does not have a network connection with the server or is working offline. This specification assumes the reader has familiarity with the address book concepts and requirements of the Address Book Object protocol, as specified in [MS-OXOABK]. Those concepts and requirements are not repeated in this specification.

1.1 Glossary

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

address book
Address Book object
address creation template
address list
alias
ambiguous name resolution (ANR)
ASCII
distinguished name (DN)
distribution list
GUID
Lempel-Ziv Extended (LZX)
Lempel-Ziv Extended Delta (LZXD)
little-endian
mailbox
message database (MBD)
offline address book (OAB)
public folder
property tag
relative distinguished name (RDN)
recipient
Rich Text Format (RTF)
Simple Mail Transfer Protocol (SMTP)
X500 DN

The following data types are defined in [MS-OXCDATA]:

PtypBinary
PtypBoolean
PtypInteger32
PtypMultipleInteger32
PtypMultipleString
PtypMultipleString8

PtypString
PtypString8

The following terms are specific to this document:

mail agent: An **Address Book object** other than a **remote mail user**, **mail user**, **distribution list**, or **public folder**.

narrow character set: A character set that represents text characters as a sequence of bytes, where each byte represents a unique character. The **ASCII** character set is a **narrow character set**.

parent DN (PDN): The **distinguished name** of the next immediate object closer to the root of the tree of **relative distinguished names (RDNs)**.

remote mail user: A collection of properties such as telephone numbers, e-mail addresses, and pager numbers pertaining to a person or business external to the messaging server.

X509: An ITU-T standard for Public Key Infrastructure subsequently adapted by the IETF, as specified in [RFC3280].

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/IEC 8802-3] International Organization for Standardization, "Information technology -- Telecommunications and information exchange between systems -- Local and metropolitan area networks -- Specific requirements -- Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications", ISO/IEC 8802-3:2000, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=31002.

[MS-MCI] Microsoft Corporation, "MCI Compression and Decompression", June 2008.

[MS-OXCDATA] Microsoft Corporation, "Data Structures Protocol Specification", June 2008.

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

[MS-OXOABK] Microsoft Corporation, "Address Book Object Protocol Specification", June 2008.

[MS-OXOABKT] Microsoft Corporation, "Address Book User Interface Templates Protocol Specification", June 2008.

[MS-OXPFOAB] Microsoft Corporation, "Offline Address Book (OAB) Public Folder Retrieval Protocol Specification", June 2008.

[MS-OXPROPS] Microsoft Corporation, "Exchange Server Protocols Master Property List Specification", June 2008.

[MS-PATCH] Microsoft Corporation, "LZX DELTA Compression and Decompression", June 2008.

[RFC2044] Yergeau, F., "UTF-8, a transformation format of Unicode and ISO 10646", RFC 2004, October 1996, http://www.ietf.org/rfc/rfc2044.txt.

[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.

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

1.2.2 Informative References

[ISO/IEC 8825-1] "ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ISO/IEC 8825-1:1998, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=32306.

[MS-ADTS] Microsoft Corporation, "Active Directory Technical Specification", July 2006, http://go.microsoft.com/fwlink/?LinkId=112149.

[MS-OXWOAB] Microsoft Corporation, "Offline Address Book (OAB) Retrieval Protocol Specification", June 2008.

[RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax", RFC 2315, March 1998, http://www.ietf.org/rfc/rfc2315.txt.

[RFC3280] Housley, R., Polk, W., Ford, W., and Solo, D., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002, http://www.ietf.org/rfc/rfc3280.txt.

1.3 Structure Overview

A server can choose to make user properties, such as job titles, addresses, and telephone numbers, available to its clients in an **address book**. The address book can then be browsed or searched by clients looking for recipient information. To organize the contents of an address book, the server can divide recipients into containers and the client can choose which container to browse or search.

Each address book container is known as an **address list**. The collection of available containers, or address lists, is the address book. When the client is unable to reach the server, which can be caused by working offline or having high network costs to access the server, the client can use a local copy of the address book or address lists to retrieve user information. The local copy of the address book is known as an **offline address book (OAB)**.

An OAB is composed of three or more files that provide the full functionality of the online address book when the client is working offline. This specification describes the structure of each of the files required to create an OAB version 2 and OAB version 4 file.

1.3.1 OAB Version 2

The OAB version 2 file format specifies the structure of files that are downloaded from the server to the client to support an offline address book. The OAB version 2 file consists of the following files:

- Browse file. The Browse file contains one fixed size record per user, with members that point to offsets in the RDN Index, ANR Index, and Details files. The fixed size record contains data and offsets that account for all of the user's data in the OAB version 2 file. For an overview of the Browse file, see section 1.3.1.1. For information about the structure of the Browse file, see section 2.3.

- RDN Index file. The **relative distinguished name (RDN)** Index file is used for primary key lookups based on the **X500 DN** and **Simple Mail Transfer Protocol (SMTP)** address properties of the **Address Book object**. For an overview of the RDN Index file, see section 1.3.1.2. For information about the structure of the RDN Index file, see section 2.4.

- ANR Index file. The ANR Index file is used for **ambiguous name resolution (ANR)**. Values for the display name, surname, office location, and e-mail **alias** are all sorted together into one structure so that a single search can find Address Book objects based on multiple properties. For an overview of the ANR Index file, see section 1.3.1.3. For information about the structure of the ANR Index file, see section 2.5.

- Details file. The Details file contains all other properties for **Address Book** objects in the version 2 OAB. The Details file is not indexed. The client can choose not to download the Details file in order to save space and bandwidth since there is no information in there that is required for basic e-mail addressing. For an overview of the Details file, see section 1.3.1.4. For information about the structure of the Details file, see section 2.6.

- Display Template files. For an overview of the Display Template file, see section 1.3.1.5. For information about the structure of the Display Template file used by OAB version 2 and OAB version 4, see section 2.2.

Each of these files is compressed before synchronization to save network bandwidth.

Figure 1 shows each of these OAB files and the indexes that point from one file to another. After an OAB has been downloaded to the client, incremental updates can be downloaded using a Changes file.

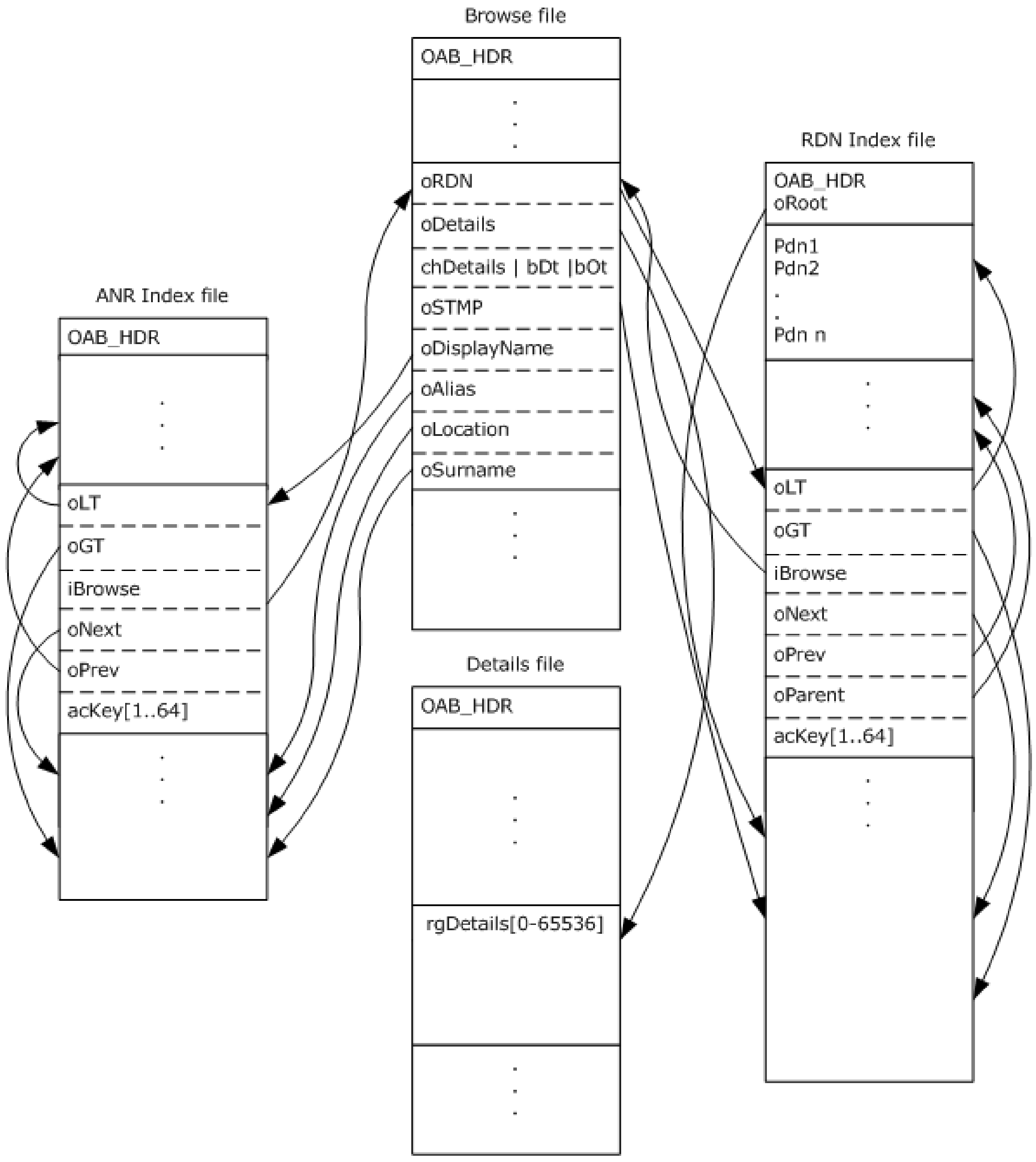


Figure 1: Relationship of the OAB version 2 files

1.3.1.1 Uncompressed Browse File

The Browse file is sorted in alphabetical order according to **Address Book** object display names and allows for fast paging of **Address Book** object data. It has offsets into the other files for the display name, the surname, the office location, the X500 DN, the SMTP address, the e-mail alias, and the details record. It also maintains values for the object type and **Address Book** object display type. Each record is a fixed size. Fetching an entire record requires that the client follow each link from the Browse file and retrieve data from the other files. The header of the Browse file includes a file type, a record count, and a serial number. The serial number is a rotating hash of the RDN value of each record in the Browse file order.

1.3.1.2 Uncompressed RDN Index File

The RDN Index file is split into two sections: the **parent distinguished name (PDN)** table and the RDN index. The PDN table contains the list of all parent **distinguished name** values for X500 DNs and all domain names used by SMTP addresses. The last RDN of the X500 DNs and the local-part of SMTP addresses are stored in the key field of the records in the RDN index section.

Records in the RDN index part of the file are of variable size, contain the index key value, and have pointers to the record in the PDN table so that the original value of the X500 DN or SMTP address can be reconstructed. In the record is an index of the related browse record in the Browse file and four more offsets are stored to create a threaded tree structure within the RDN Index file. An offset in the header of the RDN Index file points past the end of the PDN table to the root of the RDN index tree.

1.3.1.3 Uncompressed ANR Index File

The ANR Index file is structured similarly to the RDN Index file, but does not contain a PDN table. Each record is a variable size and has four offsets that construct a threaded tree structure. Records have an index of master records in the **Browse** file and the value portion is either an office location string, a surname string, an alias string, or a display name string. The root of the ANR index tree is always the first node in the file; therefore no root offset is required in the header.

1.3.1.4 Uncompressed Details File

The Details file contains variable size records that store a fixed set of properties for each **Address Book** object. Each record can be up to 65536 bytes long and all the stored properties for a single **Address Book** object have to fit into that record. The data is not indexed and there are no links from this file to any of the other files, but the Browse file does have links to this file.

1.3.1.5 Uncompressed Display Template File

The Template file describes how the **Address Book** object data can be presented to a user, as specified in [MS-OXOABKT].

1.3.1.6 Uncompressed Changes File

The Changes file describes the changes that need to happen to the other files to produce a file set that represents the next generational version of the OAB version 2 files. It consists of a sequence of variable size records that contain data to update individual records.

Numerous change files might be required to make a set of OAB version 2 files current with the server.

1.3.1.7 Compressed OAB Version 2 Files

OAB version 2 files are compressed by the server before being transferred to the client. A compressed file starts with a header and then a sequence of compressed blocks. All OAB version 2 files are compressed the same way. For more information about the compression of OAB version 2 files, see [MS-MCI].

1.3.2 OAB Version 4

The OAB version 4 file format specifies the structure of three files that are downloaded from the server to the client.

- Full Details file. The Full Details file contains the entire offline address book, including all **Address Book** objects, the list of property types that can be found in the address book, and information about the address book itself, including its name, a unique identity identifier, a version number, and a hash value. For an overview of the Full Details file, see section 1.3.2.1. For the structure of the Full Details file, see section 2.9.

- Differential Patch file. A Differential Patch file can be used to transform a previously downloaded version of the Full Details file to the next version of the Full Details file, which saves the client from downloading the entire Full Details file again. For an overview of the Differential Patch file, see section 1.3.2.3. For the structure of the Differential Patch file, see section 2.10.

- Display Template file. A Display Template file describes how the **Address Book** objects in the OAB can be rendered by the client on a display device to the user, as specified in [MS-OXOABKT]. For an overview of the Display Template file, see section 1.3.2.4. For the structure of the Display Template file used by OAB version 2 and OAB version 4, see section 2.2.

The **Address Book** object data in the Full Details file is not sorted in a predetermined manner, thus it is up to the client to decompress and index the file to enable fast retrieval and searches.

The files stored on the server are in a compressed format, as specified in [MS-PATCH]. All the uncompressed OAB version 4 files contain the same header structure. The OAB version 2 file consists of the following files:

- A 32 bit **little-endian** file version number. The version number used to determine the type of file: Full Details or Display Template.

- A 32 bit little-endian serial number. The serial number is a calculated value in the Full Details file and is used to validate file consistency. It is the Cyclic Redundancy Check (CRC)-32 checksum of the file not including the header structure itself. For more information about CRC-32, see [ISO/IEC 8802-3] section 3.2.8.

- A 32 bit little-endian record count. The record count tells the client how many **Address Book** objects exist in the Full Details file.

1.3.2.1 Uncompressed Full Details File

Apart from the OAB header, the uncompressed Full Details file consists of the following three sections:

- OAB meta-data record

- OAB header record

- One or more **Address Book** object records. Each **Address Book** object record starts with a little-endian 32 bit value that specifies the size of the record in bytes, including the record size field itself.

The OAB metadata record describes the schema of the OAB header record and **Address Book** object records. It starts with a record size value, then two schema tables: one for the OAB header record, and one for the **Address Book** object records. The tables are stored sequentially after each other. The schema tables contain a 32 bit little-endian record count followed by the specified number of 32 bit **property tag** and 32 bit flag value pairs. The flag value is used to tell the client which properties are supposed to be indexed to match the behavior of a client working online.

The first property in the OAB header record and **Address Book** object records is the record size value, followed by a presence bit array, and then the property values. The property values appear in the order provided in the property table in the metadata record. The presence bit array is used to indicate whether the property exists in the OAB header record or **Address Book** object records.

The OAB header record contains information about the address list itself, including the Unicode OAB name, the **ASCII** X500 distinguished name of the OAB, an integer sequence number, and the OAB **GUID** formatted as an ASCII string.

Address Book object records contain at minimum an ASCII SMTP address, an ASCII distinguished name, a Unicode display name, an integer display type, and an integer object type. The number of **Address Book** object records matches the record count contained in the file header.

1.3.2.2 Property Encodings

ASCII strings are encoded as null terminated strings.

Unicode strings are stored as null terminated UTF-8 strings [RFC2204].

Integer values are treated as unsigned and stored in one to five bytes. If the value is less than 0x80, the value is stored as a single byte. If the value is larger than or equal to 0x80, the number of bytes that can minimally hold the value is added to 0x80 and followed by the bytes of the value itself in little-endian format. Values 0x00 through 0x7f are encoded as themselves. Values 0x80 through 0xFF are encoded as 0x81 0xXX. Values 0x0100 through 0xFFFF are encoded as 0x82 0xLSB 0xMSB. Values 0x00010000 through 0x00FFFFFF are encoded as 0x83 0xLSB 0xXX 0xMSB, and values 0x01000000 through 0xFFFFFFFF are encoded as 0x84 0xLSB 0xXX 0xXX 0xMSB.

Boolean values are stored as single bytes: 0x00 for FALSE, and 0x01 for TRUE.

Octet strings are stored using an integer byte length field first (encoded by using the preceding integer encoding rules) followed by the octet stream.

Multi-valued properties are encoded with an integer value count first (encoded by using the preceding integer encoding rules) followed by the specified number of values as encoded by the preceding rules. Multi-valued properties cannot contain empty values.

Null or empty strings are not encoded as single null terminators, but are indicated as not-present using the presence bit array.

Data encoding is specified in more detail in section 2.9.6.

1.3.2.3 Uncompressed Differential Patch File

The Differential Patch file cannot be uncompressed by itself as it requires the original Full Details file. The Differential Patch file describes how to transform an outdated Full Details file into another Full Details file. During transformation, the Differential Patch file is read by the client one block at a time to determine how large a block of the original Full Details file to read, how large the output block will be, and what the compressed patch data is. The patch file starts with a patch header that contains the file format version numbers, a maximum block size value, source and target file sizes, and the source and target file CRC-32 hash codes. The maximum block size value tells the client the maximum size it can expect to be required to read from the original Full Details file, the maximum size it can expect to have to write to the output file, and the size of the largest patch record that will be produced. Following the patch header are a series of patch blocks. The patch block contains the patch size in bytes to be read from the patch file, the size in bytes of the target block that will be produced, the size in bytes of the block to be read from the original Full Details file, and the CRC-32 hash that the resulting output block will have. The start and end of the source and output blocks do not necessarily fall on record boundaries of the source or output files.

1.3.2.4 Uncompressed Display Template File

The Display Template file describes how the **Address Book** object data can be presented to a user, as specified in [MS-OXOABKT].

1.3.2.5 Compressed OAB Details File and Compressed OAB Template file

Uncompressed Details and Display Template files can be very large due to the amount of information stored. In order to reduce the network traffic between the client and the server, these files are transmitted in a compressed form. A compressed file always starts with a **LZX_HDR** structure followed by one or more **LZX_BLK** structures. The **LZX_HDR** structure contains a maximum block size field that is used to tell the client the maximum size of a block it can expect to have to read from the compressed file and the maximum size of a block it can expect to have to write to an output file. It is passed so that the client can pre-allocate buffers before attempting to decompress a file. Also included in the compressed Details or Display Template file is a length field that indicates what the size of the resulting decompressed file will be. It is provided to help the client allocate disk storage and determine whether the resulting output file size is correct.

Each **LZX_BLK** structure contains a flag indicating whether the data field is compressed. If the size of a compressed block is larger than the source data, the server might choose to not compress the block and just pass it verbatim. A CRC-32 hash of the expected decompressed output block is passed to the client to help it determine if the results of decompression are valid.

1.3.2.6 Truncated Properties

Stored on each **Address Book** object record is a **PidTagOfflineAddressBookTruncatedProperties** attribute. This contains the list of property tags that have been truncated or dropped due to size limits. Clients ought to check the property being retrieved from the OAB record against the list of truncated properties for the record. If the property is included in the truncated property list, the value stored in the OAB file is not the same as the address book value that is available online.<1>

Two properties are exempt from truncation: **PidTagEmailAddress** (X500 DN) and **PidTagAddressBookHomeMessageDatabase** [home-message database (MDB)]. These two properties are not limited because they are primary key values that uniquely identify an object.

1.4 Relationship to Protocols and Other Structures

Distributing OABs requires a means of distributing the files to clients by using either public folders or a Web-based distribution method, as described in [MS-OXPFOAB] and [MS-OXWOAB] respectively.

In order to minimize communication costs, the data in the OAB is compressed, as described in [MS-PATCH] and [MS-MCI].

After the data is available to the client, a way of displaying the data is required. The client is free to choose its own method or the server's format can be used, as described in [MS-OXOABKT].

The method of naming properties in the OAB is based on the property tag naming convention, as described in [MS-OXPROPS] section 1.3.3.

1.5 Applicability Statement

The OAB structures are used to download information about the **Address Book** objects for use when working offline or in cached mode.

1.6 Versioning and Localization

None.

1.7 Vendor-Extensible Fields

The OAB version 2 and 4 structures make use of property tags, but OAB version 4 has an extensible schema. New properties can be added to OAB version 4 by a vendor by assigning property tags to Active Directory directory service properties, as described in [MS-ADTS] section 3.1.1.2.3.

2 Structures

All integer fields in the OAB structures are unsigned and use little-endian byte order.

All CRC 32 hash values are calculated using the IEEE 802.3 CRC polynomial of 0xEDB88320 ($x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$) and are seeded with the value 0xFFFFFFFF. For more details, see [ISO/IEC 8802-3].

All structures are packed on single byte boundaries.

All offsets are measured in bytes from the beginning of the specified file.

2.1 X500 Distinguished Name

X500 DNs are used to uniquely identify **Address Book** objects in the OAB. Each **Address Book** object MUST have a unique X500 DN value. The X500 DN is stored in the **PidTagEmailAddress** property, as specified in [MS-OXOABK] section 2.2.3.14. X500 DNs are structured as the following ABNF [RFC4234] definition illustrates:

```
x500-dn    =      org org-unit 0*13(container) object-rdn
                  ; x500-dns are limited to 16 levels

org           =    "/o=" rdn

org-unit      =    "/ou=" rdn

container     =    "/cn=" rdn
```

```
object-rdn   =    "/cn=" rdn

rdn          =    ( non-space-teletex ) /
                  ( non-space-teletex *62(teletex-char)
                    non-space-teletex )
                  ; rdn values are limited to 64 characters
                  ; the number of rdns is limited to 16 but the
                  ; total cumulative length of rdn characters in
                  ; an x500-dn is limited to 256.



teletex-char =    SP / non-space-teletex

non-space-teletex = "!" / DQUOTE / "%" / "&" / "\" / "(" / ")" /
                    "*" / "+" / "," / "-" / "." / "0" / "1" /
                    "2" / "3" / "4" / "5" / "6" / "7" / "8" /
                    "9" / ":" / "<" / "=" / ">" / "?" / "@" /
                    "A" / "B" / "C" / "D" / "E" / "F" / "G" /
                    "H" / "I" / "J" / "K" / "L" / "M" / "N" /
                    "O" / "P" / "Q" / "R" / "S" / "T" / "U" /
                    "V" / "W" / "X" / "Y" / "Z" / "[" / "]" /
                    "_" / "a" / "b" / "c" / "d" / "e" / "f" /
                    "g" / "h" / "i" / "j" / "k" / "l" / "m" /
                    "n" / "o" / "p" / "q" / "r" / "s" / "t" /
                    "u" / "v" / "w" / "x" / "y" / "z" / "|"

addresslist-x500-dn    = "/guid=" 32(HEXDIG) / "/" / x500-dn
```

2.2 Uncompressed OAB Display Template File

The Display Template file is a file that describes to the client how **Address Book** objects and e-mail addresses SHOULD be displayed to the client. The Display Template file is a package that wraps display template and **address creation template** data structures. For more details, see [MS-OXOABKT]. The following ABNF definition shows the format of an uncompressed Display Template file.

```
template-file      =     OAB_HDR mail-user-template
                         distribution-list-template
                         forum-template agent-template
                         organization-template
                         private-distributionlist-template
                         remote-mailuser-template
                         NAMES_STRUCT
                         address-templates data

mail-user-template =     TMPLT_ENTRY
                         ; display template for mailboxes
```

```
distribution-list-template = TMPLT_ENTRY
                        ; display template for distribution lists

forum-template    =     TMPLT_ENTRY
                        ; display template for public folders

agent-template    =     TMPLT_ENTRY
                        ; display template for mail agents

organization-template = TMPLT_ENTRY
                        ; Unused, SHOULD be set to all zeros.

private-distributionlist-template = TMPLT_ENTRY
                        ; Unused, SHOULD be set to all zeros.

remote-mailuser-template = TMPLT_ENTRY
                        ; display template for external email
                        ; addresses

address-templates =     oot-count *(address-creation-template)

oot-count         =     %x00000000-%xFFFFFFFF
                        ; 32 bits of data

address-creation-template     =     TMPLT_ENTRY
                        ; an address creation display template
                        ; The x500 DN MUST end in the value
                        ; /CN=XXXX where XXXX is the mail-type
                        ; eg: SMTP, X400, or MSMAIL


data              =     *(OCTET)
                        ; unstructured data section
```

All the following fields that start with an ‘o’ indicate an offset from the beginning of the file into the unstructured data section.

2.2.1 OAB_HDR

The **OAB_HDR** structure is used to determine the OAB file format version.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
ulVersion																															
ulSerial																															
ulTotRecs																															

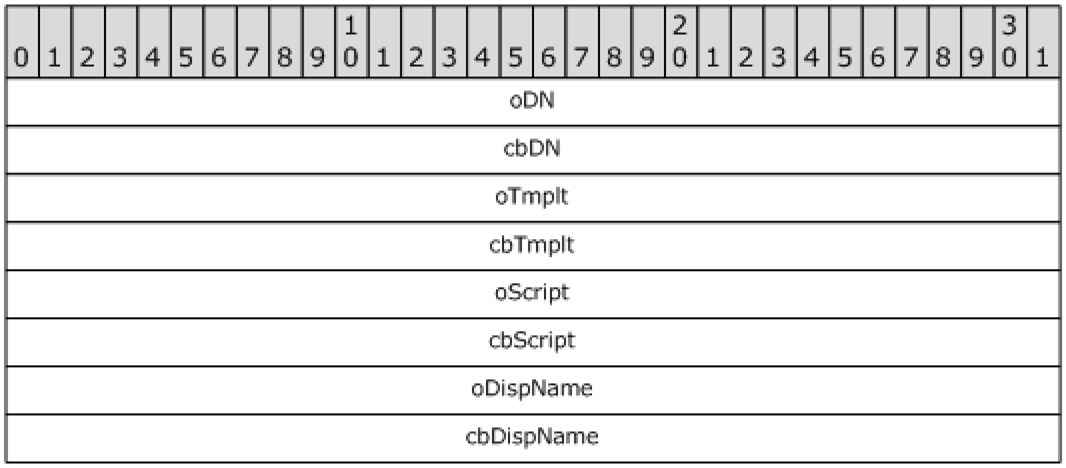
ulVersion (4 bytes): MUST be set to 0x00000007 for uncompressed Display Template files.

ulSerial (4 bytes): Unused, SHOULD be set to 0. Other values MUST be ignored.

ulTotRecs (4 bytes): Unused, SHOULD be set to 0. Other values MUST be ignored.

2.2.2 TMPLT_ENTRY

The **TMPLT_ENTRY** structure is used to encode properties of an individual display template.



oDN (4 bytes): Absolute offset in the Display Template file to the X500 DN of the template.

cbDN (4 bytes): Length of the X500 DN value in bytes including the null terminator.

oTmplt (4 bytes): Absolute offset in the Display Template file to the template structure data. For more details, see [MS-OXOABKT].

cbTmplt (4 bytes): Length of the template structure data in bytes.

oScript (4 bytes): Absolute offset in the Display Template file of the Script file for the template. For more details, see [MS-OXOABKT] section 2.2.2.2.

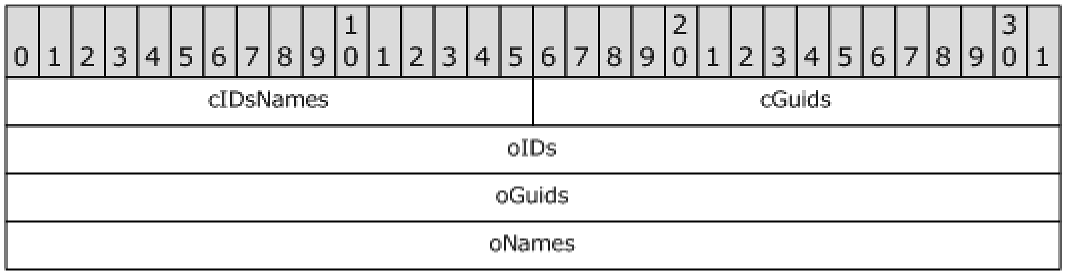
cbScript (4 bytes): Length of the Script file data in bytes.

oDispName (4 bytes): Absolute offset in the Display Template file to the display name for the template. A null terminated ANSI string.

cbDispName (4 bytes): Length of the display name in bytes including null terminator.

2.2.3 NAMES_STRUCT

The **NAMES_STRUCT** structure is used to map GUIDs to and from property tags.



cIDsNames (2 bytes): Count of property IDs and named properties.

cGuids (2 bytes): Count of GUIDs.

oIDs (4 bytes): Absolute offset in the Display Template file to the ID table. Each ID is a 4 byte integer that represents a property tag. For more details, see [MS-OXOABKT].

oGuids (4 bytes): Absolute offset in the Display Template file to the GUID table. Each GUID is stored in binary format in 16 bytes. For more details, see [MS-OXOABKT].

oNames (4 bytes): Absolute offset in the Display Template file to the MAPINAMEID structure table. For more details, see [MS-OXOABKT].

2.3 *Uncompressed OAB Version 2 Browse file*

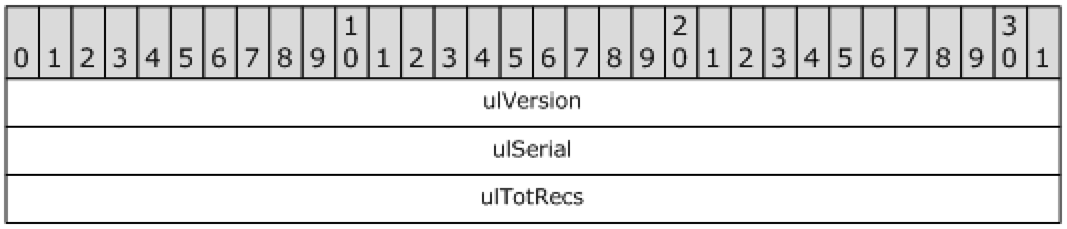
The following ABNF definition shows the format of an uncompressed OAB version 2 Browse file.

```
browse-file       =     OAB_HDR 1*16777213(B2_REC)

display-type      =     DT-MAILUSER / DT-DISTLIST /
                        DT-FORUM / DT-AGENT / DT-ORGANIZATION /
                        DT-REMOTE-MAILUSER
                        ; 8 bit value

DT-MAILUSER       =     %x00
                        ; mailbox display type
```

```
DT-DISTLIST       =     %x01
                        ; distribution list display type

DT-FORUM          =     %x02
                        ; public folder display type

DT-AGENT          =     %x03
                        ; mail agent display type

DT-ORGANIZATION   =     %x04
                        ; department or organization display type

DT-REMOTE-MAILUSER =    %x06
                        ; external e-mail address display type

object-type       =     MAPI-FOLDER / MAPI-MAILUSER /
                        MAPI-DISTLIST
                        ; 8 bit value - high order bit is set to
                        ; 1 if the entry can receive all
                        ; message content, including Rich Text
                        ; Format (RTF) and OLE objects
                        ; see [MS-OXPROPS] section 2.924

MAPI-FOLDER       =     %x03

MAPI-MAILUSER     =     %x06

MAPI-DISTLIST     =     %x08
```

2.3.1 OAB_HDR

The **OAB_HDR** structure is used to determine the OAB file format version and the number of **Address Book** object records in the address list, and it contains a hash value for consistency checks.



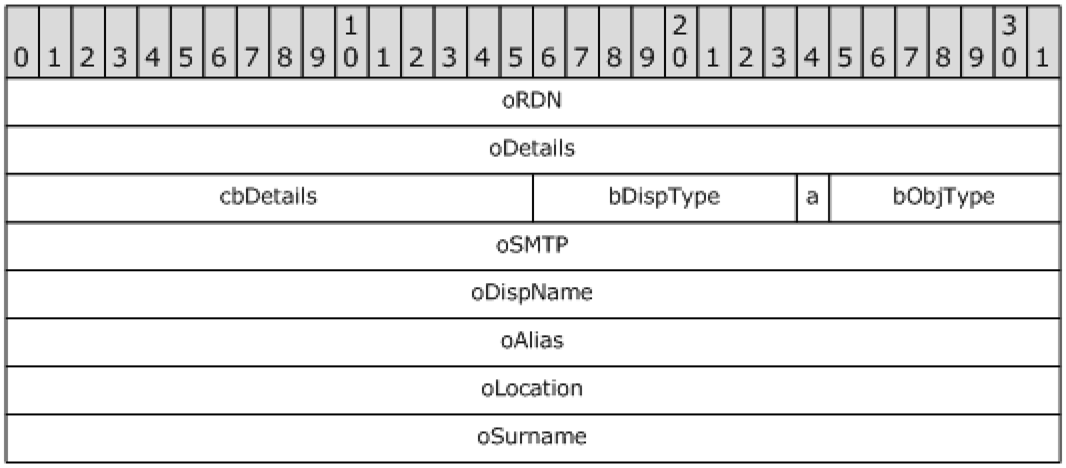
ulVersion (4 bytes): MUST be set to 0x0000000A for uncompressed version 2 OAB Browse files.

ulSerial (4 bytes): A hash of the RDN records for the current set of files.

ulTotRecs (4 bytes): The number of **B2_REC** records stored in the Browse file. MUST be 1 or larger and MUST be less than 16,777,213.

2.3.2 B2_REC

The **B2_REC** structure is used to encode an **Address Book** object in the Browse file. The **Address Book** objects are sorted in the Browse file by alphabetical display name order. The locale that is used by the server to sort the files SHOULD be stored on the **public folder** message that contains the files. The client SHOULD use the stored locale for string comparison when searching the files. For more details, see [MS-OXPFOAB] section 2.2.1.5.



oRDN (4 bytes): Offset of the RDN record in the RDN Index file.

oDetails (4 bytes): Offset of the details record in the Details file.

cbDetails (2 bytes): Size of the details record in the Details file.

bDispType (1 byte): Display type of the **Address Book** object. MUST be set to one of the values in the following table.

Value	Meaning
0x00	DT_MAILUSER
0x01	DT_DISTLIST
0x02	DT_FORUM
0x03	DT_AGENT
0x06	DT_REMOTE_MAILUSER

a (1 bit): SHOULD be set to 1 if the **Address Book** object can receive all message content, including **Rich Text Format (RTF)** and OLE objects. SHOULD be set to 0 if the **Address Book** object cannot receive all message content. For more details, see [MS-OXOABK] section 2.2.3.18.

bObjType (7 bits): Object type of the **Address Book** object. MUST be set to one of the values in the following table.

Value	Meaning
0x03	MAPI-FOLDER
0x06	MAPI-MAILUSER
0x08	MAPI-DISTLIST

oSMTP (4 bytes): Offset of the SMTP address record in the RDN Index file.

oDispName (4 bytes): Offset of the display name record in the ANR Index file.

oAlias (4 bytes): Offset of the alias record in the ANR Index file.

oLocation (4 bytes): Offset of the office location record in the ANR Index file.

oSurname (4 bytes): Offset of the surname record in the ANR Index file.

2.3.3 RDN Hash Computation

The RDN hash value stored in the **OAB_HDR** record of the Browse file is calculated by seeding a 4 byte integer with 0x00000000 and updated by combining the current value with a hash value of the RDN property for each record in the OAB in Browse file order.

The hash value for each RDN value is computed from the RDN value by padding the end of the null terminated string with extra nulls to align it to a 4 byte boundary. Then all the 4 byte blocks are XOR together along with the input seed. Each block is treated as a little-endian integer value. Finally the value is shifted to the left by one bit with the highest order bit being rotated into the lowest order bit.

2.4 Uncompressed OAB Version 2 RDN Index File

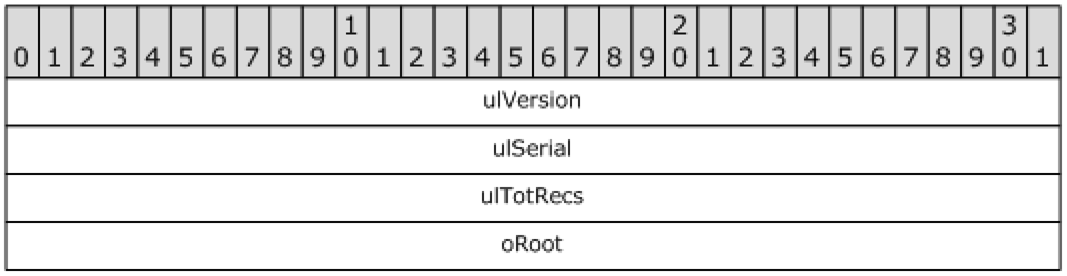
The following ABNF definition illustrates an uncompressed OAB version 2 RDN Index file.

```
rdn-file          =     RDN_HDR 1*pdn-record 1*RDN2_REC
```

```
pdn-record        =      1*(CHAR) %x00
```

2.4.1 RDN_HDR

The **RDN_HDR** structure is used to determine the OAB file format version and the number of RDN records in the RDN Index file, and it contains a hash value for consistency checks.



ulVersion (4 bytes): MUST be set to 0x0000000A for uncompressed version 2 RDN Index files.

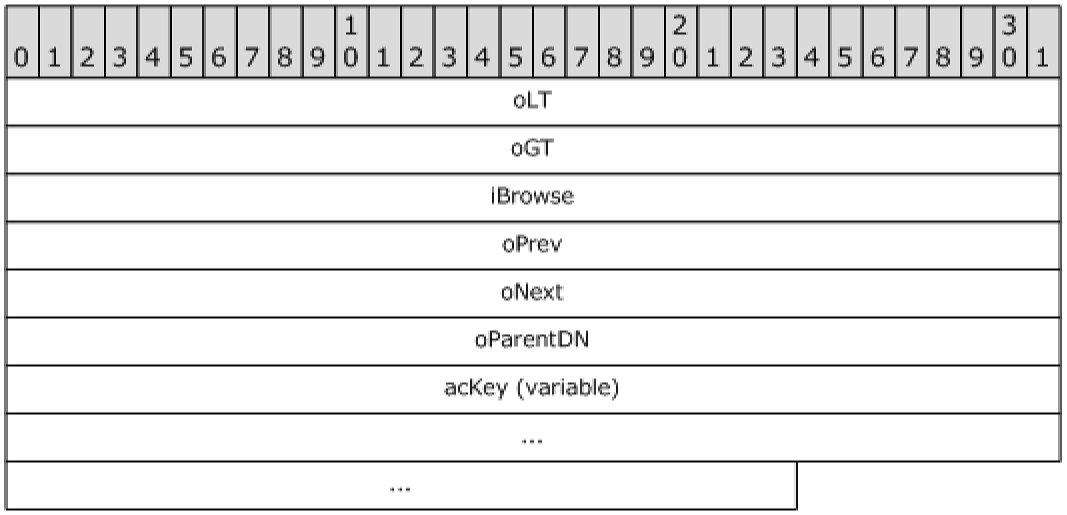
ulSerial (4 bytes): Unused, SHOULD be set to zero and MUST be ignored by the client.

ulTotRecs (4 bytes): The number of **RDN2_REC** records stored in the RDN Index file.

oRoot (4 bytes): The offset of the root **RDN2_REC** node of the RDN index tree. This record MUST be after the last **pdn-record** in the file.

2.4.2 RDN2_REC

Each **RDN2_REC** structure corresponds to a node in the RDN index tree. The tree is constructed as a threaded tree so that searches and moving to the next and previous records are efficient.



oLT (4 bytes): Offset of the left **RDN2_REC** child of the current node in the RDN Index file. The left child MUST sort to the same value as the current node or less. MUST be set to 0x00000000 to indicate that there is no left child node.

oGT (4 bytes): Offset of the right **RDN2_REC** child of the current node in the RDN Index file. The right child MUST sort to the same value as the current node or greater. MUST be set to 0x00000000 to indicate that there is no right child node.

iBrowse (4 bytes): Index to the **B2_REC** in the browse file that references this record. The values 0x00000000 through 0x00000002 are reserved and MUST NOT be used. The index value in the Browse file is computed by using the following equation: iBrowse – 0x00000003.

oPrev (4 bytes): Offset of the previous **RDN2_REC** record in the RDN Index file when sorted as a flat list. MUST be set to 0x00000000 to indicate that this is the first node in the list.

oNext (4 bytes): Offset of the next **RDN2_REC** record in the RDN Index file when sorted as a flat list. MUST be set to 0x00000000 to indicate that this is the last node in the list.

oParentDN (4 bytes): Offset of the null-terminated ANSI **pdn-record** string in the RDN Index file. MUST NOT be set to 0x00000000.

acKey (Variable): The null-terminated ANSI string value of the record. It MUST be 64 characters or fewer including the null terminator.

For RDN records, "/cn=" MUST be removed from the final RDN before storing in the RDN Index file. The **oParentDN** points at the parent X500 DN; therefore, the actual value is computed by prepending the **acKey** value with "/cn=" then appending that result onto the end of the **parent DN** value.

For SMTP records, the SMTP address is split after '@' and the local-part of the SMTP address including the '@' is stored in the **acKey** field. The domain name part of the SMTP address is pointed to by the **oParentDN** offset.

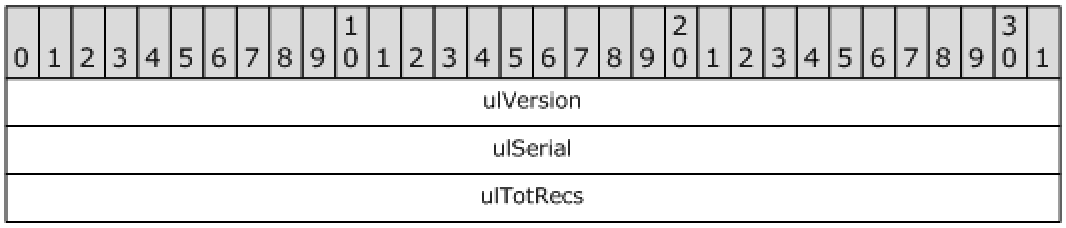
2.5 Uncompressed OAB Version 2 ANR Index File

The following ABNF definition shows the format of an uncompressed OAB version 2 ANR Index file.

```
anr-file          =      OAB_HDR 1*ANR_REC
```

2.5.1 OAB_HDR

The **OAB_HDR** structure is used to determine the OAB file format version and the number of **Address Book** object records in the ANR Index file, and it contains a hash value for consistency checks.



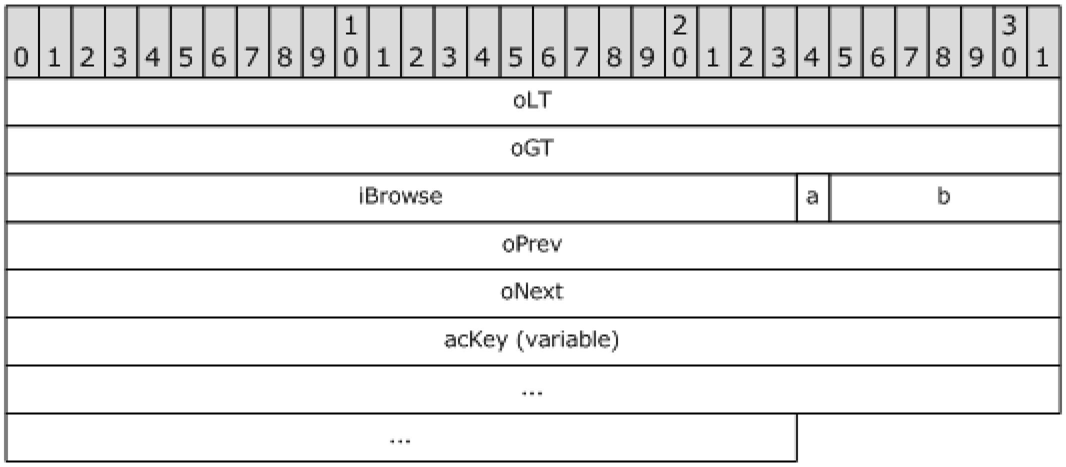
ulVersion (4 bytes): MUST be set to 0x0000000A for uncompressed OAB version 2 ANR Index files.

ulSerial (4 bytes): Unused, SHOULD be set to zero. Other values MUST be ignored.

ulTotRecs (4 bytes): The number of **ANR_REC** records stored in the ANR Index file.

2.5.2 ANR_REC

Each **ANR_REC** structure corresponds to a node in the ANR index tree. The tree is constructed as a threaded tree so that searches are efficient, and traversing to the next and previous records is also efficient. The root of the tree MUST be the first **ANR_REC** in the ANR Index file.



oLT (4 bytes): Offset of the left **ANR_REC** child of the current node in the ANR Index file. The left child MUST sort to the same value as the current node or less. MUST be set to 0x00000000 to indicate that there is no left child node.

oGT (4 bytes): Offset of the right **ANR_REC** child of the current node in the ANR Index file. The right child MUST sort to the same value as the current node or greater. MUST be set to 0x00000000 to indicate that there is no right child node.

iBrowse (3 bytes): Index to the B2_REC in the Browse file that references this record. The values 0x000000 through 0x000002 are reserved and MUST NOT be used. The index value in the browse file is computed by using the following equation: iBrowse – 0x000003.

a (1 bit): MUST be set to 1 for e-mail alias records. MUST be set to 0 for display name, office location, and surname records.

b (7 bits): MUST be all zeros.

oPrev (4 bytes): Offset of the previous **ANR_REC** record in the ANR Index file when sorted as a flat list. MUST be set to 0x00000000 when this is the first node in the list.

oNext (4 bytes): Offset of the next **ANR_REC** record in the ANR Index file when sorted as a flat list. MUST be set to 0x00000000 when this is the last node in the list.

acKey (Variable): The null-terminated ANSI string value of the record. It MUST be 64 characters or fewer including the null terminator.

2.6 Uncompressed OAB Version 2 Details File

The following ABNF definition shows the format of an uncompressed OAB version 2 Details file.

```
v2-details-file   =     OAB_HDR 1*details-record

details-record    =     user-certificate business-telephone
                        given-name initials street-address
                        city-locality state-province postal-code
                        country-region title company-name
                        assistant-name
                        department-name null home-telephone
                        business2-telephone home2-telephone
                        primary-fax mobile-telephone
                        assistant-telephone pager-telephone
                        comment proxy-addresses smime-certs
                        x509-certs

user-certificate  =     binary-value

business-telephone =    string-value

given-name        =     string-value

initials          =     string-value

street-address    =     string-value

city-locality     =     string-value

state-province    =     string-value

postal-code       =     string-value

country-region    =     string-value

title             =     string-value

company-name      =     string-value

assistant-name    =     string-value

department-name   =     string-value

home-telephone    =     string-value

business2-telephone =   string-value

home2-telephone   =     string-value
```

```
primary-fax        =     string-value

mobile-telephone   =     string-value

assistant-telephone =    string-value

pager-telephone    =     string-value

comment            =     string-value

proxy-addresses    =     multivalued-string

smime-certs        =     multivalued-binary

x509-certs         =     multivalued-binary

string-value       =     *(ansi-char) null / null

ansi-char          =     %x01-%xFF
                         ; 8 bits of data

null               =     %x00
                         ; 8 bits of data

multivalued-string =     count 0*255(string-value) / null

count              =     %x00-%xFF
                         ; 8 bits of data

binary-value       =     byte-count 0*65535(OCTET) / null

byte-count         =     %x0000-%xFFFF
                         ; 16 bits of data

multivalued-binary =     count 0*255(binary-value) / null
```

Each Details record MUST fit into 65535 bytes. If a value is not present, a null byte MUST be encoded. All strings MUST be null terminated. Multivalued-binary or multivalued-string encodings with one or more values MUST NOT have any zero length elements.

The details elements map directly to the following property tag table. For details about the following properties, see [MS-OXOABK].

Property tag name	Property tag	Property type	Description

Property tag name	Property tag	Property type	Description
PidTagUserCertificate	0x3A220102	**PtypBinary**	The **user-certificate** property contains an ASN.1 authentication certificate for a messaging user. For more details, see [ISO/IEC 8825-1]. This property is deprecated and SHOULD be set to a null entry.
PidTagBusinessTelephoneNumber	0x3A08001E	**PtypString8**	The **business-telephone** property contains the primary telephone number of the place of business of the **Address Book** object.
PidTagGivenName	0x3A06001E	**PtypString8**	The **given-name** property contains the given name of the **Address Book** object.
PidTagInitials	0x3A0A001E	**PtypString8**	The **initials** property contains the initials for parts of the full name of the **Address Book** object.
PidTagStreetAddress	0x3A29001E	**PtypString8**	The **street-address** property contains the street address of the **Address Book** object.
PidTagLocality	0x3A27001E	**PtypString8**	The **city-locality** property contains the name of the locality of the **Address Book** object, such as the town or city.
PidTagStateOrProvince	0x3A28001E	**PtypString8**	The **state-province** property contains the name of the state or province where the **Address Book** object is located.

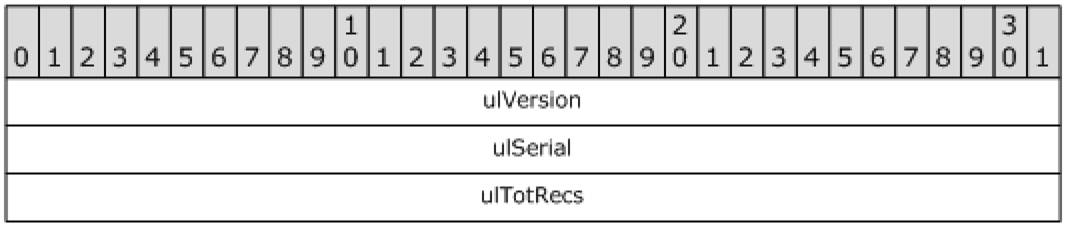
Property tag name	Property tag	Property type	Description
PidTagPostalCode	0x3A2A001E	**PtypString8**	The **postal-code** property contains the postal code of the **Address Book** object.
PidTagCountry	0x3A26001E	**PtypString8**	The **country-region** property contains the name of the country or region where the **Address Book** object is located.
PidTagTitle	0x3A17001E	**PtypString8**	The **title** property contains the job title of the **Address Book** object.
PidTagCompanyName	0x3A16001E	**PtypString8**	The **company-name** property contains the name of the company that employs the **Address Book** object.
PidTagAssistant	0x3A30001E	**PtypString8**	The **assistant-name** property contains the name of the administrative assistant for the **Address Book** object.
PidTagDepartmentName	0x3A18001E	**PtypString8**	The **department-name** property contains the department name in which the **Address Book** object works.
null	0x3A08001E	**PtypString8**	Exchange 2003 and Exchange 2007 duplicate the **PidTagBusinessTelephoneNumber** property in this field. It is not used by Outlook 2003 or Outlook 2007 and MUST be ignored by clients.

Property tag name	Property tag	Property type	Description
PidTagHomeTelephoneNumber	0x3A09001E	**PtypString8**	The **home-telephone** property contains the primary home telephone number for the **Address Book** object.
PidTagBusiness2TelephoneNumber	0x3A1B001E	**PtypString8**	The **business2-telephone** property contains a secondary business telephone for the **Address Book** object.
PidTagHome2TelephoneNumber	0x3A2F001E	**PtypString8**	The **home2-telephone** property contains a secondary home telephone number for the **Address Book** object.
PidTagPrimaryFaxNumber	0x3A23001E	**PtypString8**	The **primary-fax** property contains the telephone number for the fax machine of the **Address Book** object.
PidTagMobileTelephoneNumber	0x3A1C001E	**PtypString8**	The **mobile-telephone** property contains the mobile telephone number of the **Address Book** object.
PidTagAssistantTelephoneNumber	0x3A2E001E	**PtypString8**	The **assistant-telephone** property contains the telephone number for the administrative assistant of the **Address Book** object.
PidTagPagerTelephoneNumber	0x3A21001E	**PtypString8**	The **pager-telephone** property contains the pager telephone number of the **Address Book** object.

Property tag name	Property tag	Property type	Description
PidTagComment	0x3004001E	**PtypString8**	The **comment** property contains a description of the purpose or content of an object.
PidTagAddressBookProxyAddresses	0x800F101E	**PtypMultipleString8**	The **proxy-addresses** property contains a list of e-mail addresses that this **Address Book** object is known by. Each value MUST begin with an e-mail address type followed by a colon character then followed by the address value.
PidTagUserX509Certificate	0x3A701102	**PtypMultipleBinary**	The **smime-certs** property contains SMIME certificates formatted as PKCS-7 encodings. For more details, see [RFC2315].
PidTagAddressBookX509Certificate	0x8C6A1102	**PtypMultipleBinary**	The **x509-certs** property contains ASN.1 [ISO/IEC 8825-1] encoded X.509 certificates. For more details, see [RFC3280].

2.6.1 OAB_HDR

The **OAB_HDR** structure is used to determine the OAB file format version and the number of **Address Book** object records in the address list, and it contains a hash value for consistency checks.



ulVersion (4 bytes): MUST be set to 0x00000007 for uncompressed version 2 Details files.

ulSerial (4 bytes): Unused, SHOULD be set to zero. Other values MUST be ignored.

ulTotRecs (4 bytes): Unused, SHOULD be set to zero. Other values MUST be ignored.

2.7 Uncompressed OAB Version 2 Changes File

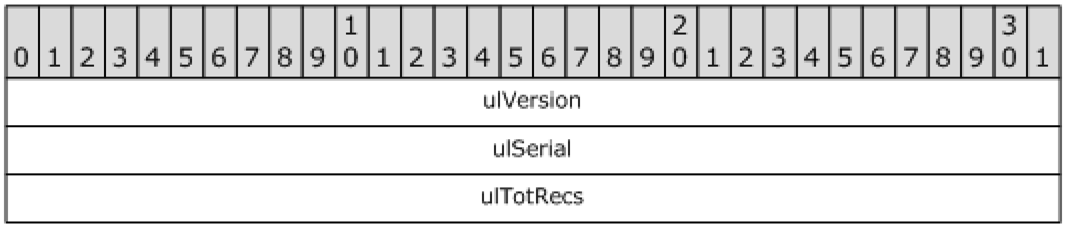
The following ABNF definition shows the format of an uncompressed OAB version 2 Changes file.

```
changes-file      =     OAB_HDR 1*change-record

change-record     =     CHG_REC [display-name parent-dn-offset
rdn]
                        [domain-name-offset local-portion]
                        [alias] [location] [surname]
                        [byte-count 0*65535(OCTET)]
                        [display-type] [object-type]

display-name      =     string-value

parent-dn-offset  =     %x00000000-%xFFFFFFFF
                        ; little endian 32 bit value
                        ; offset of the pdn-record in the
                        ; rdn index file

domain-name-offset =    %x00000000-%xFFFFFFFF
                        ; little endian 32 bit value
                        ; offset of the domain name record in the
                        ; rdn index file

local-portion     =     1*62(ansi-char) '@' null

alias             =     1*63(ansi-char) null

location          =     0*63(ansi-char) null

surname           =     0*63(ansi-char) null
```

```
display-type       =     DT-MAILUSER / DT-DISTLIST /
                         DT-FORUM / DT-AGENT / DT-ORGANIZATION /
                         DT-REMOTE-MAILUSER
                         ; 8 bit value

DT-MAILUSER        =     %x00
                         ; mailbox display type

DT-DISTLIST        =     %x01
                         ; distribution list display type

DT-FORUM           =     %x02
                         ; public folder display type

DT-AGENT           =     %x03
                         ; mail agent display type

DT-ORGANIZATION    =     %x04
                         ; department or organization display type

DT-REMOTE-MAILUSER =     %x06
                         ; external e-mail address display type

object-type        =     MAPI-FOLDER / MAPI-MAILUSER /
                         MAPI-DISTLIST
                         ; 8 bit value - high order bit is set to
                         ; 1 if the entry can receive all
                         ; message content, including Rich Text
                         ; Format (RTF) and OLE objects
                         ; For details, see section 2.786
                         ; in [MS-OXPROPS]

MAPI-FOLDER        =     %x03

MAPI-MAILUSER      =     %x06

MAPI-DISTLIST      =     %x08
```

2.7.1 OAB_HDR

The **OAB_HDR** structure is used to determine the OAB file format version and the number of **Address Book** object records in the address list, and it contains a hash value for consistency checks.



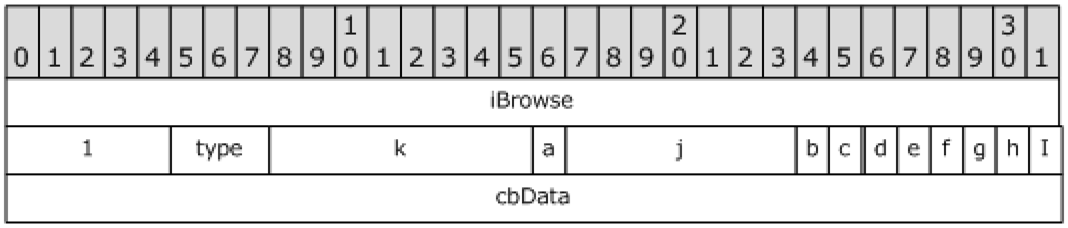
ulVersion (4 bytes): MUST be set to 0x0000000B for uncompressed version 2 Changes files.

ulSerial (4 bytes): MUST be set to the **ulSerial** value of the version 2 OAB Browse file that these changes are to be applied against. The client MUST NOT apply a Changes file to a set of OAB files if the serial number does not match.

ulTotRecs (4 bytes): The count of the **change-record** structures in the Changes file.

2.7.2 CHG_REC

The **CHG_REC** structure is used to tell the client which record to update and what attributes are included in the change record.



iBrowse (4 bytes): The index of the record to be changed. The values 0x00000000 through 0x00000002 are reserved and MUST not be used. The index value in the browse file is computed by using the following equation: iBrowse – 0x00000003.

If the change type is an addition, the **iBrowse** points to the record in the old file that the new record MUST be inserted before. For example, if the record is to be inserted at the beginning of the file, the **iBrowse** value will be 0x00000003. If the record is to be appended at the end of the file, the **iBrowse** will be one plus the maximum **iBrowse** index in the old file. If the change type is a modification, the **iBrowse** points at the record in the old file that MUST be modified. If the change type is a deletion, the **iBrowse** points at the record in the old file that MUST be removed.

l (5 bits): MUST be 00000. Other values MUST be ignored.

type (3 bits): MUST be 000, 001, or 010. A value of 000 indicates a modification record, a value of 001 indicates a record addition, and a value of 010 indicates a record deletion. A value of 010 means that fields **a** through **j** MUST be 0 and that **display-name**, **parent-dn-offset**, and **rdn** MUST be present in the change record. A value of 001 means that fields **a** through **j** MUST be 1. A value of 000 means that fields **a** through **i** are set according to the presence of the data fields in the change record.

k (8 bits): MUST be 0.

j (7 bits): MUST be all 0 for a modification or deletion record. MUST be all 1s for an addition record.

a (1 bit): 1 indicates that the **object-type** field MUST be present in the change-record. 0 indicates that it MUST NOT be present.

b (1 bit): 1 indicates that the **local-portion** field MUST be present in the change-record. The value of this field MUST be the same as field **c**.

c (1 bit): 1 indicates that the **domain-name-offset** field MUST be present in the change-record. 0 indicates that it MUST NOT be present.

d (1 bit): 1 indicates that the **alias** field MUST be present in the change-record. 0 indicates that it MUST NOT be present.

e (1 bit): 1 indicates that the **location** field MUST be present in the change-record. 0 indicates that it MUST NOT be present.

f (1 bit): 1 indicates that the **surname** field MUST be present in the change-record. 0 indicates that it MUST NOT be present.

g (1 bit): 1 indicates that the **details** field MUST be present in the change-record. 0 indicates that it MUST NOT be present.

h (1 bit): 1 indicates that the **details** field MUST be present in the change-record and that it is larger than the old details record in the old Details file. 0 indicates that the size of the `details` field is equal to or smaller than the old record in the Details file. If field **g** is 0 then field **h** MUST be set to 0.

i (1 bit): 1 indicates that the **display-type** field MUST be present in the change-record. 0 indicates that it MUST NOT be present.

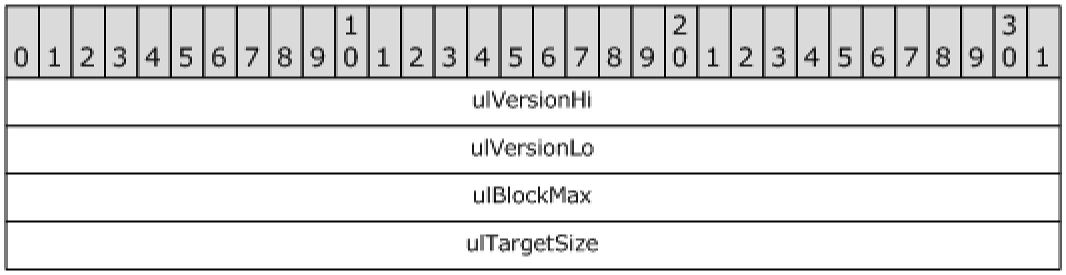
2.8 Compressed OAB Version 2 File

A compressed OAB version 2 file is structured as the following ABNF definition illustrates.

```
v2-compressed-file      =     MDI_HDR 1*MDI_BLK
```

2.8.1 MDI_HDR

The **MDI_HDR** structure contains versioning information to indicate that it is an OAB version 2 compressed file. It contains the target file size value that SHOULD be used by the client to check that the final result is correct.



ulVersionHi (4 bytes): An integer value that MUST be 0x00000002.

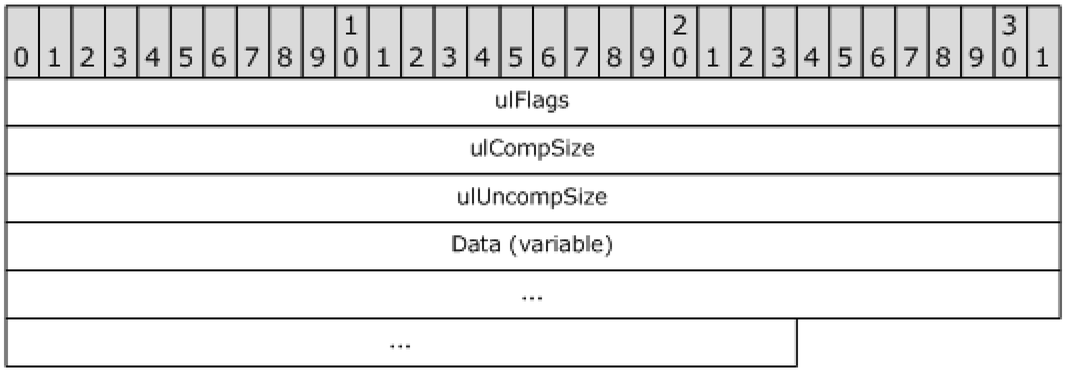
ulVersionLo (4 bytes): An integer value that MUST be 0x00000001.

ulBlockMax (4 bytes): An integer value that indicates in bytes the largest size of a block that will be read from the source compressed input file or written to the target output file. This field is here so that the client can pre-allocate required buffers. MUST be 0x00008000.

ulTargetSize (4 bytes): An integer value that specifies the expected length of the resulting output target file. This value SHOULD be used by the client to ensure that the target output file was generated correctly.

2.8.2 MDI_BLK

The **MDI_BLK** structure is used to split the decompression process into more easily handled smaller sized blocks.



ulFlags (4 bytes): An integer value that indicates whether the data field is compressed. MUST be either 0x00000000 to indicate the data field is not compressed and can be written out directly to the target file, or 0x00000001 to indicate the data field is compressed and ought to be decompressed using MCI decompression first.

ulCompSize (4 bytes): An integer value that specifies the size of the data field in bytes.

ulUncompSize (4 bytes): An integer value that specifies the size in bytes of the output target block to be written to the output file.

data (variable): Either a raw data stream or a compressed byte stream depending on the value of the **ulFlags** field. For more details, see [MS-MCI].

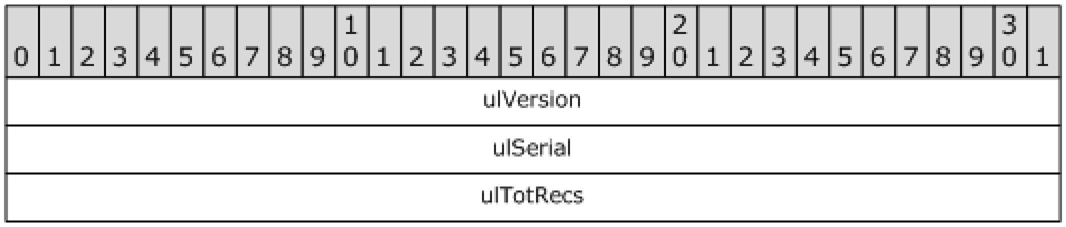
2.9 Uncompressed OAB Version 4 Full Details File

The following ABNF definition shows the format of an uncompressed OAB version 4 Details file.

```
v4-details-file    =    OAB_HDR OAB_META_DATA
                        header-record
                        1*address-book-object-record

header-record   =       OAB_V4_REC

address-book-object-record =  OAB_V4_REC
```

2.9.1 OAB_HDR

The **OAB_HDR** structure is used to determine the OAB file format version and the number of **Address Book** object records in the address list, and it contains a hash value for consistency checks.



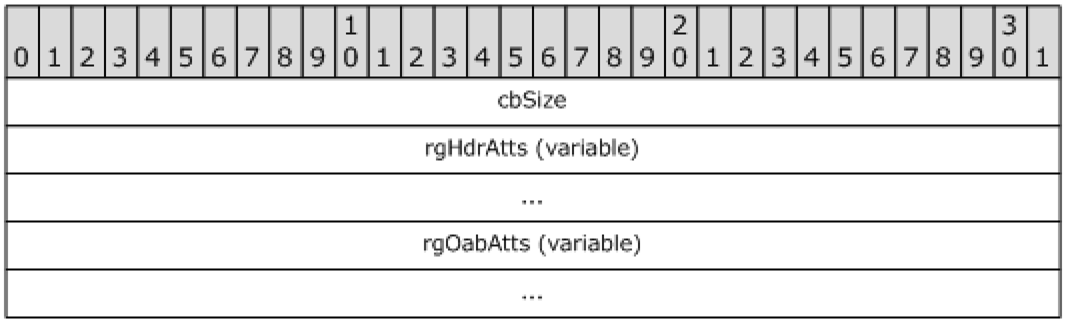
ulVersion (4 bytes): Set to 0x00000020 for uncompressed version 4 OAB Full Details files. Set to 0x00000007 for uncompressed Details Template files.

ulSerial (4 bytes): The CRC-32 hash of the rest of the file not including this header structure. All CRC checksums are calculated with an initial seed of 0xFFFFFFFF and use the IEEE 802.3 [ISO/IEC 8802-3] CRC polynomial of 0xEDB88320.

ulTotRecs (4 bytes): The number of **address-book-object-records** stored in the file.

2.9.2 OAB_META_DATA

The **OAB_META_DATA** structure contains information about the schema of all properties that can be represented in an OAB header or **Address Book** object record.



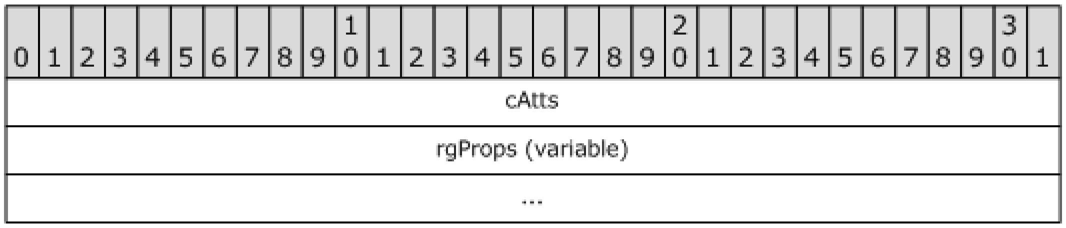
cbSize (4 bytes): The length of the **OAB_META_DATA** structure in bytes. This count includes both the **cbSize** field and the combined length of the **rgHdrAtts** and **rgOabAtts** fields.

rgHdrAtts (Variable): An **OAB_PROP_TABLE** structure that describes the properties that can be present in the **header-record**. MUST contain 0 or more header property records.<2>

rgOabAtts (Variable): An **OAB_PROP_TABLE** structure that describes the properties that can be present in any **address-book-object-record**. MUST contain 0 or more **Address Book** object property records.<3>

2.9.3 OAB_PROP_TABLE

The **OAB_PROP_TABLE** structure represents the property schema of either the OAB header record or all the **Address Book** object records. It contains a list of **OAB_PROP_REC** structures.

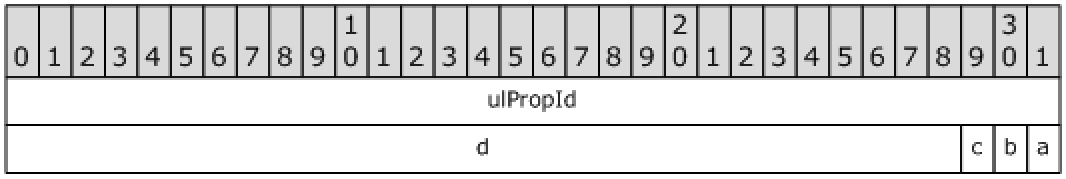


cAtts (4 bytes): An integer that specifies the number of **OAB_PROP_REC** records in **rgProps**.

rgProps (Variable): A list of 0 or more **OAB_PROP_REC** structures.

2.9.4 OAB_PROP_REC

The **OAB_PROP_REC** structure defines a property that can be stored in an OAB header or **Address Book** object record and describes how the attribute is used online.



ulPropId (4 bytes): A property tag. The property type portion of the property tag MUST be one of the following values. For more details about the data types provided in the table, see [MS-OXCDATA] section 2.13.1.

Value	Meaning
0x0003	**PtypInteger32**
0x000B	**PtypBoolean**
0x001E	**PtypString8**
0x001F	**PtypString**
0x0102	**PtypBinary**
0x1003	**PtypMultipleInteger32**
0x101E	**PtypMultipleString8**
0x101F	**PtypMultipleString**
0x1102	**PtypMultipleBinary**

a (1 bit): 1 indicates that the property is part of the ANR property set online. 0 indicates that it is not part of the ANR property set online.

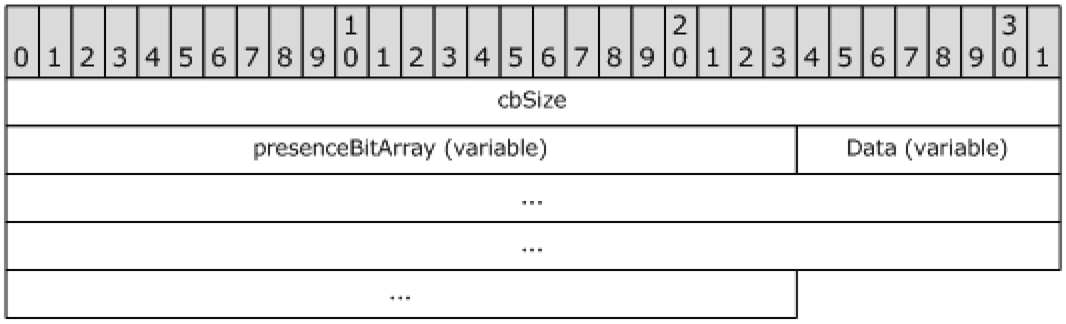
b (1 bit): 1 indicates that the property is a primary key index when used online and a value MUST be present on **every address-book-object-record** in the OAB version 4 Full Details file.

c (1 bit): 1 indicates that the property is indexed separately online. The client MAY choose to index the property locally.

d (29 bits): All bits of **d** MUST be 0 and ignored on receipt.

2.9.5 OAB_V4_REC

The **OAB_V4_REC** structure represents either the OAB header record or an individual **Address Book** object record in an OAB file.



cbSize (4 bytes): The length of the **OAB_V4_REC** structure in bytes. This count includes both the **cbSize** field and the combined length of the **presenceBitArray** and **data** fields.

presenceBitArray (variable): A bit array that indicates whether a property specified in the **OAB_PROP_TABLE** structure is present in the data field. The first element of the bit array is the most significant bit of the first byte. The size of the **presenceBitArray** field in bytes MUST be the value of the **cAtts** field of the appropriate **OAB_PROP_TABLE** structure divided by 8 and rounded up to the nearest integer value. A 0 record in the **presenseBitArray** indicates that the property is not present in the data field. 1 indicates the property is present. The index of the property in the **OAB_PROP_TABLE** structure MUST match the index of the value in the **presenceBitArray**. Unused bits in the final byte MUST be set to 0.

data (variable): The set of property values for the **address-book-object-record** or **header-record**. The format of the **data** field is specified in section 2.9.6.

2.9.6 Data Encoding

Property values are encoded in the data field based on the property type and are packed on byte boundaries. The properties are laid out in the order that the property definition exists in the **OAB_PROP_TABLE** structure. If a property does not exist, the **presenceBitArray** value MUST be 0 and no value is encoded in the data field.

2.9.6.1 PtypInteger32 (0x0003) Value Encoding

All integer values are considered unsigned and MUST fit in the range of a 32 bit integer ($0 – 2^{32}$-1). Integers equal to or less than 127 MUST be encoded as a single byte. Integers 128 or greater are encoded with first a byte count byte with the most significant bit set, then the little-endian value encoding. The byte count, if required, MUST be 0x81, 0x82, 0x83, or 0x84 representing 1, 2, 3, or 4 bytes. The most significant byte of the value representation MUST NOT be 0x00, a lower byte count MUST be used. For example, 0x0000007F MUST be encoded as 0x7F and MUST NOT be encoded as 0x81 0x7F, 0x82 0x7F 0x00, 0x83 0x7F 0x00 0x00, or 0x84 0x7F 0x00 0x00 0x00.

For more details about the **PtypInteger32** data type and the data types specified in the following encoding sections, see [MS-OXCDATA] section 2.13.1.

2.9.6.2 PtypBoolean (0x000B) Value Encoding

All Boolean values are encoded as a single byte. TRUE MUST be encoded as 0x01 and FALSE MUST be encoded as 0x00.

2.9.6.3 PtypString8 (0x001E) Value Encoding

All narrow or multi-byte character set strings are encoded as byte sequences and MUST be terminated by a single 0x00 byte. A string sequence MUST NOT contain a 0x00 byte as part of the string itself. A zero length or empty string MUST NOT be encoded, but MUST be marked as not present in the **presenceBitArray**.

2.9.6.4 PtypString (0x001F) Value Encoding

All Unicode strings are encoded as UTF-8 byte sequences and MUST be terminated by a single 0x00 byte. A string encoding MUST NOT contain a 0x00 byte as part of the string itself. A zero length or empty string MUST NOT be encoded, but MUST be marked as not present in the **presenceBitArray**.

2.9.6.5 PtypBinary (0x0102) Value Encoding

All raw byte sequences are encoded by a length value followed by the specified number of bytes. The length value is encoded as a **PtypInteger32** as shown in section 2.9.6.1. For example, the byte sequence 0x22 0xF8 0xFF 0x00 0x22 would be encoded as 0x05 0x22 0xF8 0xFF 0x00 0x22. A zero length **PtypBinary** value MUST NOT be encoded, but MUST be marked as not present in the **presenceBitArray**.

2.9.6.6 PtypMultipleInteger32 (0x1003) Value Encoding

Multi-valued integer encodings start with an integer count encoding followed by the specified number of integer value encodings. All integer encodings, including the value count, are encoded in the same way that **PtypInteger32** is encoded. All values MUST be unique. Values MAY appear in any order.

2.9.6.7 PtypMultipleString8 (0x101E) Value Encoding

Multi-valued string encodings start with an integer count encoding followed by the specified number of string value encodings. The count encoding is encoded in the same way that **PtypInteger32** is encoded. The individual string encodings are encoded in the same way that **PtypString8** is encoded. Strings MUST be case-insensitive. All values MUST be unique. Values MAY appear in any order. All strings MUST NOT be zero length or empty.

2.9.6.8 PtypMultipleString (0x101F) Value Encoding

Multi-valued Unicode string encodings start with an integer count encoding followed by the specified number of Unicode string value encodings. The count encoding is encoded in the same way that **PtypInteger32** is encoded. The individual string encodings are encoded in the same way that **PtypString** is encoded. Strings MUST be case-insensitive. All values MUST be unique. Values MAY appear in any order. All strings MUST NOT be zero length or empty.

2.9.6.9 PtypMultipleBinary (0x1102) Value Encoding

Multi-valued binary octet encodings start with an integer count encoding, followed by the specified number of binary value encodings. The count encoding is encoded in the same way that **PtypInteger32** is encoded. The individual binary encodings are encoded in the same way that **PtypBinary** is encoded. All values MUST be unique. Values MAY appear in any order. Any binary value MUST NOT be zero length.

2.10 Compressed OAB Version 4 Differential Patch File

The following ABNF definition shows the format of a compressed OAB version 4 Differential Patch file.

```
    patch-file    =   PATCH_HDR 1*PATCH_BLK
```

Patch files are only applied against OAB version 4 Full Details files to produce the next generation of the file.

2.10.1 PATCH_HDR

The **PATCH_HDR** structure contains versioning information to indicate that it is an OAB version 4 patch file. It contains source and target file hash and file size values that SHOULD be used by the client to check that the patch is being applied against the correct file and that the final result is correct.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
ulVersionHi																															
ulVersionLo																															
ulBlockMax																															
ulSourceSize																															
ulTargetSize																															

ulSourceCRC
ulTargetCRC

ulVersionHi (4 bytes): An integer value that MUST be 0x00000003.

ulVersionLo (4 bytes): An integer value that MUST be 0x00000002.

ulBlockMax (4 bytes): An integer value that indicates in bytes the largest size of a block that will be read from the source OAB Details input file, written to the target OAB details output file, or read from the Differential Patch file. This field is here so that the client can pre-allocate required buffers.

ulSourceSize (4 bytes): An integer value that specifies the length in bytes that the source input file is expected to be. This value SHOULD be used by the client to make sure that the correct input file is being read.

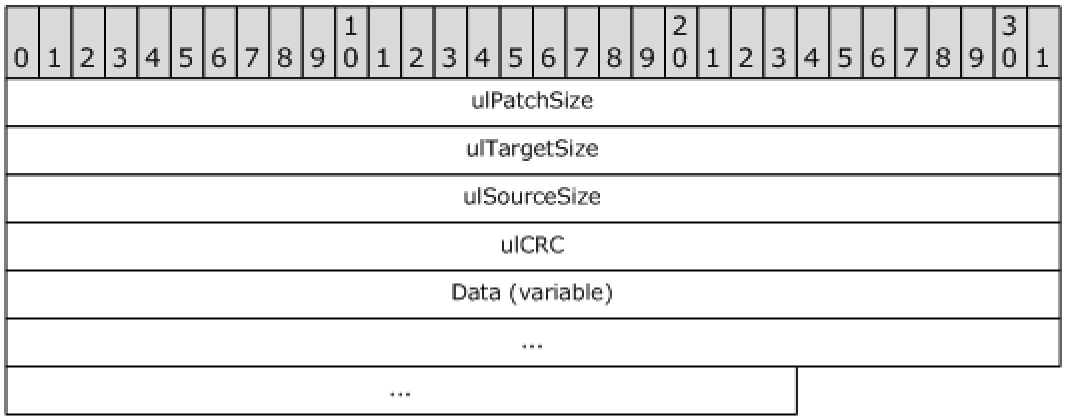
ulTargetSize (4 bytes): An integer value that specifies the length that the resulting output target file is expected to be. This value SHOULD be used by the client to ensure that the target output file was generated correctly.

ulSourceCRC (4 bytes): An integer value that represents the CRC-32 hash of the source input file (excluding the **OAB_HDR** structure). This value SHOULD be used by the client to make sure that the correct input source file is being read.

ulVersionLo (4 bytes): An integer value that represents the CRC-32 hash of the target output file (excluding the **OAB_HDR** structure). This value SHOULD be used by the client to ensure that output target file was generated correctly.

2.10.2 PATCH_BLK

The **PATCH_BLK** structure is used to split the patch process into more easily handled smaller-sized blocks.



0 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 20 1 2 3 4 5 6 7 8 9 30 1
ulPatchSize
ulTargetSize
ulSourceSize
ulCRC
Data (variable)
...
...

ulPatchSize (4 bytes): An integer value that specifies the size of the data field in bytes.

ulTargetSize (4 bytes): An integer value that specifies the size in bytes of the output target block to be written to the output file.

ulSourceSize (4 bytes): An integer value that specifies the size in bytes of the source input block to be read from the source input file and used to generate the output block.

ulCRC (4 bytes): An integer value that specifies the CRC-32 hash of the resulting target block. This value SHOULD be used by the client to make sure that the correct output block has been generated.

data (variable): A byte stream of **LZXD** compressed differences to apply to the source block that results in the target block. For more details, see [MS-PATCH].

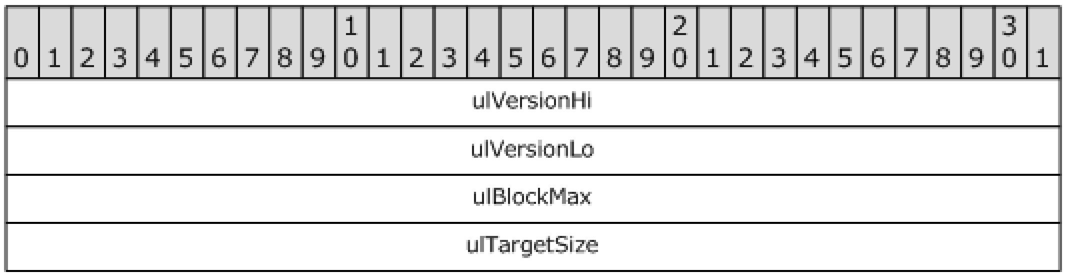
2.11 Compressed OAB Version 4 file

The following ABNF definition shows the format of a compressed OAB version 4 file.

```
v4-compressed-file    =       LZX_HDR 1*LZX_BLK
```

2.11.1 LZX_HDR

The **LZX_HDR** structure contains versioning information to indicate that it is an OAB version 4 compressed file. It contains the target file size value that SHOULD be used by the client to check that the final result is correct.



ulVersionHi (4 bytes): An integer value that MUST be 0x00000003.

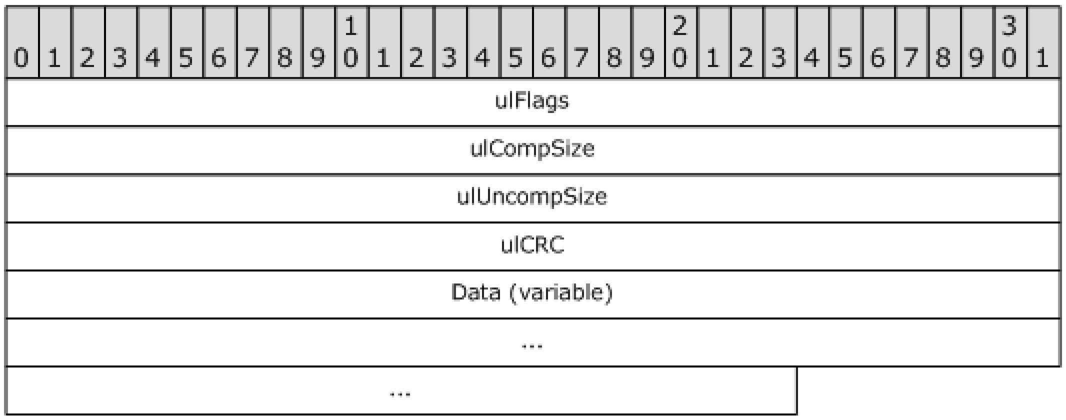
ulVersionLo (4 bytes): An integer value that MUST be 0x00000001.

ulBlockMax (4 bytes): An integer value that indicates in bytes the maximum block size that will be read from the source compressed input file or written to the target output file. This field is provided so that the client can pre-allocate required buffers.

ulTargetSize (4 bytes): An integer value that specifies the expected length of the resulting output target file. This value SHOULD be used by the client to ensure that the target output file was generated correctly.

2.11.2 LZX_BLK

The **LZX_BLK** structure is used to split the decompression process into more easily handled smaller-sized blocks.



ulFlags (4 bytes): An integer value that indicates whether the data field is compressed. MUST be either 0x00000000 to indicate that the data field is not compressed and can be written out directly to the target file, or 0x00000001 to indicate the that data field is compressed and ought to be decompressed using **LZX** decompression first.

ulCompSize (4 bytes): An integer value that specifies the size of the data field in bytes.

ulUncompSize (4 bytes): An integer value that specifies the size in bytes of the output target block to be written to the output file.

ulCRC (4 bytes): An integer value that specifies the CRC-32 hash of the resulting target block. This value SHOULD be used by the client to ensure that the correct output block has been generated.

data (variable): Either a raw data stream or a compressed byte stream, depending on the value of the **ulFlags** field. For more details, see [MS-PATCH].

3 Structure Examples

The examples in this section illustrate the data after it is downloaded to the client and decompressed when they have an OAB installed. The client can use the data in these files to retrieve user information when working offline. The structure of the data in each file is specified in section 2.

3.1 Full OAB Version 2 Offline Address List

The following data show the contents of a sample OAB version 2 Browse file. All data in this section is shown in actual byte order.

```
OAB_HDR
      ulVersion    0a 00 00 00
      ulSerial     bd 32 79 d3
      ulTotRecs    02 00 00 00

B2_REC
      oRDN         d2 00 00 00
      oDetails     0c 00 00 00
      cbDetails    39 00
      bDispType    00
      bObjType     06
      oSmtp        8c 00 00 00
      oDispName    69 00 00 00
      oAlias       2c 00 00 00
      oLocation    00 00 00 00
      oSurname     00 00 00 00

B2_REC
      oRDN         68 00 00 00
      oDetails     45 00 00 00
      cbDetails    35 00
      bDispType    00
      bObjType     06
      oSmtp        b3 00 00 00
      oDispName    0c 00 00 00
      oAlias       8b 00 00 00
      oLocation    00 00 00 00
      oSurname     4e 00 00 00
```

The following data show the contents of a sample OAB version 2 ANR Index file.

```
OAB_HDR
      ulVersion    0a 00 00 00
      ulSerial     00 00 00 00
      ulTotRecs    05 00 00 00


ANR_REC (offset 0x0000000C)
      oLT          2c 00 00 00
      oGT          4e 00 00 00
      iBrowse      04 00 00 00
      oPrev        69 00 00 00
      oNext        8b 00 00 00
      acKey        4c 69 73 61 20 4d 69 6c 6c 65 72 00
                   ; 'Lisa Miller'
```

```
ANR_REC (offset 0x0000002C)
      oLT          00 00 00 00 ; 0 = no left child
      oGT          69 00 00 00
      iBrowse      03 00 00 80 ; high order bit = alias field
      oPrev        00 00 00 00 ; 0 = left-most record
      oNext        69 00 00 00
      acKey        41 64 6d 69 6e 69 73 74 72 61 74 6f 72 00
                   ; 'Administrator'

ANR_REC (offset 0x0000004E)
      oLT          8b 00 00 00
      oGT          00 00 00 00 ; 0 = no right child
      iBrowse      04 00 00 00
      oPrev        8b 00 00 00
      oNext        00 00 00 00 ; 0 = right most record
      acKey        4d 69 6c 6c 65 72 00
                   ; 'Miller'

ANR_REC (offset 0x00000069)
      oLT          00 00 00 00 ; 0 = no left child
      oGT          00 00 00 00 ; 0 = no right child
      iBrowse      03 00 00 00
      oPrev        2c 00 00 00
      oNext        0c 00 00 00
      acKey        41 64 6d 69 6e 69 73 74 72 61 74 6f 72 00
                   ; 'Administrator'

ANR_REC (offset 0x0000008B)
      oLT          00 00 00 00 ; 0 = no left child
      oGT          00 00 00 00 ; 0 = no right child
      iBrowse      04 00 00 80 ; high order bit = alias field
      oPrev        0c 00 00 00
      oNext        4e 00 00 00
      acKey        4c 69 73 61 4d 69 6c 6c 65 72 00
                   ; 'LisaMiller'
```

The following code shows the contents of a sample OAB version 2 RDN Index file.

```
OAB_HDR
      ulVersion    0a 00 00 00
      ulSerial     00 00 00 00
      ulTotRecs    04 00 00 00
      oRoot        68 00 00 00

pdn-record (offset 0x00000010) '/o=example/ou=Exchange
Administrative Group (FYDIBOHF23SPDLT)/cn=Recipients'
                   2f 6f 3d 65 78 61 6d 70 6c 65 2f 6f 75 3d 45 78
                   63 68 61 6e 67 65 20 41 64 6d 69 6e 69 73 74 72
                   61 74 69 76 65 20 47 72 6f 75 70 20 28 46 59 44
```

```
                    49 42 4f 48 46 32 33 53 50 44 4c 54 29 2f 63 6e
                    3d 52 65 63 69 70 69 65 6e 74 73 00
pdn-record (offset 0x0000005C) 'example.com'
                    65 78 61 6d 70 6c 65 2e 63 6f 6d 00

RDN2_REC (offset 0x00000068)
      oLT           8c 00 00 00
      oGT           b3 00 00 00
      iBrowse       04 00 00 00
      oPrev         8c 00 00 00
      oNext         b3 00 00 00
      oParentDN     10 00 00 00
      acKey         4c 69 73 61 20 4d 69 6c 6c 65 72 00
                    ; 'Lisa Miller'

RDN2_REC (offset 0x0000008C)
      oLT           d2 00 00 00
      oGT           00 00 00 00
      iBrowse       03 00 00 00
      oPrev         d2 00 00 00
      oNext         68 00 00 00
      oParentDN     5c 00 00 00
      acKey         41 64 6d 69 6e 69 73 74 72 61 74 6f 72 40 00
                    ; 'Administrator@'

RDN2_REC (offset 0x000000B3)
      oLT           00 00 00 00
      oGT           00 00 00 00
      iBrowse       04 00 00 00
      oPrev         68 00 00 00
      oNext         00 00 00 00
      oParentDN     5c 00 00 00
      acKey         4c 69 73 61 4d 40 00
                    ; 'LisaM@'

RDN2_REC (offset 0x000000d2)
      oLT           00 00 00 00
      oGT           00 00 00 00
      iBrowse       03 00 00 00
      oPrev         00 00 00 00
      oNext         8c 00 00 00
      oParentDN     10 00 00 00
      acKey         41 64 6d 69 6e 69 73 74 72 61 74 6f 72 00
                    ; 'Administrator'
```

The following data show the contents of a sample OAB version 2 Details file.

```
OAB_HDR
      ulVersion     07 00 00 00
```

```
        ulSerial     00 00 00 00
        ulTotRecs    00 00 00 00

  Details-Record (offset 0x0000000C)
        ; empty values for first 22 properties
        00 00 ; empty binary property
        00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 ; empty ANSI properties
        01 ; 1 value for multivalued PidTagAddressBookProxyAddresses
        53 4d 54 50 3a 41 64 6d 69 6e 69 73 74 72 61 74
        6f 72 40 65 78 61 6d 70 6c 65 2e 63
        6f 6d 00
        ; 'SMTP:Administrator@example.com'
        00 ; empty multivalued binary property
        00 ; empty multivalued binary property

  Details-Record (offset 0x00000045)
        00 00; empty binary property
        00 00 00  ; empty ANSI properties
        4c 69 73 61 00 ; 'Lisa' PidTagGivenName
        00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        00 00 00 ; empty ANSI properties
        01 ; 1 value for multivalued PidTagAddressBookProxyAddresses
        01 53 4d 54 50 3a 4c 69 73 61 4d 40 65 78 61 6d
        70 6c 65 2e 63 6f 6d 00
        ; 'SMTP:LisaM@example.com'
        00 ; empty multivalued binary property
        00 ; empty multivalued binary property
```

3.2 Full OAB Version 4 Details File

The following code shows the contents of a sample OAB version 4 Details file. All data in this section are shown in actual byte order.

```
OAB_HDR
      ulVersion          20 00 00 00
      ulSerial           f7 da c0 7f
      ulTotRecs          02 00 00 00

      OAB_META_DATA
            cbSize       5c 00 00 00
            pHdrAtts
              cAtts      04 00 00 00
              rgProps [0]
                  ulPropID    1f 00 00 68
                  ulFlags     00 00 00 00 ; combination of fields a,b,c,d
              rgProps [1]
                  ulPropID    1e 00 04 68
                  ulFlags     00 00 00 00
```

```
          rgProps [2]
              ulPropID     03 00 01 68
              ulFlags      00 00 00 00
          rgProps [3]
              ulPropID     1e 00 02 68
              ulFlags      00 00 00 00
        pOabAtts
          cAtts            06 00 00 00
          rgProps [0]
              ulPropID     1e 00 03 30
              ulFlags      02 00 00 00 ; combination of fields a,b,c,d
          rgProps [1]
              ulPropID     1f 00 fe 39
              ulFlags      02 00 00 00
          rgProps [2]
              ulPropID     1f 00 01 30
              ulFlags      01 00 00 00
          rgProps [3]
              ulPropID     03 00 fe 0f
              ulFlags      00 00 00 00
          rgProps [4]
              ulPropID     03 00 00 39
              ulFlags      00 00 00 00
          rgProps [5]
              ulPropID     03 10 05 68
              ulFlags      00 00 00 00

OAB_V4_REC (Header Properties)
      cbSize               42 00 00 00
      PresenceArray        f0
      Att [0] (Utf8)       5c 47 6c 6f 62 61 6c 20
                           41 64 64 72 65 73 73 20
                           4c 69 73 74 00
      Att [1] (String)     2f 00
      Att [2] (Integer)    06
      Att [3] (String)     64 34 66 32 34 34 61 38
                           2d 61 38 65 63 2d 34 34
                           32 61 2d 38 37 61 33 2d
                           35 32 33 36 66 38 32 63
                           61 62 64 63 00

OAB_V4_REC (Address book object 0)
      cbSize         80 00 00 00
      PresenceArray        f8
      Att [0] (string)     2f 6f 3d 65 78 61 6d 70
                           6c 65 2f 6f 75 3d 45 78
                           63 68 61 6e 67 65 20 41
                           64 6d 69 6e 69 73 74 72
                           61 74 69 76 65 20 47 72
                           6f 75 70 20 28 46 59 44
                           49 42 4f 48 46 32 33 53
```

```
                         50 44 4c 54 29 2f 63 6e
                         3d 52 65 63 69 70 69 65
                         6e 74 73 2f 63 6e 3d 4c
                         69 73 61 20 4d 69 6c 6c
                         65 72 00
      Att [1] (Utf8)     4c 69 73 61 4d 40 65 78
                         61 6d 70 6c 65 2e 63 6f
                         6d 00
      Att [2] (Utf8)     4c 69 73 61 20 4d 69 6c
                         6c 65 72 00
      Att [3] (Integer)  06
      Att [4] (Integer)  00


OAB_V4_REC (Address book object 1)
      cbSize       8c 00 00 00
      PresenceArray      f8
      Att [0] (string)   2f 6f 3d 65 78 61 6d 70
                         6c 65 2f 6f 75 3d 45 78
                         63 68 61 6e 67 65 20 41
                         64 6d 69 6e 69 73 74 72
                         61 74 69 76 65 20 47 72
                         6f 75 70 20 28 46 59 44
                         49 42 4f 48 46 32 33 53
                         50 44 4c 54 29 2f 63 6e
                         3d 52 65 63 69 70 69 65
                         6e 74 73 2f 63 6e 3d 41
                         64 6d 69 6e 69 73 74 72
                         61 74 6f 72 00
      Att [1] (Utf8)     41 64 6d 69 6e 69 73 74
                         72 61 74 6f 72 40 65 78
                         61 6d 70 6c 65 2e 63 6f
                         6d 00
      Att [2] (Utf8)     41 64 6d 69 6e 69 73 74
                         72 61 74 6f 72 00
      Att [3] (Integer)  06
      Att [4] (Integer)  00


Flat OAB header version 32, serial 7FC0DAF7, records 2
------------------------
Header Attributes
Property   Flags
cAtts = 4
0x6800001F: 0      PidTagOfflineAddressBookName
0x6804001E: 0      PidTagOfflineAddressBookDistinguishedName
0x68010003: 0      PidTagOfflineAddressBookSequence
0x6802001E: 0      PidTagOfflineAddressBookContainerGuid
------------------------
OAB Attributes
Property   Flags
```

```
cAtts = 6
0x3003001E: 2      PidTagEmailAddress
0x39FE001F: 2      PidTagSmtpAddress
0x3001001F: 1      PidTagDisplayName
0x0FFE0003: 0      PidTagObjectType
0x39000003: 0      PidTagDisplayType
0x68051003: 0      PidTagOfflineAddressBookTruncatedProperties
------------------------
OAB Meta Data
0x6800001F: \Global Address List
0x6804001E: /
0x68010003: 6
0x6802001E: d4f244a8-a8ec-442a-87a3-5236f82cabdc
------------------------

------------------------
Record 0
------------------------
0x3003001E: /o=example/ou=Exchange Administrative Group
(FYDIBOHF23SPDLT)/cn=Recipients/cn=Lisa Miller
0x39FE001F: LisaM@example.com
0x3001001F: Lisa Miller
0x0FFE0003: 6
0x39000003: 0
------------------------
Record 1
------------------------
0x3003001E: /o=example/ou=Exchange Administrative Group
(FYDIBOHF23SPDLT)/cn=Recipients/cn=Administrator
0x39FE001F: Administrator@example.com
0x3001001F: Administrator
0x0FFE0003: 6
0x39000003: 0
------------------------
```

4 Security Considerations

Data stored in OAB files contain personally identifiable information. Implementers have to ensure that only authorized individuals have access to the data.

5 Appendix A: Office/Exchange Behavior

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

- Office 2003 with Service Pack 3 applied
- Exchange 2003 with Service Pack 2 applied
- Office 2007 with Service Pack 1 applied

- Exchange 2007 with Service Pack 1 applied

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.

<1> Section 1.3.2.6: For string and Unicode attributes, Exchange 2003 SP2 and Exchange 2007 SP1 truncate strings to a size limit. For binary properties, Exchange 2003 SP2 and Exchange 2007 SP1 will drop the entire property if it exceeds the size limit. For multi-valued properties, Exchange 2003 SP2 and Exchange 2007 SP1 will drop individual values for both string and binary properties if the combined size of all the values exceeds a size limit.

The following table defines the default minimum and maximum values of limit settings for **String** and **Binary** data types for files generated by Exchange 2003 SP2 and Exchange 2007 SP1. The minimum limit value is the smallest value that a limit can be set to, not the smallest size an actual value can be. The maximum limit value is the largest value that a size limit can be set to, and does reflect the largest size a property can be.

Data type	Type	Minimum limit value (in bytes)	Maximum limit value (in bytes)
String limit	DWORD	32	3400
Binary limit	DWORD	1024	32768
String multivalued limit	DWORD	512	65536
Binary multivalued limit	DWORD	2048	65536

<2> Section 2.9.2: The rgHdrAtts table MUST have at least the four following attributes for compatibility with Outlook 2007 SP1:

Index Number	Property tag name	Property Tag	Property type	Description
1	**PidTagOabName**	0x6800001F	**PtypString**	Display name of the address list. MAY change between generation versions of the same address list.

2	**PidTagOabDn**	0x6804001E	**PtypString8**	The *addresslist-X500-dn* of the address list container object. MAY change between generation versions of the same address list.
3	**PidTagOabSequence**	0x68010003	**PtypInteger32**	The sequence number of the OAB. MUST increase by one between generation versions of the same address list.
4	**PidTagOabContainerGuid**	0x6802001E	**PtypString8**	A string formatted GUID that represents the address list container object. This value MUST never change between generation versions of the same address list. This value MUST be formatted as "`xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`"

<3> Section 2.9.2: The **rgOabAtts** table MUST have at least the five following attributes for compatibility with Outlook 2007 SP1 and MUST be present on all address book object records:

1. **PidTagEmailAddress** – this MUST be the first entry.
2. **PidTagSmtpAddress** – this MUST be the second entry.
3. **PidTagDisplayName**
4. **PidTagDisplayType**
5. **PidTagObjectType**

The following table describes the default attributes populated on address book object records by Exchange 2007 SP1 in the OAB version 4 Full Details file.

Properties populated in the OAB Version 4 Data file by Exchange 2007 SP1

Index Number	Property tag name	Property Tag	Property type	Description
1	**PidTagEmailAddress**	0x3003 001E	**PtypString8**	Contains the X500 DN.
2	**PidTagSmtpAddress**	0x39fe0 01f	**PtypString**	Contains the SMTP mailing address of the sender.
3	**PidTagDisplayName**	0x3001 001F	**PtypString**	Contains the display name for a given Address Book object.
4	**PidTagEmsAbPhoeneticDisplayName**	0x8C92 001F	**PtypString**	Contains the phonetic display name of an object.
5	**PidTagAccount**	0x3A00 001F	**PtypString**	Contains the account name for the Address Book object.
6	**PidTagSurname**	0x3A11 001F	**PtypString**	Contains the family name of the Address Book object.
7	**PidTagEmsAbPhoneticSurname**	0x8C8F 001F	**PtypString**	Contains the phonetic spelling of the surname.
8	**PidTagGivenName**	0x3A06 001F	**PtypString**	Contains the given name of the Address Book object.

9	**PidTagEmsAbPhoneticGivenName**	0x8C8E001F	**PtypString**	Contains the phonetic given name of the Address Book object.
10	**PidTagEmsAbProxyAddresses**	0x800f101f	**PtypMultipleString**	Contains the e-mail proxy addresses of the Address Book object. For example, SMTP:Laura.Miller@example.com or X400:c=US;a= ;p=example;o=example;s=Miller;g=Laura;.
11	**PidTagOfficeLocation**	0x3A19001F	**PtypString**	Contains the office location of the Address Book object.
12	**PidTagDisplayType**	0x39000003	**PtypInteger32**	Contains a value that is used to associate an icon with a particular row of a table.
13	**PidTagObjectType**	0x0FFE0003	**PtypInteger32**	Contains the type of an object. The object type corresponds to the primary interface that is available for an object that is available through the **OpenEntry** interface.
14	**PidTagSendRichInfo**	0x3A40000B	**PtypBoolean**	Contains **TRUE** if the entry can receive all message content, including RTF and OLE objects.
15	**PidTagBusinessTelephoneNumber**	0x3A08001F	**PtypString**	Contains the primary business telephone for the Address Book object.
16	**PidTagInitials**	0x3A0A001F	**PtypString**	Contains the initials for parts of the full name of the Address Book object.
17	**PidTagStreetAddress**	0x3A29001F	**PtypString**	Contains the street address of the Address Book object.

18	**PidTagLocality**	0x3A27001F	**PtypString**	Contains the name of the locality for the Address Book object, such as the town or city.
19	**PidTagStateOrProvince**	0x3A28001F	**PtypString**	Contains the name of the state or province the Address Book object is located in.
20	**PidTagPostalCode**	0x3A2A001F	**PtypString**	Contains the postal code for the postal address for the Address Book object.
21	**PidTagCountry**	0x3A26001F	**PtypString**	Contains the name of the country or region where the Address Book object is located.
22	**PidTagTitle**	0x3A17001F	**PtypString**	Contains the job title of the Address Book object.
23	**PidTagCompanyName**	0x3A16001F	**PtypString**	Contains the name of the company associated with the Address Book object.
24	**PidTagEmsAbPhoneticCompanyName**	0x8C91001F	**PtypString**	Contains the phonetic spelling of the company name.
25	**PidTagAssistant**	0x3A30001F	**PtypString**	Contains the name of the administrative assistant for the Address Book object.
26	**PidTagDepartment**	0x3A18001F	**PtypString**	Contains a name for the department in which the Address Book object works.
27	**PidTagEmsAbPhoneticDepartmentName**	0x8C90001F	**PtypString**	Contains the phonetic spelling of the department.
28	**PidTagEmsAbTargetAddress**	0x8011001F	**PtypString**	Contains the destination address for this object.

29	**PidTagHomeTelephoneNumber**	0x3A09001F	**PtypString**	Contains the primary home telephone number for the Address Book object.
30	**PidTagBusiness2TelephoneNumber**	0x3A1B101F	**PtypMultipleString**	Contains secondary business telephone numbers for the Address Book object.
31	**PidTagHome2TelephoneNumber**	0x3A2F101F	**PtypMultipleString**	Contains secondary home telephone numbers for the Address Book object.
32	**PidTagPrimaryFaxNumber**	0x3A23001F	**PtypString**	Contains the telephone number of the primary fax machine used by the Address Book object.
33	**PidTagMobileTelephoneNumber**	0x3A1C001F	**PtypString**	Contains the cellular telephone number for the Address Book object.
34	**PidTagAssistantTelephoneNumber**	0x3A2E001F	**PtypString**	Contains the telephone number of the administrative assistant for the Address Book object.
35	**PidTagPagerTelephoneNumber**	0x3A21001F	**PtypString**	Contains the pager telephone number for the Address Book object.
36	**PidTagComment**	0x3004001F	**PtypString**	Contains a comment about the purpose or content of an object.
37	**PidTagUserCertificate**	0x3A220102	**PtypBinary**	Contains an ASN.1 authentication certificate for a messaging user.
38	**PidTagUserX509Certificate**	0x3A701102	**PtypMultipleBinary**	Contains X.509 version 3 security certificates for the Address Book object, as described in [RFC2459].

39	**PidTagEmsAbX509Cert**	0x8C6A1102	**PtypMultipleBinary**	Contains ASN.1 encoded X.509 certificates, as described in [RFC2459].
40	**PidTagEmsAbHomeMdb**	0x8006001e	**PtypString8**	Contains the X500 DN of the **message database (MDB)** for this mailbox. This property value is not subject to truncation.
41	**PidTagEmsAbDisplayNamePrintable**	0x39FF001e	**PtypString8**	Contains the printable string version of the display name.
42	**PidTagEmsAbDisplayType**	0x39050003	**PtypInteger32**	Contains a value used to associate an icon with a particular row of a table.
43	**PidTagOabTruncatedProps**	0x68051003	**PtypMultipleInteger32**	Contains a list of the property tags that have been truncated or limited by the server. If no properties have been removed or limited, the attribute will not be present.

Outlook 2007 SP1 expects the following attributes to be populated in the OAB Version 4 **rgOabAtts** table, and if they are missing, will not try to contact the server to retrieve the values when using the OAB files as a cache:

- **PidTagSmtpAddress**
- **PidTagDisplayName**
- **PidTagAccount**
- **PidTagSurname**
- **PidTagGivenName**
- **PidTagEmsAbProxyAddresses**
- **PidTagOfficeLocation**
- **PidTagDisplayType**
- **PidTagObjectType**
- **PidTagSendRichInfo**
- **PidTagBusinessTelephoneNumber**
- **PidTagInitials**

- **PidTagStreetAddress**
- **PidTagLocality**
- **PidTagStateOrProvince**
- **PidTagPostalCode**
- **PidTagCountry**
- **PidTagTitle**
- **PidTagCompanyName**
- **PidTagAssistant**
- **PidTagDepartment**
- **PidTagEmsAbTargetAddress**
- **PidTagHomeTelephoneNumber**
- **PidTagBusiness2TelephoneNumber**
- **PidTagHome2TelephoneNumber**
- **PidTagPrimaryFaxNumber**
- **PidTagMobileTelephoneNumber**
- **PidTagAssistantTelephoneNumber**
- **PidTagPagerTelephoneNumber**
- **PidTagComment**
- **PidTagUserCertificate**
- **PidTagUserX509Certificate**
- **PidTagEmsAbX509Cert**
- **PidTagEmsAbHomeMdb**
- **PidTagEmsAbDisplayNamePrintable**

Index