[MS-OXPSVAL]:
E-Mail Postmark Validation Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp) or the Community Promise (available here: http://www.microsoft.com/interop/cp/default.mspx). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Copyright © 2010 Microsoft Corporation.

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability.
06/27/2008	1.0		Initial Release.
08/06/2008	1.01		Updated references to reflect date of initial release.
09/03/2008	1.02		Revised and edited technical content.
12/03/2008	1.03		Revised and edited technical content.
03/04/2009	1.04		Revised and edited technical content.
04/10/2009	2.0		Updated technical content and applicable product releases.
07/15/2009	3.0		Revised and edited technical content.
11/04/2009	3.1.0	Minor	Updated the technical content.
02/10/2010	4.0.0	Major	Updated and revised the technical content.
05/05/2010	5.0.0	Major	Updated and revised the technical content.
08/04/2010	5.1	Minor	Clarified the meaning of the technical content.

Copyright © 2010 Microsoft Corporation.

Contents

Copyright © 2010 Microsoft Corporation.

Copyright © 2010 Microsoft Corporation.

1 Introduction

One of the advantages of e-mail is that it is easy and cheap to send. Unfortunately, this also makes it useful to spammers, as it enables them to send huge amounts of bulk e-mail.

Postmarking is computational "postage" imposed when sending e-mail **messages**. This is a small burden for an individual user, but a large burden for spammers. Spammers rely on being able to send thousands of pieces of mail per hour. To send **spam** with postmarking turned on, they would have to invest a large amount of money to expand their computational power.

This document specifies the E-Mail Postmark Validation protocol, which is responsible for the following:

- The process through which a protocol client can create a message that has the postmark **property**.
- The process through which an application can validate the postmark property in the message to help determine whether it is spam.

1.1 Glossary

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

ASCII
base64 encoding
binary large object (BLOB)
GUID
handle
header
identifier
message
message ID (MID)
messaging object
Multipurpose Internet Mail Extensions (MIME)
non-Unicode
property (3)
recipient (1)
resource
Simple Mail Transfer Protocol (SMTP)
spam
spam filter
Unicode

The following terms are specific to this document:

postmark: A computational proof that is applied to outgoing **messages** to help recipient messaging systems distinguish legitimate e-mail **messages** from junk e-mail **messages**, reducing the chance of false positives.

presolution header: A string that contains the prepended solutions for the **puzzle**.

Pre-Solver: The component that, given specific inputs, generates a **message** **postmark**.

puzzle: The computational problem used in this protocol. The sending client solves the puzzle to demonstrate that the **message** **postmark** is valid.

Copyright © 2010 Microsoft Corporation.

x-header: An extended **Simple Mail Transfer Protocol (SMTP)** mail **message** header.

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

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[FIPS180] Federal Information Processing Standards Publication, "SECURE HASH STANDARD", FIPS PUB 180-1, April 1995, http://www.itl.nist.gov/fipspubs/fip180-1.htm

[MS-DTYP] Microsoft Corporation, "Windows Data Types", March 2007, http://msdn.microsoft.com/en-us/library/cc230273.aspx

[MS-OXCNOTIF] Microsoft Corporation, "Core Notifications Protocol Specification", April 2008.

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

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

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts -- Application and Support", RFC 1123, October 1989, http://www.ietf.org/rfc/rfc1123.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

[RFC2821] Klensin, J., Ed., "Simple Mail Transfer Protocol", RFC 2821, April 2001, http://www.ietf.org/rfc/rfc2821.txt

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

1.2.2 Informative References

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

1.3 Overview

Postmark validation is a computational proof that a messaging client applies to outgoing messages to help **recipient** messaging systems distinguish legitimate e-mail messages from junk e-mail messages. This feature helps reduce the chance of the recipient messaging system incorrectly identifying the message as spam. In the context of **spam filtering**, a false positive exists when a spam filter incorrectly identifies a message from a legitimate sender as spam. When E-mail postmark validation is enabled, the spam filter parses the inbound message for a computational postmark**header**. The presence of a valid, solved computational postmarkheader in the message indicates that the client computer that is sending the message has solved the computational postmark and has included the **puzzle** solution in the message headers.

Copyright © 2010 Microsoft Corporation.

Computers do not require significant processing time to solve individual computational postmarks. However, the processing time required to compute individual postmarks for large numbers of messages is expected to be prohibitive, and therefore will discourage malicious e-mail senders. Individual systems that send millions of spammessages are unlikely to invest the processing power required to solve each computational postmark for each message. For that reason, when a sender's e-mail message contains a valid, solved computational postmark, it is unlikely that the sender is a malicious sender.

1.4 Relationship to Other Protocols

When the e-mail client and recipient server are communicating via the E-mail object protocol, as specified in [MS-OXOMSG], the E-Mail Postmark Validation protocol defines two properties that the client attaches to an e-mail message. Therefore, the E-Mail Postmark Validation protocol relies on the underlying message structures and the handling specified in [MS-OXOMSG].

The Core Notifications protocol, as specified in [MS-OXCNOTIF], provides more information about the properties that are used to send and receive messages.

The Exchange Server Protocols Master property List Specification, as specified in [MS-OXPROPS], provides more information about the data types that are used in this protocol.

1.5 Prerequisites/Preconditions

The E-Mail Postmark Validation protocol assumes that the client has successfully logged on to the server.

1.6 Applicability Statement

This protocol specification defines how e-mail messaging clients can generate and understand computational postmarks. By using this protocol, the client can reduce the number of false positives detected by the recipient server when it tries to identify spam e-mail messages.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

Copyright © 2010 Microsoft Corporation.

2 Messages

2.1 Transport

The transport protocols used by this specification are defined in [MS-OXOMSG].

2.2 Message Syntax

The following sections specify the properties that are specific to the E-Mail Postmark Validation protocol. Before sending these requests to the server, the messaging client MUST be logged on to the server. The protocol client MUST open/acquire **handles** to all **messaging objects** and properties that are set or retrieved.

2.2.1 Input Parameters for Generating the Puzzle

The input parameters specified in the following sections are used to calculate the puzzle.

Note All "String" values, unless otherwise specified, MUST be in **Unicode** format UTF-16 or UCS-2. It is up to the client implementation to choose which format to use; the algorithm treats both formats identically.<1>

2.2.1.1 Number of Recipients

This parameter specifies the total count of **SMTP** message recipients on the "To:" and "Cc: " lines.

This parameter MUST be a decimal value formatted as type "String".

Note Non-SMTP message recipients MUST NOT be counted.

2.2.1.2 Message "To: " and "Cc: " Recipients

This parameter is a string that contains a semicolon separated list of SMTP [RFC2821] addresses that are found on the "To: " and "Cc: " lines.

This parameter MUST be formatted as type "String" and MUST be **base64 encoded**.

Note Addresses on the "Bcc:" lines MUST NOT be used.

Note Accounts that are compatible with [MS-OXOMSG] MUST reference the following properties:

- PidTagEmailAddress
- PidTagAddressType

The recipient string is calculated by means of the following pseudo-logic:

```
For each of the recipients in the [Recipient List] {
    Get the PidTagAddressType and PidTagEmailAddress properties.
    if (PidTagAddressType  == "SMTP") {
        Append PidTagEmailAddress value, followed by a semi-colon,
        to recipient string.
    }
}
Remove the last semi-colon at the end of the recipient string.
```


Copyright © 2010 Microsoft Corporation.

2.2.1.3 Algorithm type

This parameter contains the algorithm type that is used to generate the puzzle.

This parameter MUST be a formatted as type "String".

Note The puzzle-solving system MUST use "sosha1_v1", as it is currently the only valid algorithm type.

2.2.1.4 Degree of Difficulty

This parameter contains the degree of difficulty for which a puzzle solution is sought. A larger Degree of Difficulty value indicates that the puzzle-generating application used more computing **resources** to create the puzzle. Therefore, the receiving system typically assumes that a larger Degree of Difficulty value corresponds to a lower likelihood that the message is spam. This is only a generally accepted guideline, and is not a protocol requirement.

This parameter MUST be a positive integer value that is formatted as type "String".<2>

2.2.1.5 Message Identifier

This parameter contains a unique ID that is represented by a **GUID**.

This parameter MUST be formatted as type "String" and MUST be enclosed in brackets "{}".

2.2.1.6 Message "From: "Address

This parameter contains the sender's SMTP e-mail "From: " address.

This parameter MUST be formatted as type "String" and MUST be base64 encoded.

Note Accounts that are compatible with [MS-OXOMSG] MUST use the PidTagSenderEmailAddress property.

2.2.1.7 Datetime

This parameter contains the creation time of the puzzle.

This parameter MUST consist of **ASCII** characters and MUST be formatted as specified in [RFC1123].

2.2.1.8 Subject Line

This parameter contains the subject of the message, as specified in [RFC2822].

This parameter MUST be formatted as type "String" and MUST be base64 encoded.

Note Accounts that are compatible with [MS-OXOMSG] MUST reference the PidTagSubject property.

2.2.2 Pre-Solver Output values

The **Pre-Solver** will return two values, which are then stored in the message header as **x-header** properties.

Copyright © 2010 Microsoft Corporation.

2.2.2.1 "X-CR-PuzzleID" X-Header Property

The value of the "**X-CR-PuzzleID**" x-header property MUST be the same value as the **message ID** specified in section 2.2.1.5.

The "**X-CR-PuzzleID**" x-header property MUST be formatted as type "String".

2.2.2.2 "X-CR-HashedPuzzle" X-Header Property

The value of the "**X-CR-HashedPuzzle**" x-header property contains the puzzle solution as defined in section 3.1.4.1.1.

The "**X-CR-PuzzleID**" x-header property MUST be formatted as type "String".

Copyright © 2010 Microsoft Corporation.

3 Protocol Details

3.1 Client 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

3.1.4.1 Submit Message Event

3.1.4.1.1 Generating X-CR-HashedPuzzle

The puzzle P takes the following parameters as input (see section 2.2.1):

- Number of recipients *r*.
- E-mail addresses of the recipients *t*.
- Algorithm type *a*.
- A 'degree of difficulty' *n*.
- A message id *m*.
- An e-mail 'From: address' *f*.
- A datetime *d*.
- A subject line *s*.

From these parameters, a document D is formed by concatenating all the parameters together, separating each field with ';'. The constructed document D is represented in a **non-Unicode** string.

Given the sequence of bytes comprising a document D, the computational task involved in the puzzle is to find and exhibit a set of sixteen documents δ such that both of the following are true:

- When each δ is prepended to the hash under the Son-of-SHA-1 hash algorithm H (see section 3.1.4.2) of D with its whitespace removed and then hashed again to form H(δ o H(NWS(D))), the result is zero in at least the first n bits (taken most significant bit first within each byte taken in order). Here, NWS is the function that takes a sequence of bytes as input, removes all those that are legal characters that could match the FWS production specified in [RFC2822], and produces the remaining as output.
- The last 12 bits of each H(δ o H(NWS(D)) are the same (the particular 12-bit suffix value shared by these documents does not matter).

Copyright © 2010 Microsoft Corporation.

Note In the previous two computations, the o operator denotes string concatenation.

That is, the answer to the puzzle P(*t*, *n*, *m*, *f*, *d*, *s*) is a set of 16 documents δ each with these characteristics. The hash H(NWS(D)) is used as the suffix to which each δ is prepended rather than simply D in order to minimize the effect of variation in the length of D on the length of time required to solve the puzzle. Whitespace is stripped from D before being input to the hash in order to minimize sensitivity to the encoding of D in header fields where it can be subjected to folding.

No means other than brute force is known to locate satisfactory δ; however, that a given set of δ indeed answers the puzzle can be quickly verified. The particular brute force approach of first trying all one-byte solutions, then trying all two-byte solutions, then all three-byte solutions, and so on, is as good a solution algorithm as any other, but has the additional benefit that the solutions found will be as small as possible. Furthermore, for puzzles that have reasonable degrees of difficulty, solutions with four or fewer bytes will be typical.

Specifically, the following pseudo code describes the brute force algorithm:

```
Solution = 0;
While(true){
    Hash = H(concatenate(Solution, H(NWS(D))))
    If Verify(Solution, Puzzle) succeeds {
        Remember this solution and Hash
        If we have 16 solutions whose last 12 bits of their
        corresponding Hash are the same {
            Return these 16 solutions
        }
    }
Solution ++
}
```

After the solutions for puzzle P are found, a **presolution header** is generated. The presolution header MUST be the concatenation of the solutions string and the document D separated by a semicolon. The solutions string MUST be a "String" formed by base64 encoding each of the 16 puzzle solutions and concatenating them together, with a '" (space) delimiter.

The value of **X-CR-HashedPuzzle** MUST be set to the presolution header. See section 4 for examples.

3.1.4.2 Son-Of-SHA-1 Hash Algorithm

The Son-of-SHA-1 algorithm is defined as a constrained perturbation of the [FIPS180] algorithm. The intent of defining a new hash algorithm that is unique to the proposed use of computational puzzles for spam reduction is to reduce the ease with which hardware accelerators can be applied to reduce the cost and duration of puzzle solving. In conformant systems, the Son-of-SHA-1 algorithm MUST NOT be implemented in hardware.

In "§5 Functions Used", as specified in [FIPS180], a set of eighty functions are defined that are subsequently used in the core of the algorithm specified in §7 and §8. Each f_t, 0 <= *t* <= 79, operates on three 32-bit words B, C, D and produces a 32-bit word as output.

The Son-Of-SHA-1 algorithm differs from [FIPS180] only in the specification of these functions. Specifically, where [FIPS180] specifies the eighty functions as follows:

f_t(B,C,D) = (B AND C) OR ((NOT B) AND D) (0 <= *t* <= 19)

f_t(B,C,D) = B XOR C XOR D (20 <= *t* <= 39)

Copyright © 2010 Microsoft Corporation.

f_t(B,C,D) = (B AND C) OR (B AND D) OR (C AND D) (40 <= t <= 59)

f_t(B,C,D) = B XOR C XOR D (60 <= t <= 79)

The Son-of-SHA-1 algorithm instead specifies the first of these functions as involving an additional **XOR** operation:

f_t(B,C,D) = g(B,C,D) XOR ((B AND C) OR ((NOT B) AND D)) (0 <= t <= 19)

f_t(B,C,D) = B XOR C XOR D (20 <= t <= 39)

f_t(B,C,D) = (B AND C) OR (B AND D) OR (C AND D) (40 <= t <= 59)

f_t(B,C,D) = B XOR C XOR D (60 <= t <= 79)

The supporting function g(B,C,D) is defined as follows:

g_t(B,C,D) = n(r(m(B,C), m(C,D)))

The binary function m() takes two 32-bit words as input and produces a non-negative 64-bit integer as output by concatenating the two 32-bits words together with the first word, forming the high-order bits of the following result:

m(B,C) = (B << 32) OR C

The unary function n() takes a single 64-bit integer as input and returns the word consisting of the following lower 32 bits:

n(x) = x AND FFFFFFFF

Finally, the binary function r() takes two 64-bit integers as input and computes the 64-bit integer that is the remainder of the first when divided by the second (unless the latter is zero). Specifically, r(x, y) is defined by the following relations:

If $y \neq 0$: $x = k\,y + r(x, y)$ for some non-negative integer k, where $0 <= r(x, y) < y$

If $y = 0$: $x = r(x, y)$

Other than the introduction of function g(), another difference between Son-Of-SHA-1 and [FIPS180] is that in [FIPS180], the following are the constants that are used:

K = 5A827999 (0 <= t <= 19)

K_t = 6ED9EBA1 (20 <= t <= 39)

K_t = 8F1BBCDC (40 <= t <= 59)

K_t = CA62C1D6 (60 <= t <= 79).

In Son-Of-SHA-1, the constants are instead the following:

K = 041D0411 (0 <= t <= 19)

K_t = 416C6578 (20 <= t <= 39)

K_t = A116F5B6 (40 <= t <= 59)

K_t = 404B2429 (60 <= t <= 79).

Copyright © 2010 Microsoft Corporation.

In all other ways, the Son-of-SHA-1 algorithm is identical to [FIPS180].

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 On Message Delivery

3.1.5.1.1 Determining When to Validate

The presence of the custom SMTP header **X-CR-HashedPuzzle** indicates that the message is a presolved message.

The receiving client SHOULD verify that the parameters, as expressed in the puzzle, match the fields of the e-mail message as specified in section 2, in order to prevent spammers from reusing the same presolved message **binary large object (BLOB)** for multiple recipients, thereby allowing them to get away with doing less computation.

The actual difficulty of computing a presolution can be expressed as the difficulty indicated by n, multiplied by the number of To: and Cc: recipients in the presolved message indicated by r (in other words, the number of Recipient tags in the presolution data).

3.1.5.1.2 Validating the Puzzle

The process of validating the puzzle is performed on the receiving end of the communication. The server-side Mail Transport Authority (MTA) SHOULD validate the puzzle. Also, e-mail clients SHOULD validate the puzzle.

The validating process is divided into the following two steps:

1. Validate the puzzle part inside the presolution, making sure that the puzzle is generated for the received e-mail message. An e-mail message passes this validation if all the following tests pass:
    1. Extract recipient part information from the puzzle string (r & t).
        1. The recipient part SHOULD be a subset of the **MIME** recipients extracted from the MIME header of the e-mail message.
        2. The recipient part SHOULD contain the recipient's SMTP address.
            1. If the algorithm is being run on an e-mail client, the client will have a list of e-mail accounts, recipient catalog. At least one e-mail address from the recipient catalog MUST be in recipient part.
            2. If the algorithm is being run on an e-mail server, the protocol server will have a list of e-mail addresses, and received recipients from the RCPT TO command as part of the SMTP [RFC2821] process. The received recipients MUST be a subset of recipient part.
    2. Extract the message identifier from the puzzle string m. The **identifier** MUST match the puzzle ID extracted from the **X-CR-PuzzleID** header.
    3. Extract the sender part from the puzzle string f. The sender's e-mail address MUST match the FROM address in the MIME header of the e-mail message.
    4. Extract the subject line from the puzzle string s. The subject line MUST match the subject extracted from the MIME header of the e-mail message.

Copyright © 2010 Microsoft Corporation.

2. Validate the solution part inside the presolution. The solution for the puzzle MUST meet the difficulty level n.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

The server SHOULD validate postmarks after the e-mail message arrives at the server. The content specified in section 3.1.5.1 is symmetrical on both the client and the server when an e-mail message is received.

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

None.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

Copyright © 2010 Microsoft Corporation.

4 Protocol Examples

4.1 Example 1

Input	Parameter	Value	Base64 encoded
Input	Number of recipients	1	
	recipient list	"user1@example.com"	dQBzAGUAcgAxAEAAZQB4AGEAbQBwAGwAZQAuAGMAbwBtAA==
	Algorithm type	"sosha1_v1"	
	Degree of difficulty	7	
	message IDentifier	"{d04b23f4-b443-453a-abc6-3d08b5a9a334}"	
	From address	"sender@example.com"	cwBlAG4AZABlAHIAQABlAHgAYQBtAHAAbABlAC4AYwBvAG0A
	DateTime	"Tue, 01 Jan 2008 08:00:00 GMT"	
	Subject	"Hello"	SABlAGwAbABvAA==
Result	"X-CR-HashedPuzzle: BjHi CbbP CsE4 DoWO EhAv FJE7 FMx3 FOJO FjsQ HDPJ IFAE IRyJ I5E3 I+BV KBb7 L+gd;1;dQBzAGUAcgAxAEAAZQB4AGEAbQBwAGwAZQAuAGMAbwBtAA==;Sosha1_v1;7;{d04b23f4-b443-453a-abc6-3d08b5a9a334};cwBlAG4AZABlAHIAQABlAHgAYQBtAHAAbABlAC4AYwBvAG0A;Tue, 01 Jan 2008 08:00:00 GMT;SABlAGwAbABvAA==X-CR-PuzzleID: {d04b23f4-b443-453a-abc6-3d08b5a9a334}"		

4.2 Example 2

Input	Parameter	Value	Base64 encoded
Input	Number of recipients	2	
	recipient list	"user1@example.com; user2@example.com"	dQBzAGUAcgAxAEAAZQB4AGEAbQBwAGwAZQAuAGMAbwBtADsAdQBzAGUAcgAyAEAAZQB4AGEAbQBwAGwAZQAuAGMAbwBtAA==
	Algorithm type	"sosha1_v1"	

Copyright © 2010 Microsoft Corporation.

Input	Parameter	Value	Base64 encoded
	Degree of difficulty	7	
	messageIDentifier	"{d04b23f4-b443-453a-abc6-3d08b5a9a334}"	
	From address	"sender@example.com"	cwBlAG4AZABlAHIAQABlAHgAYQBtAHAAbABlAC4AYwBvAG0A
	DateTime	"Tue, 01 Jan 2008 08:00:00 GMT"	
	Subject	"Hello"	SABlAGwAbABvAA==
Result	"X-CR-HashedPuzzle: AejA Arsz Bwjf DuSf Een1 Et0s FrxA GmCG HaiQ It8u Jpqj QdZB R6vS SDZh SrAv UANK;2;dQBzAGUAcgAxAEAAZQB4AGEAbQBwAGwAZQAuAGMAbwBtADsAdQBzAGUAcgAyAEAAZQB4AGEAbQBwAGwAZQAuAGMAbwBtAA==;Sosha1_v1;7;{d04b23f4-b443-453a-abc6-3d08b5a9a334};cwBlAG4AZABlAHIAQABlAHgAYQBtAHAAbABlAC4AYwBvAG0A;Tue, 01 Jan 2008 08:00:00 GMT;SABlAGwAbABvAA==X-CR-PuzzleID: {d04b23f4-b443-453a-abc6-3d08b5a9a334}"	

4.3 Example 3

The following table provides four examples of hash values that result from using the [FIPS180] Son-of-SHA-1 algorithm on the indicated input values.

Input	Son-of-SHA-1 hash value
The string "abc"	FA12E295 9DB79C97 25338C0F D4DE3E01 78C286BD
The string "abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq"	48F6CE9F DCF53F40 89200091 ED9739E1 7D73D975
A string consisting of 1,000,000 "a" characters	57338A4C C33E70D4 3A3D3AD7 E93C85ED E6996CCD
And empty string	7A790886 F5044A7B DA812BA8 BFC286C4 F51E7B34

Copyright © 2010 Microsoft Corporation.

5 Security

5.1 Security Considerations for Implementers

There are no special security considerations specific to the E-Mail Postmark Validation protocol. General security considerations that pertain to the underlying E-mail Object protocol, as specified in [MS-OXOMSG], apply.

5.2 Index of Security Parameters

None.

Copyright © 2010 Microsoft Corporation.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products:

- Microsoft® Exchange Server 2003
- Microsoft® Office Outlook® 2007
- Microsoft® Exchange Server 2007
- Microsoft® Exchange Server 2010

Exceptions, if any, are noted below. If a service pack number appears with the product version, behavior changed in that service pack. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that product does not follow the prescription.

<1> Section 2.2.1: Office Outlook 2007 always formats parameters as UTF-16.

<2> Section 2.2.1.4: Office Outlook 2007 always uses "7" as the Degree of Difficulty value.

Copyright © 2010 Microsoft Corporation.

7 Change Tracking

This section identifies changes that were made to the [MS-OXPSVAL] protocol document between the May 2010 and August 2010 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

Copyright © 2010 Microsoft Corporation.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type "Editorially updated."

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1.1 Glossary	56305 Added the following to the list of terms that are defined in [MS-OXGLOS]: base64 encoding, header, identifier, message ID (MID), recipient, and resource. Removed the term "spam confidence level" from the list of terms that are defined in [MS-OXGLOS].	N	Content update.
1.2.1 Normative References	55751 Moved [MS-OXGLOS] from Normative References section to Informative References section.	N	Content update.
2.2.1 Input Parameters for Generating the Puzzle	48540 Added information to indicate that the client chooses between UCS-2 and UTF-16.	N	New content added.
Multiple sections	55921 Changed "[FIP180-1]" to "[FIPS180]".	N	Content update.

Copyright © 2010 Microsoft Corporation.

8 Index

Copyright © 2010 Microsoft Corporation.