

[MS-OXOSMIME]: S/MIME E-mail Object Protocol Specification

Intellectual Property Rights Notice for Protocol Documentation

- **Copyrights.** This protocol documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the protocols, and may distribute portions of it in your implementations of the protocols or your documentation as necessary to properly document the implementation. This permission also applies to any documents that are referenced in the protocol documentation.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the protocols. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, the protocols may be covered by Microsoft's Open Specification Promise (available here: <http://www.microsoft.com/interop/osp/default.mspx>). If you would prefer a written license, or if the protocols are not covered by the OSP, patent licenses are available by contacting protocol@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.

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

Preliminary Documentation. This documentation is preliminary documentation for these protocols. Since the documentation may change between this preliminary version and the final version, there are risks in relying on preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Tools. This protocol documentation is intended for use in conjunction with publicly available standard specifications and networking programming art, and assumes that the reader is either familiar with the aforementioned material or has immediate access to it. A protocol specification does not require the use of Microsoft programming tools or programming environments in order for a Licensee to develop an implementation. Licensees who have access to Microsoft programming tools and environments are free to take advantage of them.

Revision Summary			
Author	Date	Version	Comments
Microsoft Corporation	April 4, 2008	0.1	Initial Availability

Preliminary

Table of Contents

1	<i>Introduction</i>	4
1.1	Glossary	4
1.2	References.....	5
1.2.1	Normative References	5
1.2.2	Informative References	6
1.3	Protocol Overview (Synopsis).....	6
1.4	Relationship to Other Protocols.....	8
1.5	Prerequisites/Preconditions	8
1.6	Applicability Statement.....	8
1.7	Versioning and Capability Negotiation.....	8
1.8	Vendor-Extensible Fields.....	8
1.9	Standards Assignments	8
2	<i>Messages</i>	9
2.1	Transport.....	9
2.2	Message Syntax.....	9
2.2.1	Clear-Signed Message.....	9
2.2.2	Opaque-Signed and Encrypted S/MIME Message.....	10
3	<i>Protocol Details</i>	11
3.1	Common Details.....	11
3.1.1	Abstract Data Model	11
3.1.2	Timers	11
3.1.3	Initialization	11
3.1.4	Higher-Layer Triggered Events.....	11
3.1.5	Message Processing Events and Sequencing Rules	11
3.1.6	Timer Events.....	15
3.1.7	Other Local Events.....	15
4	<i>Protocol Examples</i>	15
5	<i>Security</i>	15
5.1	Security Considerations for Implementers.....	15
5.2	Index of Security Parameters.....	15
6	<i>Appendix A: Office/Exchange Behavior</i>	15
7	<i>Index</i>	17

1 Introduction

When the server receives an Internet e-mail message, it maps the message to an internal format known as the message object schema. Similarly, when the client submits an e-mail message via the server, the server maps the message from its internal format to Internet format for sending. Also, in cases where protocols supported by the server allow saving or reading e-mail messages in Internet format, similar mapping is required to and/or from internal format. For more detailed information about the mapping between internal format and Internet format, see [MS-OXCMAIL].

This document specifies the details of a message's internal format and describes the mapping between internal format and Internet e-mail format for two specific classes of Internet e-mail messages: messages signed or encrypted according to S/MIME standard, and arbitrary clear-signed messages that use the "multipart/signed" MIME format.

1.1 Glossary

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

attachment object
body part
header field
message
message body
message class
message object
message object schema
MIME
MIME entity
MIME message
named property
property
S/MIME

The following terms are specific to this document:

clear-signed message: An Internet e-mail message in the format defined by [RFC1847] and identified with the media type "multipart/signed", or the **message object** representing such a message. One important class of clear-signed message, based on a "multipart/signed" format, is **S/MIME** clear-signed message, as specified in [RFC3851] and [RFC3852].

Content-Disposition header field: A **MIME** header field defined by [RFC2045].

Content-Transfer-Encoding header field: A **MIME** header field defined by [RFC2045].

Content-Type header field: A **MIME** header field defined by [RFC2045].

encrypted S/MIME message: An Internet e-mail message in the format defined by [RFC3851] which uses EnvelopedData CMS content type as defined in [RFC3852], or the message object representing such a message.

header field parameter: A name-value pair that provides additional structured information for a header field, as defined by [RFC2045].

header of the message: The collection of header fields within a message, as defined by [RFC2822].

media type: A value in a **Content-Type Header field**, as defined by [RFC2045].

message signature: The signature defined by [RFC3851].

MIME entity body part: A **body part** defined by [RFC2045].

MIME entity header: A type of header defined by [RFC2045].

opaque-signed S/MIME message: An Internet e-mail message in the format defined by [RFC3851] that uses SignedData CMS content type as defined in [RFC3852], or the message object representing such a message.

RFC2822 message: A message in the format specified by [RFC2822].

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

[MS-OXCMAIL] Microsoft Corporation, "RFC2822 and MIME to E-mail Object Conversion Protocol Specification", April 2008.

[MS-OXCMSG] Microsoft Corporation, "Message and Attachment Object Protocol Specification", April 2008.

[MS-OXGLOS] Microsoft Corporation, "Office Exchange Protocols Master Glossary", April 2008.

[MS-OXOMSG] Microsoft Corporation, "E-mail Object Protocol Specification", April 2008.

[RFC1847] Galvin, J., Murphy, S., Crocker, S., Freed, N., "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, October 1995, <http://www.ietf.org/rfc/rfc1847.txt>.

[RFC2045] Freed, N., et al., "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996, <http://www.ietf.org/rfc/rfc2045.txt>.

[RFC2046] Freed, N. and Borenstein, N., "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996, <http://www.ietf.org/rfc/rfc2046.txt>.

[RFC2047] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, November 1996, <http://www.ietf.org/rfc/rfc2047.txt>.

[RFC2048] Freed, N., Klensin, J., Postel, J., "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", RFC 2048, November 1996, <http://www.ietf.org/rfc/rfc2048.txt>.

[RFC2049] Freed, N., Borenstein N., "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples", RFC 2049, November 1996, <http://www.ietf.org/rfc/rfc2049.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>.

[RFC2822] Resnick, P., Ed., "Internet Message Format", RFC 2822, April 2001, <http://www.ietf.org/rfc/rfc2822.txt>

[RFC3851] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, July 2004, <http://www.ietf.org/rfc/rfc3851.txt>.

1.2.2 Informative References

[RFC3852] Housley, R. "Cryptographic Message Syntax (CMS)", RFC 3852, July 2004, <http://www.ietf.org/rfc/rfc3852.txt>.

1.3 Protocol Overview (Synopsis)

The client and the server process and represent e-mail messages as message objects structured according to message object schema. For an overview of **message object schema**, the **message object**, the **attachment object**, and other details of internal format, see [MS-OXCMMSG].

A conversion between Internet e-mail format and message object schema might be necessary when an incoming message arrives or, in the reverse, when an outgoing message has to be sent in Internet format as specified by [RFC2822], [RFC2045], [RFC2046], [RFC2047], [RFC2048], [RFC2049], [RFC1847], or [RFC3851]. Such conversion maps **MIME entities** to attachment objects or **message body** and maps message header fields and **MIME entity header fields** to properties of the message object or attachment object. See section 1.3 of [MS-OXCMAIL] for an overview of the entire conversion process.

This document describes the special case of such conversion for two specific classes of Internet e-mail messages: arbitrary **clear-signed messages**, and S/MIME opaque-signed and encrypted messages. This document only specifies the special handling necessary for these two classes of messages; for more information about the general conversion process (for example, steps that are not unique to just clear-signed messages and/or opaque-signed and encrypted messages), see [MS-OXCMAIL].

Ordinarily, when an [RFC2822] message or a **MIME message** is mapped to a message object, it is completely deconstructed into a form suitable for direct consumption via a wire protocol and mappable to a typical client's message presentation. This manner of message deconstruction is not feasible for S/MIME messages for following reasons:

1. Encrypted message content and even entire message structure is not accessible without a proper decryption key, which is typically not available at delivery time.
2. Signed message content has to be preserved in its entirety, in exactly the form which was signed, in order for the message signature to be verifiable at a later date.

These two points impose restrictions on how the server and the client map an S/MIME message to a message object; general mapping as described in [MS-OXCMAIL] cannot be used without modifications.

A set of mapping conventions exist to resolve this problem and enable the handling of S/MIME messages as message objects. According to these conventions:

- Unprotected top-level message header fields and MIME entity header fields are mapped to properties of a message object or attachment object in accordance with the general mapping specified in [MS-OXCMAIL].
- The message object is identified as an S/MIME message by having its message class property (**PidTagMessageClass**) set to one of the reserved values specified in section 2.2.1 and section 2.2.2.
- The entire protected content of the S/MIME message is mapped to a single attachment object of a corresponding message object.

The following entities can participate in this protocol:

1. Any server or client that wants to represent S/MIME messages through a message object schema.
2. Any client that wants to send or receive S/MIME messages using a server that implements a message object schema.

The S/MIME E-mail Object protocol is limited to top-level clear-signed or S/MIME wrapping only; a message classified as clear-signed, opaque-signed or encrypted can contain other (nested) S/MIME wrapping layers.

This protocol does not distinguish **opaque-signed S/MIME messages** from encrypted S/MIME messages.

This specification defines the interpretation and rendering of clear-signed or S/MIME opaque-signed and encrypted messages based on the assumption that client or server that wants to interpret or render such messages can parse and interpret the corresponding Internet format defined in elsewhere (e.g. in one or more of [RFC2822], [RFC2045], [RFC2046], [RFC2047], [RFC2048], [RFC2049], [RFC1847], [RFC3851], [RFC3852]).

1.4 Relationship to Other Protocols

This protocol defines a special case of mapping between e-mail messages in Internet formats ([RFC2822], [RFC2045], [RFC2046], [RFC2047], [RFC2048], [RFC2049], [RFC1847], [RFC3851]) and a message object. General mapping is defined in [MS-OXCMAIL].

1.5 Prerequisites/Preconditions

None.

1.6 Applicability Statement

This protocol can be utilized by any server or client that wants to represent S/MIME messages using a message object schema. It can also be utilized by any client that wants to send or receive S/MIME messages using a server that implements a message object schema.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

None.

2.2 Message Syntax

None.

2.2.1 Clear-Signed Message

A clear-signed message in Internet format is a message in which the message MIME entity has the media type “multipart/signed”, as defined in [RFC1847]. Such a MIME entity has two **body parts**: the first part represents signed message content; the second part contains a message signature. For more information about multi-part/signed, see [RFC1847].

A clear-signed message in Internet format is mapped to a message object with the following structure:

1. Message class SHOULD<1><2><3> be set as “IPM.Note.SMIME.MultipartSigned”.
2. Message body SHOULD be set by promoting a primary message body MIME entity to appropriate properties of a message object as defined by [MS-OXCMAIL]. The method of identifying and promoting a message body is the following:
 - a. Consider the first body part of a multipart/signed message MIME entity as a complete Internet message.
 - b. Apply the heuristics specified in [MS-OXCMAIL] to identify a nested MIME entity as a message body and promote its content according to [MS-OXCMAIL].
3. Message object properties other than message class or message body SHOULD be set as specified in [MS-OXCMAIL] and [MS-OXOMSG].
4. The message object MUST contain exactly one attachment object.
 - a. Attachment content, stored in the **PidTagAttachDataBin** property, MUST be set as the entire outer content of a multipart/signed message MIME entity, including a **Content-Type header field** with value “multipart/signed” and any original parameters. All other message entity header fields SHOULD be excluded. It is especially important to preserve the entire original outer content of the first body part within a multipart/signed MIME entity unmodified, as it is protected by a message signature in its original form and any modification will invalidate the message signature. Note that all message header fields that are excluded are normally processed to populate message object properties as specified in [MS-OXCMAIL].
 - b. Other attachment object properties MUST be set as follows:

- i. **PidTagAttachMethod** MUST be set to a value of 0x00000001 (file attachment).
- ii. **PidTagAttachMimeTag** MUST be set to a value of "multipart/signed".
- iii. **PidTagAttachFilename** SHOULD be set to a value of "SMIME.txt".
- iv. **PidTagAttachLongFilename** SHOULD be set to a value of "SMIME.txt".
- v. **PidTagDisplayName** SHOULD be set to a value of "SMIME.txt".
- vi. Other attachment object properties MAY be set as appropriate.

2.2.2 Opaque-Signed and Encrypted S/MIME Message

An opaque-signed or encrypted S/MIME message in Internet format is identified as a MIME message consisting of exactly one MIME entity. The MIME entity usually has the media type "application/pkcs7-mime" or "application/x-pkcs7-mime", but can alternatively have the media type "application/octet-stream" if a file name, specified by Content-Type or Content-Disposition header field parameters, has a file extension ".p7m". The content of the entity body is a Cryptographic Message Syntax (CMS) encapsulation of protected message content, together with all necessary cryptographic metadata. For more information about CMS, see [RFC3852]. For the purposes of this protocol, the content is treated as opaque binary data. Message types defined in [RFC3851] other than opaque-signed or encrypted messages are not supported.

An opaque-signed or encrypted S/MIME message in Internet format is mapped to a message object with the following structure:

1. Message class SHOULD be set as "IPM.Note.SMIME".
2. Message body SHOULD NOT be set. Even for an opaque-signed message, where access to message content is possible without possessing a decryption key, the message body SHOULD NOT be promoted to a message object.
3. Message object properties other than message class or message body SHOULD be set as specified in [MS-OXCMAIL].
4. The message object SHOULD have a named property (with GUID = PS_INTERNET_HEADERS ({00020386-0000-0000-C000-000000000046}) and a string name "Content-Type") which contains the raw ASCII string value of a message MIME entity's Content-Type MIME header field, including any parameters of such header field.
5. The message MUST contain exactly one attachment object.
 - a. Attachment content, stored in the **PidTagAttachDataBin** property, MUST be set as the inner content of a message MIME entity. Any Content-Transfer-Encoding applied to a MIME entity body MUST be removed before storing entity body content in an attachment object.
 - b. Attachment object properties other than content SHOULD be set according to [MS-OXCMAIL], just as they would be if the MIME entity was a normal message attachment. In particular, the

PidTagAttachMimeTag property MUST be set to match the media type of a message MIME entity.

3 Protocol Details

3.1 Common Details

3.1.1 Abstract Data Model

None.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Clear-Signed Message Details

3.1.5.1.1 Recognizing a Clear-Signed Message in Internet Format

The media type of the message MIME entity is the value of the last Content-Type header field. If the message MIME entity's media type is "multipart/signed", the message SHOULD be treated as a clear-signed message. Additional verification steps MAY be performed. For example, a client or server could choose to verify that the multipart/signed MIME entity contains exactly two MIME body parts, as specified in [RFC1847].

3.1.5.1.2 Converting a Clear-Signed Message in Internet Format into a Message Object

To convert a clear-signed message in Internet format into a message object, the following steps SHOULD be performed:

1. From the message MIME entity, promote message header fields to message object properties as specified in [MS-OXCMAIL].
2. Create an attachment object.
3. Set attachment object properties as specified in section 2.2.1.
4. From the message MIME entity, remove all header fields except the Content-Type header field.
5. Save the resulting MIME entity as content of the attachment object created in step 2 (for example, set the value of the **PidTagAttachDataBin** property on the attachment object).

3.1.5.1.3 Recognizing a Message Object that Represents a Clear-Signed Message

If a message object has the message class “IPM.Note.SMIME.MultipartSigned” and contains exactly one attachment object, it SHOULD be treated as a clear-signed message. Additional verification steps MAY be performed to verify that the attachment object is marked with the appropriate media type (e.g. **PidTagAttachMimeTag** property has a value “multipart/signed”) and represents a valid “multipart/signed” MIME entity as specified in [RFC1847]. If message class is not “IPM.Note.SMIME.MultipartSigned” but ends with the suffix “.SMIME.MultipartSigned”, the message object MAY <2><3> be treated as a clear-signed message.

If a message object marked with message class “IPM.Note.SMIME.MultipartSigned” does not have correct structure as specified in section 2.2.1, the behavior is undefined.

3.1.5.1.4 Reconstructing an Internet Format Message from a Clear-Signed Message Object

To reconstruct an Internet format message from a clear-signed message object, the following steps SHOULD be performed:

1. Verify that the message object contains exactly one attachment object.
2. Read the attachment object’s **PidTagAttachDataBin** binary property value and treat it as a MIME entity.
3. From such MIME entity, remove all header fields except the last Content-Type header field.
4. To the MIME entity, add any message header fields resulting from promotion of message object properties, as specified in [MS-OXCMAIL].

The resulting MIME entity is a clear-signed message in its Internet format. A client or server MAY use a different approach as long as it leads to an equivalent result.

3.1.5.1.5 Reading and Interpreting a Message Object that Represents a Clear-Signed Message

For information about how to recognize a message object representing a clear-signed message, see section 3.1.5.1.3.

To read and interpret a clear-signed message, the Internet format SHOULD be reconstructed from a message object as specified in section 3.1.5.1.4. The resulting clear-signed message in its Internet format SHOULD be rendered or interpreted following guidelines specified in [RFC1847] and possibly [RFC3851] or any other similar specification. A client MAY use a different process as long as it leads to the same rendering or interpretation.

3.1.5.1.6 Composing a New Message Object that Represents a Clear-Signed Message

To compose a new message object that represents a clear-signed message, first compose a clear-signed message in its Internet format (as specified by [RFC1847]), and then convert that message to a message object as specified in section 3.1.5.1.2. A client MAY use a different process as long as it leads to the same resulting message object content.

3.1.5.2 Opaque-Signed and Encrypted S/MIME Message Details

3.1.5.2.1 Recognizing an S/MIME Opaque-Signed or Encrypted Message in Internet Format

The media type of the message MIME entity is the value of the last Content-Type header field. If the message MIME entity's media type is "application/pkcs7-mime" or "application/x-pkcs7-mime", the message SHOULD be treated as an opaque-signed or encrypted S/MIME message. Also, if the message MIME entity's media type is "application/octet-stream" and a file extension specified by the "name" parameter of a Content-Type header field or by the "filename" parameter of a Content-Disposition header field ends with ".p7m" (case-insensitive), the message SHOULD be treated as an opaque-signed or encrypted S/MIME message. Additional verification steps MAY be performed. For example, a client or server could choose to verify that MIME entity body content has valid syntax as specified in [RFC3851].

3.1.5.2.2 Converting an S/MIME Opaque-Signed or Encrypted Message in Internet Format into a Message Object

To convert an S/MIME opaque-signed or encrypted message in Internet format into a message object, the following steps SHOULD be performed:

1. From the message MIME entity, promote message header fields to message object properties as specified in [MS-OXCMAIL].
2. Save the raw ASCII string value of the last Content-Type header field, including any parameters of such header, as a message object named property with GUID = PS_INTERNET_HEADERS ({00020386-0000-0000-C000-000000000046}) and name "Content-Type".
3. Promote the message MIME entity as a new attachment object as specified in [MS-OXCMAIL] for a general conversion case.

3.1.5.2.3 Recognizing a Message Object that Represents an S/MIME Opaque-Signed or Encrypted Message

If a message object has the message class "IPM.Note.SMIME" and contains exactly one attachment object, it SHOULD be treated as an S/MIME opaque-signed or encrypted message. Additional verification steps MAY be performed to verify that the attachment object is marked with appropriate media type (e.g. **PidTagAttachMimeTag** is either "application/pkcs7-mime" or "application/x-pkcs7-mime" or it is "application/octet-stream" and filename, as specified by **PidTagAttachFilename**, has a file extension ".p7m") and represents a valid encrypted or opaque-signed message as specified in [RFC3852]. If message class is not "IPM.Note.SMIME" but ends with the suffix ".SMIME", then the message object MAY <4><3> be treated as an S/MIME opaque-signed or encrypted message.

Message class "IPM.Note.SMIME" can be ambiguous (see endnote <6> for more details).

If a message object marked with message class "IPM.Note.SMIME" does not have appropriate structure or content as specified in section 2.2.2, the behavior is undefined.

3.1.5.2.4 Reconstructing an Internet Format Message from an Opaque-Signed or Encrypted S/MIME Message Object

To reconstruct an Internet format message from an opaque-signed or encrypted message object, the following steps SHOULD be performed:

1. Verify that the message object contains exactly one attachment object.
2. Create an empty MIME entity.
3. To the MIME entity, add any message header fields resulting from promotion of message object properties, as specified in [MS-OXCMAIL].
4. To the MIME entity, add the Content-Type header field:
 - a. If the message object has a named property “Content-Type” with GUID PS_INTERNET_HEADERS ({00020386-0000-0000-C000-000000000046}), construct the Content-Type header field using the value of the named property, assuming that such value can contain unparsed MIME parameters.
 - b. Otherwise, construct the Content-Type header field using a media type string obtained from the value of the attachment object’s **PidTagAttachMimeTag** property; add a “name” parameter with a value obtained from **PidTagAttachFileName** property of the attachment object.
5. To the MIME entity, add a Content-Disposition header field with a disposition value “attachment”; add a single “filename” parameter with a value obtained from the **PidTagAttachFileName** property of the attachment object, encoded if necessary as specified in [MS-OXCMAIL].
6. To the MIME entity, add Content-Transfer-Encoding with a value “base64”.
7. Read the attachment object’s **PidTagAttachDataBin** binary property value and encode the result using base64 encoding. Add the result of the encoding as a body of the MIME entity.

The resulting MIME entity is an opaque-signed or encrypted S/MIME message in its Internet format. A client or server MAY use a different approach as long as it leads to an equivalent result.

3.1.5.2.5 Reading and Interpreting a Message Object that Represents an Opaque-Signed or Encrypted S/MIME Message

For information about how to recognize a message object representing an S/MIME opaque-signed or encrypted message, see section 3.1.5.2.3.

To read and interpret an S/MIME message, the Internet Format SHOULD be reconstructed from a message object as specified above in section 3.1.5.2.4. The resulting S/MIME message in its Internet format SHOULD be rendered or interpreted following guidelines specified in [RFC3851]. A client MAY use a different process as long as it leads to the same rendering or interpretation.

3.1.5.2.6 Composing a New Message Object that Represents an Opaque-Signed or Encrypted S/MIME Message

To compose a new message object that represents an S/MIME message, first compose an opaque -signed or encrypted S/MIME message in its Internet format (as specified by [RFC3851]), and then convert that message to a message object as specified in section 3.1.5.2.2. A client MAY use a different process as long as it leads to the same resulting message object structure and content.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

4 Protocol Examples

None.

5 Security

5.1 Security Considerations for Implementers

This protocol does not have any security implications beyond those described in [RFC3851]. Furthermore, this protocol treats S/MIME content as opaque binary data and does not deal with any sensitive material or data such as encryption keys. Although it is best for clients or servers that render, interpret, or compose S/MIME data to do so in a secure fashion, this is beyond the scope of this document.

5.2 Index of Security Parameters

None.

6 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 2.2.1: In some circumstances, Exchange 2003 SP3 sets message class as “IPM.Note.SMIME”. Other clients or servers MUST NOT do this.

<2> Section 2.2.1: Exchange 2003 SP2, Outlook 2003 SP3, and Outlook 2007 SP1 identify any message having a message class suffix of “.SMIME.MultipartSigned” as clear-signed message. In general though, clients or servers which produce a message object representing a clear-signed message SHOULD NOT assume that others will identify a message having message class suffix of “.SMIME.MultipartSigned” as a clear-signed message.

<3> Sections 2.2.1, 2.2.2, 3.1.5.2.3: Exchange 2007 SP1 and Outlook 2007 SP1 recognize Office InfoPath signed or encrypted messages, and for such messages, they use a dynamically determined message class which starts with the prefix “IPM.InfoPathForm.” and ends with the suffix “.SMIME” or “.SMIME.MultipartSigned” (see [MS-OXCMAIL] for details). Exchange 2007 SP1 recognizes such message classes as identifying opaque-signed, encrypted, or clear-signed, despite the fact that, in general, it does not recognize other message classes having suffixes “.SMIME” or “.SMIME.MultipartSigned”.

<4> Sections 2.2.1, 3.1.5.2.3: Exchange 2007 SP1, Outlook 2003 SP3, and Outlook 2007 SP1 set **PidTagAttachFilename** to a value of “SMIME.p7m”.

<5> Section 2.2.2: Exchange 2003 SP2, Outlook 2003 SP3, and Outlook 2007 SP1 set message class to “IPM.Note.Receipt.SMIME” when they identify that an S/MIME message contains a secure receipt, as indicated by “smime-type” parameter with a value “signed-receipt” on the Content-Type header field. Exchange 2003 SP2, Outlook 2003 SP3, and Outlook 2007 SP1 identify any message having a message class suffix of “.SMIME” as an opaque-signed or encrypted message, but other clients/servers SHOULD NOT assume that others will identify a message having message class suffix of “.SMIME” as an opaque-signed or encrypted message.

<6> Section 3.1.5.2.3: In Exchange 2003 SP2 only, if message object has a message class “IPM.Note.SMIME” then such message MAY represent a mislabeled clear-signed message with inner opaque-signed or encrypted content. This means that in Exchange 2003 SP2, the message class “IPM.Note.SMIME” is ambiguous with respect to defining message format. A client or server wishing to interoperate with Exchange 2003 SP2 SHOULD disambiguate the “IPM.Note.SMIME” message object either by analyzing the content of an attachment (i.e. the value of the attachment object property PidTagAttachDataBin) or by inspecting the value of attachment object property PidTagAttachMimeTag. If such value represents a valid multipart/signed MIME entity, then the client or server SHOULD identify the message as a clear-signed message and interpret it according to section 3.1.5.1.

7 Index

Applicability, 8
Common protocol details, 11
Examples, 15
Fields, vendor-extensible, 8
Glossary, 4
Index of security parameters, 15
Informative references, 6
Introduction, 4
Message syntax, 9
Message transport, 9
Messages, 9
 Syntax, 9
 Transport, 9
Normative references, 5
Office/Exchange behavior, 15
Overview, 6
Preconditions, 8
Prerequisites, 8
Protocol details, 11
 Common details, 11
References, 5
 Informative references, 6
 Normative references, 5
Relationship to other protocols, 8
Security, 15
 Index of security parameters, 15
 Security considerations for implementers, 15
Security considerations for implementers, 15
Standards assignments, 8
Vendor-extensible fields, 8
Versioning and capability negotiation, 8