[MS-SPNG]: Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) Extension

Revision Summary

Date	Revision History	Revision Class	Comments
10/22/2006	0.01		MCPP Milestone 1 Initial Availability
01/19/2007	1.0		MCPP Milestone 1
03/02/2007	1.1		Monthly release
04/03/2007	1.2		Monthly release
05/11/2007	1.3		Monthly release
06/01/2007	1.3.1	Editorial	Revised and edited the technical content.
07/03/2007	1.3.2	Editorial	Revised and edited the technical content.
07/20/2007	1.3.3	Editorial	Revised and edited the technical content.
08/10/2007	2.0	Major	Updated and revised the technical content.
09/28/2007	3.0	Major	Updated and revised the technical content.
10/23/2007	4.0	Major	Added technical clarifications.
11/30/2007	5.0	Major	Updated and revised the technical content.
01/25/2008	5.0.1	Editorial	Revised and edited the technical content.
03/14/2008	5.0.2	Editorial	Revised and edited the technical content.
05/16/2008	5.0.3	Editorial	Revised and edited the technical content.
06/20/2008	6.0	Major	Updated and revised the technical content.
07/25/2008	6.0.1	Editorial	Revised and edited the technical content.
08/29/2008	6.0.2	Editorial	Revised and edited the technical content.
10/24/2008	6.0.3	Editorial	Revised and edited the technical content.
12/05/2008	7.0	Major	Updated and revised the technical content.
01/16/2009	7.0.1	Editorial	Revised and edited the technical content.
02/27/2009	7.0.2	Editorial	Revised and edited the technical content.
04/10/2009	7.1	Minor	Updated the technical content.
05/22/2009	7.2	Minor	Updated the technical content.
07/02/2009	7.3	Minor	Updated the technical content.
08/14/2009	7.4	Minor	Updated the technical content.

Date	Revision History	Revision Class	Comments
09/25/2009	7.5	Minor	Updated the technical content.
11/06/2009	7.5.1	Editorial	Revised and edited the technical content.
12/18/2009	8.0	Major	Updated and revised the technical content.
01/29/2010	8.1	Minor	Updated the technical content.
03/12/2010	8.2	Minor	Updated the technical content.
04/23/2010	8.3	Minor	Updated the technical content.
06/04/2010	8.4	Minor	Updated the technical content.
07/16/2010	8.5	Minor	Clarified the meaning of the technical content.
08/27/2010	8.5	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	9.0	Major	Significantly changed the technical content.
11/19/2010	10.0	Major	Significantly changed the technical content.
01/07/2011	10.0	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	10.1	Minor	Clarified the meaning of the technical content.
03/25/2011	10.1	No change	No changes to the meaning, language, or formatting of the technical content.
05/06/2011	10.1	No change	No changes to the meaning, language, or formatting of the technical content.
06/17/2011	10.2	Minor	Clarified the meaning of the technical content.
09/23/2011	10.2	No change	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	11.0	Major	Significantly changed the technical content.
03/30/2012	11.0	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	11.1	Minor	Clarified the meaning of the technical content.
10/25/2012	11.2	Minor	Clarified the meaning of the technical content.
01/31/2013	11.2	No change	No changes to the meaning, language, or formatting of the technical content.
08/08/2013	12.0	Major	Significantly changed the technical content.

Contents

1 Introduction

The Simple and Protected Generic Security Service Application Program Interface Negotiation Mechanism (SPNEGO): Microsoft Extension is an extension to [RFC4178] that provides a negotiation mechanism for the **Generic Security Service** Application Program Interface (GSS-API), as specified in [RFC2743]. SPNEGO provides a framework for two parties that are engaged in authentication to select from a set of possible authentication mechanisms, in a manner that preserves the opaque nature of the **security protocols** to the **application protocol** that uses SPNEGO. SPNEGO was first defined in [RFC2478], which has been superseded by [RFC4178].

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

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

application protocol
domain name (2)
Generic Security Services (GSS)
object identifier (OID)
original equipment manufacturer (OEM) code page
remote procedure call (RPC)
security protocol
security token

The following terms are specific to this document:

ASN.1 Header: The top-level ASN.1 tag of the message.

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

1.2 References

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

A reference marked "(Archived)" means that the reference document was either retired and is no longer being maintained or was replaced with a new document that provides current implementation details. We archive our documents online [Windows Protocol].

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.

[ISO/IEC-8859-1] International Organization for Standardization, "Information Technology -- 8-Bit Single-Byte Coded Graphic Character Sets -- Part 1: Latin Alphabet No. 1", ISO/IEC 8859-1, 1998, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=28245

Note There is a charge to download the specification.

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

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000, http://www.ietf.org/rfc/rfc2743.txt

[RFC4178] Zhu, L., Leach, P., Jaganathan, K., and Ingersoll, W., "The Simple and Protected Generic Security Service Application Program Interface (GSS-API) Negotiation Mechanism", RFC 4178, October 2005, http://www.ietf.org/rfc/rfc4178.txt

[X680] ITU-T, "Abstract Syntax Notation One (ASN.1): Specification of Basic Notation", Recommendation X.680, July 2002, http://www.itu.int/rec/T-REC-X.680/en

Note There is a charge to download the specification.

[X690] ITU-T, "Information Technology - ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", Recommendation X.690, July 2002, http://www.itu.int/rec/T-REC-X.690/en

Note There is a charge to download the specification.

1.2.2 Informative References

[HTTPAUTH] Jaganathan, K., Zhu, L., and Brezak, J., "Kerberos based HTTP Authentication in Windows", July 2005, http://tools.ietf.org/html/draft-jaganathan-kerberos-http-01

[KAUFMAN] Kaufman, C., Perlman, R., and M. Speciner, "Network Security: Private Communication in a Public World, Second Edition", Prentice Hall, 2002, ISBN: 0130460192.

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

[MS-KILE] Microsoft Corporation, "Kerberos Protocol Extensions".

[MS-NLMP] Microsoft Corporation, "NT LAN Manager (NTLM) Authentication Protocol".

[MS-RPCE] Microsoft Corporation, "Remote Procedure Call Protocol Extensions".

[MS-SMB] Microsoft Corporation, "Server Message Block (SMB) Protocol".

[NEGOEX-DRAFT] Short, M., Zhu, L., Damour, K., and McPherson, D., "The Extended GSS-API Negotiation Mechanism (NEGOEX)", December 2010, http://tools.ietf.org/id/draft-zhu-negoex-02.txt

If you have any trouble finding [NEGOEX-DRAFT], please check here.

[PKU2U-DRAFT] Zhu, L., Altman, J., and Williams, N., "Public Key Cryptography Based User-to-User Authentication (PKU2U)", November 2008, http://tools.ietf.org/id/draft-zhu-pku2u-09.txt

If you have any trouble finding [PKU2U-DRAFT], please check here.

[RFC1964] Linn, J., "The Kerberos Version 5 GSS-API Mechanism", RFC 1964, June 1996, http://www.ietf.org/rfc/rfc1964.txt

[RFC2251] Wahl, M., Howes, T., and Kille, S., "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997, http://www.ietf.org/rfc/rfc2251.txt

[RFC2478] Baize, E., and Pinkas, D., "The Simple and Protected GSS-API Negotiation Mechanism", RFC 2478, December 1998, http://www.ietf.org/rfc/rfc2478.txt

[RFC4120] Neuman, C., Yu, T., Hartman, S., and Raeburn, K., "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005, http://www.ietf.org/rfc/rfc4120.txt

[UUKA-GSSAPI] Swift, M., Brezak, J., and Moore, P., "User to User Kerberos Authentication using GSS-API", October 2001, http://www.watersprings.org/pub/id/draft-swift-win2k-krb-user2user-03.txt

If you have any trouble finding [UUKA-GSSAPI], please check here.

1.3 Overview

1.3.1 Security Background

SPNEGO is a security protocol. As such, the normative references and this specification use common security-related terms. Every effort has been made to use these terms, such as principal, key, and service, in accordance with their use in [RFC4178].

Anyone who wants to understand the variations between SPNEGO Protocol Extensions and [RFC4178] should have a working knowledge of the Generic Security Service API. Several of the informative references, specifically [KAUFMAN], provide excellent top-level information about Generic Security Services (GSS) and the message flow. [KAUFMAN] also provides an excellent survey of other security protocols and concepts, and it helps to explain the terms of art that this specification uses. For more information, see [KAUFMAN].

Historically, the first GSS security mechanism defined was the Kerberos protocol (for more information, see [RFC1964]). The Kerberos protocol has influenced many other mechanisms; in some cases, it may prove helpful to have an example protocol to compare against. Finally, there are details that are not immediately apparent, as specified in [RFC4178] and [RFC2743].

1.3.2 SPNEGO Synopsis

SPNEGO is a security protocol that uses a GSS-API authentication mechanism. GSS–API is a literal set of functions that include both an API and a methodology for approaching authentication. As specified in [RFC2743], GSS-API and the individual security protocols that correspond to the GSS–API (also shortened to GSS) were developed because of the need to insulate application protocols from the specifics of security protocols as much as possible.

This approach led to a simplified form of interaction between an application protocol and an authentication protocol. In this model, an application protocol is responsible for ferrying discrete, opaque packets that the authentication protocol produces. These packets, which are referred to as **security tokens** by the GSS specifications, implement the authentication process. The application protocol has no visibility into the contents of the security tokens; its responsibility is merely to carry them.

The application protocol in this model first invokes the authentication protocol on the client. The client portion of the authentication protocol creates a security token and returns it to the calling application. The application protocol then transmits that security token to the server side of its connection, embedded within the application protocol. On the server side, the server's application protocol extracts the security token and supplies it to the authentication protocol on the server side. The server authentication protocol can process the security token and possibly generate a response;

or it can decide that authentication is complete. If another security token is generated, the application protocol MUST carry it back to the client where the process continues.

This exchange of security tokens continues until one side determines that authentication has failed or both sides decide that authentication is complete. If authentication fails, the application protocol drops the connection and indicates the error. If authentication succeeds, the application protocol can be assured of the identity of the participants as far as the supporting authentication protocol can accomplish. The onus of determining success or failure is on the abstracted security protocol, not the application protocol, which greatly simplifies the application protocol author's task.

After the authentication is complete, session-specific security services may be available. The application protocol can then invoke the authentication protocol to sign or encrypt the messages that are sent as part of the application protocol. The session-specific security services operations are done in much the same way, where the application protocol can indicate which portions of the message are to be encrypted, and the application protocol MUST include a per-message security token. By signing or encrypting the messages, the application can obtain message privacy and integrity, and detect message loss, out of order delivery and duplication.

Because more than one GSS–compatible authentication protocol exists, determining which protocol to use has become more important. The original GSS design had a static, compile-time binding between the application and the GSS implementation. More recent practice is to support more than one authentication mechanism—even for a single application protocol.

SPNEGO fills this need by presenting a GSS–compatible wrapper to other GSS mechanisms. It securely negotiates among several authentication mechanisms, selecting one for use to satisfy the authentication needs of the application protocol.

SPNG has errors in early implementations and an optimization for certain non–GSS scenarios. These variations form the basis of this specification.

1.3.3 SPNG Message Flow

SPNG message flow is composed of the following exchange:

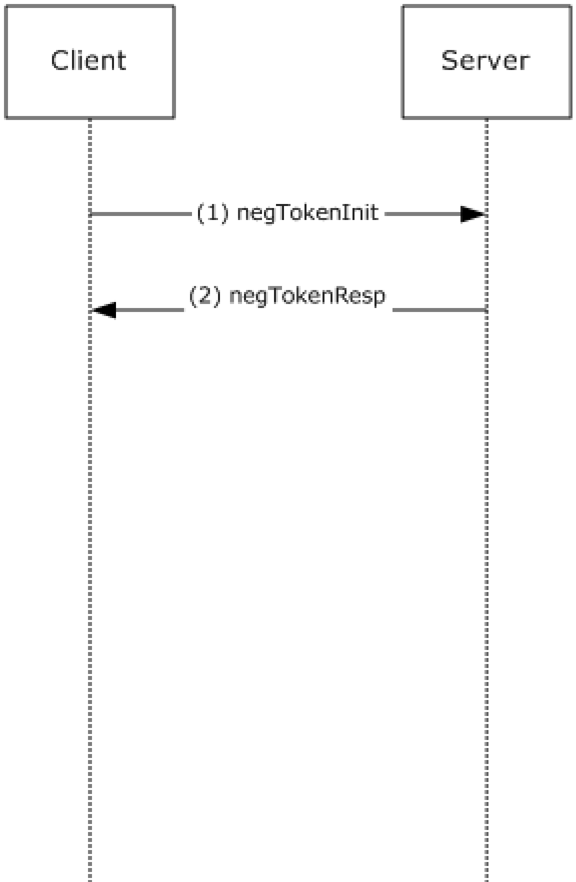

Figure 1: SPNG exchange

1. The client sends a **negTokenInit** message to the server. This message specifies the available authentication methods and an optimistic token.
2. The server sends a **negTokenResp** message to the client. The message specifies the state of the negotiation.

1.3.4 Server Initiated SPNG Message Flow

Server-initiated SPNG is composed of a three-way exchange:

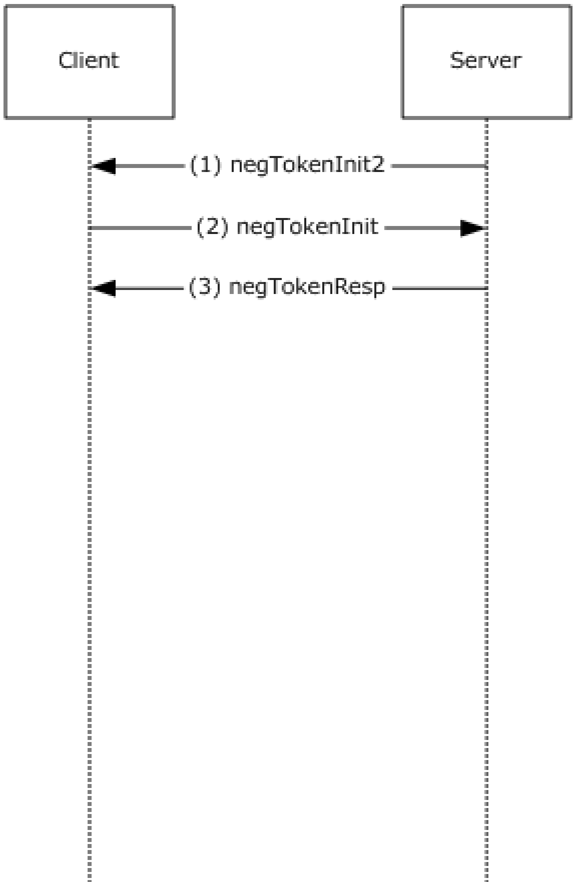

Figure 2: SPNG exchange

1. The server sends a **negTokenInit2** message to the client. This message specifies the available authentication methods and an optimistic token.
2. The client sends a **negTokenInit** message to the server. This message specifies the available authentication methods and an optimistic token.
3. The server sends a **negTokenResp** message to the client. The message specifies the state of the negotiation.

1.4 Relationship to Other Protocols

SPNEGO requires at least one other GSS–compatible authentication protocol to be present for it to work. It does not depend on a specific protocol. Windows implementations of SPNEGO negotiate the following authentication protocols by using the **object identifier (OID)** assigned to them:

- Kerberos Network Authentication Service (V5) protocol [RFC4120] [MS-KILE].
- User to User Kerberos Authentication [UUKA-GSSAPI].
- Extended GSS-API Negotiation Mechanism (NEGOEX) protocol [NEGOEX-DRAFT].<1> The OID assigned for NEGOEX is

iso.org.dod.internet.private.enterprise.Microsoft.security.mechanisms.NegoEx (1.3.6.1.4.1.311.2.2.30).

- NT LAN Manager (NTLM) Authentication Protocol [MS-NLMP].

Since NEGOEX negotiates security mechanisms, applications which use SPNEGOas their authentication protocol can use protocols supported by NEGOEX. Windows implementations of NEGOEX negotiate the following authentication protocols by the corresponding OIDs and AuthScheme **GUIDs**: so.org.dod.internet.security.kerberosv5.PKU2U<2> The OID and GUID assigned for PKU2U [PKU2U-DRAFT] is (1.3.6.1.5.2.7) 235F69AD-73FB-4dbc-8203-0629E739339B.

Many application protocols make use of SPNEGO as their authentication protocol. Such protocols include the Common Internet File System (CIFS)/Server Message Block (SMB) [MS-SMB]; HTTP [HTTPAUTH]; RPCE [MS-RPCE]; and the Lightweight Directory Access Protocol (LDAP) [RFC2251].

SPNEGO is a meta protocol that travels entirely in other application protocols; it is never used directly without an application protocol.

After SPNEGO has completed the ferrying of the other security protocol's authentication tokens, SPNEGO is finished. All further access to security context state and per-message services, such as signatures or encryption, is done by directly using the "real" security protocol whose authentication tokens were communicated via SPNEGO.

1.5 Prerequisites/Preconditions

Because SPNEGO relies on other security protocols that perform authentication, those protocols must be available to SPNEGO for it to operate. The set of protocols is implementation-dependent upon the installation.<3>

Applications typically establish a connection before they invoke SPNEGO, although establishing a connection before invoking SPNEGO is not required by the SPNEGO protocol.

1.6 Applicability Statement

As a GSS protocol, SPNEGO can be used almost anywhere that an application protocol uses GSS to perform authentication. The protocol must be connection-oriented because it is not designed to tolerate packet loss; datagram-only protocols cannot support negotiation of this form.

1.7 Versioning and Capability Negotiation

SPNEGO does not contain any versioning capacity. The same is true for capabilities: any capability negotiation must be performed by the actual authentication protocols that SPNEGO is carrying.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

1.9.1 Use of Constants Assigned Elsewhere

SPNEGO has been assigned the following object identifier (OID):

- iso.org.dod.internet.security.mechanism.snego

(1.3.6.1.5.5.2)

2 Messages

2.1 Transport

SPNEGO is transported only when encapsulated in an application protocol. As such, it can travel over whatever transports the application protocol uses. By itself, SPNEGO never causes network traffic.

2.2 Message Syntax

The messages that the base SPNEGO protocol uses are specified in [RFC4178], in terms of ASN.1, as specified in [X680]. There are only two messages in SPNEGO, negTokenInit and negTokenResp, both of which are defined in [RFC4178].

The negTokenInit message is sent from the client to the server and is used to begin the negotiation. The client uses that message to specify the set of authentication mechanisms that are supported and an opportunistic authentication message from the mechanism that the client believes will be agreed upon with the server.

The negTokenResp message is used thereafter as the server selects the mechanism to use, and the two parties exchange authentication messages that are wrapped in the negTokenResp message until completion. SPNG supports the NegTokenInit2 message.

2.2.1 NegTokenInit2

The SPNEGO Protocol Extensions extend the NegTokenInit with a negotiation hints field. The NegTokenInit2 message is structured as follows.<4>

```
NegHints ::= SEQUENCE {
        hintName[0] GeneralString OPTIONAL,
        hintAddress[1] OCTET STRING OPTIONAL
}
NegTokenInit2 ::= SEQUENCE {
        mechTypes[0] MechTypeList OPTIONAL,
        reqFlags [1] ContextFlags OPTIONAL,
        mechToken [2] OCTET STRING OPTIONAL,
        negHints [3] NegHints OPTIONAL,
        mechListMIC [4] OCTET STRING OPTIONAL,
        ...
}
```

mechTypes: The list of authentication mechanisms that are available, by OID, as specified in [RFC4178] section 4.1.

reqFlags: As specified in [RFC4178] section 4.2.1 This field SHOULD be omitted by the sender.

mechToken: The optimistic mechanism token ([RFC4178] section 4.2.1).

negHints: The server supplies the negotiation hints using a **negHints** (negotiation hints) structure that is assembled as follows.

- **hintName**: Contains the string "not_defined_in_RFC4178@please_ignore".<5>
- **hintAddress**: Never present. MUST be omitted by the sender. Note that the encoding rules, as specified in [X690], require that this structure not be present at all, not just be zero.

mechListMIC: The MIC token ([RFC4178] section 4.2.1).

Note In the ASN.1 description in the preceding, the NegTokenInit2 message occupies the same context-specific ([X690] section 8.1.2.2) message ID (0) as does NegTokenInit in SPNEGO.

3 Protocol Details

3.1 Common Details

The following are common variations, as specified in [RFC4178], for both client and server processing in the SPNEGO Protocol Extensions.

3.1.1 Abstract Data Model

The SPNEGO Protocol Extensions make no extensions to the abstract data model for SPNEGO.

This protocol includes the following ADM elements, which are directly accessed from NLMP as specified in [MS-NLMP] section 3.4.1:

- ClientHandle
- ServerHandle

SPNEGO exports a set of abstract parameters that describe the security services that a caller wants to have available for use on the communication session after it has been established. SPNEGO cannot directly act on these parameters because it does not perform the actual authentication. They are passed through to the underlying security protocols as an indication of the caller's eventual plans. These parameters are:

- Integrity: A Boolean setting that indicates that the caller wants to sign messages so that they cannot be tampered with while in transit.
- Replay Detect: A Boolean setting that indicates that the caller wants to sign messages so that they cannot be replayed.
- Sequence Detect: A Boolean setting that indicates that the caller wants to sign messages so that they cannot be sent out of order.
- Confidentiality: A Boolean setting that indicates that the caller wants to encrypt messages so that they cannot be read while in transit.
- Delegate: A Boolean setting that indicates that the caller wants to make its own identity available to the server for further identification to other services.
- Mutual Authentication: A Boolean setting that indicates that the client and server MUST authenticate each other; unidirectional authentication is not permissible.

  These flags correspond to the **reqFlags:ContextFlags** field in the NegTokenInit structure. As specified in [RFC4178], the **reqFlags:ContextFlags** field is now only for legacy purposes and SHOULD NOT be filled in. For more information about the **reqFlags:ContextFlags** field, see section 3.1.5.3.

- Extended Error: A Boolean setting that indicates that the caller wants the underlying protocol to perform the extended error handling, potentially including retries within the GSS exchange.
- FragmentToFit: A Boolean setting that indicates that the caller directs the underlying protocol to fragment messages.<6>
- MaxOutputTokenSize: The maximum size, in bytes, of output_token that can be returned to the caller. This value MUST be at least 5 bytes to contain the entire **ASN.1 header**, so that the recipient can reconstruct the length of the completed message. Applications that request small

buffers can significantly increase the number of round trips. An application can have limitations on the number of round trips allowed, which means that setting the buffers too small can cause failures. Also, authentication protocols can be sensitive to clock skews between the client and server, which can cause failures if the amount of time required to transmit all the messages is too long.

The following temporary variables are used by the fragmenting functions:

- FragmentInputToken: A Boolean setting that indicates that more fragments of input_token remain.
- ReceivedInputToken: The fragments of input_token received.
- TokenLength: The length of input_token.
- FragmentOutputToken: A Boolean setting that indicates that more fragments of output_token remain.
- RemainingOutputToken: The remaining message to be sent.

The following temporary variable is used to reset the NLMP RC4 handle:

- OriginalHandle

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Trigger Events

None.

3.1.5 Processing Events and Sequencing Rules

The following fields are processed differently than as specified in [RFC4178].

3.1.5.1 mechListMIC Processing

[RFC2478] inadequately specifies the processing of the mechanism list Message Integrity Code, or **mechListMIC** field. [RFC4178] clarifies the processing of the **mechListMIC** field.<7>

3.1.5.2 mechTypes Identification of Kerberos

An implementation SHOULD use the standard Kerberos OID (1.2.840.113554.1.2.2), as described in [RFC4120], for identification of the Kerberos mechType<8> and the OID described in [UUKA-GSSAPI] section 4 for identification of the Kerberos user-to-user mechType.

3.1.5.3 reqFlags Processing

[RFC2478], the predecessor to [RFC4178], includes the **reqFlags** field in the protocol. This field is intended for the client to indicate the requested behavior according to the GSS abstract variables, such as confidentiality and integrity. However, the **reqFlags** field is not covered by the signature of the message; therefore, it can be tampered with while in transit.

As specified in [RFC4178], use of this field is explicitly discouraged due to the lack of integrity protection, and the acceptor (server) MUST ignore the **reqFlags**, if present.

3.1.5.4 InitAssembleToken()

```
InitFragmentToken (Token, MaxOutputTokenSize, OutputToken)
-- Input:
--   MaxOutputTokenSize - Maximum size, in bytes, of OutputToken that can be
     returned to the caller. MUST be greater than 5.
--   Token – The Token message to be fragmented.
-- Internal Temporary variables that do not pass over the wire are defined below:
--   RemainingOutputToken – The remaining message to be sent.
--   FragmentOutputToken - A Boolean setting that indicates that more fragments of the
output_token remain.
-- Output:
--   OutputToken – The first fragment of the message passed to the caller.

Initialize RemainingOutputToken to Token.
Set FragmentOutputToken to TRUE
Set OutputToken to first MaxOutputTokenSize bytes of RemainingOutputToken
Delete first MaxOutputTokenSize bytes of RemainingOutputToken
```

3.1.5.5 FragmentToken()

```
FragmentToken(OutputToken)
-- Internal Temporary variables that do not pass over the wire are defined below:
--   MaxOutputTokenSize - Maximum size, in bytes, of the OutputToken that can be
     returned to the caller. MUST be greater than 5.
--   RemainingOutputToken – The remaining message to be sent.
--   FragmentOutputToken - A Boolean setting that indicates that more fragments of the
OutputToken remain.
-- Output:
--   OutputToken – The OutputToken passed to the client.

If size of RemainingOutputToken > MaxOutputTokenSize
   Set OutputToken to first MaxOutputTokenSize bytes of RemaininggOutputToken
   Delete first MaxOutputTokenSize bytes of RemainingOutputToken
Else
   Set OutputToken to RemainingOutputToken
   Set RemainingOutputToken to empty
   Set FragmentOutputToken to FALSE
EndIf
```

3.1.5.6 Send Fragmented Messages

The first fragment includes the ASN.1 header for the message, so that the recipient can reconstruct the length of the completed message. This requires that **MaxOutputTokenSize** be at least 5 bytes.

SPNG calls InitFragmentToken (section 3.1.5.4), where:

- **Token** contains the message.
- **MaxOutputTokenSize** contains the **MaxOutputTokenSize** provided by the application.

SPNG MUST return GSS_S_CONTINUE_NEEDED and an initial packet containing **OutputToken**.

When **FragmentOutputToken** is set to TRUE, SPNG calls FragmentToken (section 3.1.5.5) to get the next fragment, and MUST return GSS_S_CONTINUE_NEEDED and **OutputToken**. If **FragmentOutputToken** is not set to TRUE, SPNG MUST return GSS_S_COMPLETE.

If the server does not support fragmentation, the application service receives an error from its GSS_Accept_sec_context call, and the negotiation fails. Whether the client application receives the error depends on the application service behavior.

3.1.5.7 InitAssembleToken()

```
InitAssembleToken (Input_Token)
-- Input:
--  InputToken – The Input_Token received.
--  Temporary variables that do not pass over the wire are defined below:
--   ReceivedInputToken – The message fragments received so far.
--   TokenLength - Length of message from the ASN.1 header.
--   FragmentInputToken - A Boolean setting that indicates that more fragments of the message
remain.

Initialize TokenLength to the length of the message from the ASN.1 header in InputToken.
Initialize ReceivedInputToken to InputToken.
Set FragmentInputToken to TRUE.
```

3.1.5.8 AssembleToken()

```
AssembleToken(Input_Token, OutputToken)
-- Input:
--   InputToken – The Input_Token received.
-- Temporary variables that do not pass over the wire are defined below:
--  ReceivedInputToken – The message fragments received so far.
--  TokenLength - Length of message from the ASN.1 header.
--  FragmentInputToken - A Boolean setting that indicates that more fragments of the
InputToken remain.
-- Output:
--   OutputToken – The OutputToken returned, or the complete InputToken.

Append InputToken to ReceivedInputToken
If TokenLength > length of ReceivedInputToken
   Set OutputToken to empty
Else
   Set OutputToken to ReceivedInputToken
   Set ReceivedInputToken to empty
   Set FragmentInputToken to FALSE.
EndIf
```

3.1.5.9 Receive Fragmented Messages

The length specified in the ASN.1 header of the first packet is used to determine the number of bytes necessary to assemble the complete message. SPNG calls InitAssembleToken (section 3.1.5.7), where **Input_Token** contains the **Input_Token** received from the caller. To receive the next fragment, SPNG MUST return GSS_S_CONTINUE_NEEDED with an empty OutputToken.

When **FragmentInputToken** is set to TRUE, SPNG calls AssembleToken (section 3.1.5.8), where **Input_Token** contains the **Input_Token** received. If the **OutputToken** is not empty, the message is complete and processing can continue as normal. Otherwise, to receive the next fragment, SPNG MUST return GSS_S_CONTINUE_NEEDED with an empty **OutputToken**.

If the context is terminated before reassembly of the message is complete (for example, because the network connection to the other entity is interrupted), the entire message MUST be discarded.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server (Acceptor) Role Details

3.2.1 Abstract Data Model

The abstract data model for the server is specified in section 3.1.1.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Processing Events and Sequencing Rules

The server SHOULD ignore the negHints in the negTokenInit2 message.

The server MUST use the erroneous Kerberos value (1.2.840.48018.1.2.2) as the **supportedMech** field in the response negotiation token if the optimistic Kerberos token (1.2.840.48018.1.2.2) is accepted.

The SPNG server SHOULD invoke Send Fragmented Messages (section 3.1.5.6) when a GSS_Accept_sec_context() ([RFC2743] section 2.2.2) with the *FragmentToFit* parameter set to TRUE (section 3.1.1) is received, and either:

- The Negotiate Token ([RFC4178] section 4.2) to be sent exceeds **MaxOutputTokenSize**, or

- FragmentOutputToken is set to TRUE.

The server MUST invoke Receive Fragmented Messages (section 3.1.5.9) when a packet is received and either:

- the packet contains a valid ASN.1 header but an incomplete body, or
- FragmentOutputToken is set to TRUE.

3.2.5.1 NTLM RC4 Key State for MechListMIC and First Signed Message

When NTLM is negotiated, the SPNG server MUST set **OriginalHandle** to **ServerHandle** before generating the mechListMIC, then set **ServerHandle** to **OriginalHandle** after generating the mechListMIC. This results in the RC4 key state being the same for the mechListMIC and for the first message signed by the application.

Because the RC4 key state is the same for the mechListMIC and for the first message signed by the application, the SPNG server MUST set **OriginalHandle** to **ClientHandle** before validating the mechListMIC and then set **ClientHandle** to **OriginalHandle** after validating the mechListMIC.

3.2.5.2 NegTokenInit2 Variation for Server-Initiation

Standard GSS has a strict notion of client (initiator) and server (acceptor). If client has not sent a negTokenInit ([RFC4178] section 4.2.1) message, no context establishment token is expected from the server.

SPNG allows the server to generate a context establishment token message such as a NegTokenInit2 message and send it to the client when GSS_Accept_sec_context() is called without an input_token.

The server generates a NegTokenInit2 message that includes the OIDs of the security protocols that are present and available on the server in the **mechTypes** field.

In the **negHints** field, the server places the string "not_defined_in_RFC4178@please_ignore"<9>, expressed as ANSI encoding, as specified in [ISO/IEC-8859-1], in the **hintName** field. For more information about how the **hintName** field is populated, see section 2.2.1.

The **hintAddress** field MUST be omitted and not transmitted. The NegTokenInit2 token is then passed to the client within the application protocol. When encoding the name, the configured locale on the computer SHOULD be used for the resulting character set.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 Client (Initiator) Role Details

3.3.1 Abstract Data Model

The abstract data model for the client is specified in section 3.1.1.

3.3.2 Timers

None.

3.3.3 Initialization

The client MUST request Mutual Authentication services, as defined in section 3.1.1.

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

The SPNG client SHOULD invoke Send Fragmented Messages (section 3.1.5.6) when a GSS_Accept_sec_context() ([RFC2743] section 2.2.2) with the *FragmentToFit* parameter set to TRUE (section 3.1.1) is received, and either:

- The Negotiate Token ([RFC4178] section 4.2) to be sent exceeds **MaxOutputTokenSize**, or
- FragmentOutputToken is set to TRUE.

The server MUST invoke Receive Fragmented Messages (section 3.1.5.9) when a packet is received and either:

- The packet contains a valid ASN.1 header but an incomplete body, or
- FragmentOutputToken is set to TRUE.

To support non-complaint implementations of [RFC4178] that send a **supportedMech** field in a subsequent NegTokenResp message, the SPNG client MAY accept the message without returning an error, but MUST ignore the new **supportedMech** field.<10>

3.3.5.1 NTLM RC4 Key State for MechListMIC and First Signed Message

When NTLM is negotiated, the SPNG client MUST set **OriginalHandle** to **ClientHandle** before generating the mechListMIC and then set **ClientHandle** to **OriginalHandle** after generating the mechListMIC. This results in the RC4 key state being the same for the mechListMIC and for the first message signed by the application.

Because the RC4 key state is the same for the mechListMIC and for the first message signed by the application, the SPNG server MUST set **OriginalHandle** to **ServerHandle** before validating the mechListMIC and then set **ServerHandle** to **OriginalHandle** after validating the mechListMIC.

3.3.5.2 NegTokenInit2 Variation for Server-Initiation

Standard GSS has a strict notion of client (initiator) and server (acceptor). If the client is not waiting for a response from the server from a sent negTokenInit ([RFC4178] section 4.2.1) and the client receives a NegTokenInit2 (section 2.2.1) message from a server, then the client SHOULD process messages for the received token.

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

4 Protocol Examples

The following is an annotated hex dump of an ASN.1 encoded NegTokenInit2 (section 2.2.1) message.

```
00000000  60 82 01 5d 06 06 2b 06 01 05 05 02 a0 82 01 51  `..]..+........Q
00000010  30 82 01 4d a0 1a 30 18 06 0a 2b 06 01 04 01 82  0..M..0...+.....
00000020  37 02 02 1e 06 0a 2b 06 01 04 01 82 37 02 02 0a  7.....+.....7...
00000030  a2 82 01 01 04 81 fe 4e 45 47 4f 45 58 54 53 01  .......NEGOEXTS.
00000040  00 00 00 00 00 00 00 60 00 00 00 70 00 00 00 cf  .......`...p....
00000050  fa 11 76 5e 12 59 9a 34 7d 76 68 52 bf ce 70 97  ..v^.Y.4}vhR..p.
00000060  45 87 10 bb 82 42 b4 c7 df ba d2 da 89 7a a3 11  E....B.......z..
00000070  a7 d8 68 46 34 30 95 25 62 dc 13 c5 54 f2 01 00  ..hF40.%b...T...
00000080  00 00 00 00 00 00 00 60 00 00 00 01 00 00 00 00  .......`........
00000090  00 00 00 00 00 00 00 5c 33 53 0d ea f9 0d 4d b2  .......\3S....M.
000000a0  ec 4a e3 78 6e c3 08 4e 45 47 4f 45 58 54 53 03  .J.xn..NEGOEXTS.
000000b0  00 00 00 01 00 00 00 40 00 00 00 8e 00 00 00 cf  .......@........
000000c0  fa 11 76 5e 12 59 9a 34 7d 76 68 52 bf ce 70 5c  ..v^.Y.4}vhR..p\
000000d0  33 53 0d ea f9 0d 4d b2 ec 4a e3 78 6e c3 08 40  3S....M..J.xn..@
000000e0  00 00 00 4e 00 00 00 30 4c a0 4a 30 48 30 2a 80  ...N...0L.J0H0*.
000000f0  28 30 26 31 24 30 22 06 03 55 04 03 13 1b 58 4d  (0&1$0"..U....XM
00000100  4c 50 72 6f 76 69 64 65 72 20 49 6e 74 65 72 6d  LProvider Interm
00000110  65 64 69 61 74 65 20 43 41 30 1a 80 18 30 16 31  ediate CA0...0.1
00000120  14 30 12 06 03 55 04 03 13 0b 58 4d 4c 50 72 6f  .0...U....XMLPro
00000130  76 69 64 65 72 a3 2a 30 28 a0 26 1b 24 6e 6f 74  vider.*0(.&.$not
00000140  5f 64 65 66 69 6e 65 64 5f 69 6e 5f 52 46 43 34  _defined_in_RFC4
00000150  31 37 38 40 70 6c 65 61 73 65 5f 69 67 6e 6f 72  178@please_ignor
00000160  65                                               e
```

The first part is the ASN.1 encoding of the NegTokenInit2 message. This is the same as for the netTokenInit ([RFC4178] section 4.2) message:

```
00000000  60 82 01 5d 06 06 2b 06 01 05 05 02 a0 82 01 51  `..]..+........Q
00000010  30 82 01 4d a0 1a 30 18                          0..M..0.
```

The **mechTypes** field is the first field of the NegTokenInit2 message. Since this is a local logon, two types are offered:

- SPNegoEx: iso(1).org(3).dod(6).internet(1).private(4).enterprise(1).Microsoft(311).security(2).mechanisms(2).snegoex(30)
- NLMP: iso(1).org(3).dod(6).internet(1).private(4).enterprise(1).Microsoft(311).security(2).mechanisms(2).ntlm(10)

```
00000010                          06 0a 2b 06 01 04 01 82          ..+.....
00000020  37 02 02 1e 06 0a 2b 06 01 04 01 82 37 02 02 0a  7.....+.....7...
```

Next is the **mechToken** field.

```
00000030  a2 82 01 01 04 81 fe 4e 45 47 4f 45 58 54 53 01  .......NEGOEXTS.
```

```
00000040  00 00 00 00 00 00 00 60 00 00 00 70 00 00 00 cf  .......`...p....
00000050  fa 11 76 5e 12 59 9a 34 7d 76 68 52 bf ce 70 97  ..v^.Y.4}vhR..p.
00000060  45 87 10 bb 82 42 b4 c7 df ba d2 da 89 7a a3 11  E....B.......z..
00000070  a7 d8 68 46 34 30 95 25 62 dc 13 c5 54 f2 01 00  ..hF40.%b...T...
00000080  00 00 00 00 00 00 00 60 00 00 00 01 00 00 00 00  .......`........
00000090  00 00 00 00 00 00 00 5c 33 53 0d ea f9 0d 4d b2  .......\3S....M.
000000a0  ec 4a e3 78 6e c3 08 4e 45 47 4f 45 58 54 53 03  .J.xn..NEGOEXTS.
000000b0  00 00 00 01 00 00 00 40 00 00 00 8e 00 00 00 cf  .......@........
000000c0  fa 11 76 5e 12 59 9a 34 7d 76 68 52 bf ce 70 5c  ..v^.Y.4}vhR..p\
000000d0  33 53 0d ea f9 0d 4d b2 ec 4a e3 78 6e c3 08 40  3S....M..J.xn..@
000000e0  00 00 00 4e 00 00 00 30 4c a0 4a 30 48 30 2a 80  ...N...0L.J0H0*.
000000f0  28 30 26 31 24 30 22 06 03 55 04 03 13 1b 58 4d  (0&1$0"..U....XM
00000100  4c 50 72 6f 76 69 64 65 72 20 49 6e 74 65 72 6d  LProvider Interm
00000110  65 64 69 61 74 65 20 43 41 30 1a 80 18 30 16 31  ediate CA0...0.1
00000120  14 30 12 06 03 55 04 03 13 0b 58 4d 4c 50 72 6f  .0...U....XMLPro
00000130  76 69 64 65 72 a3 2a 30 28 a0 26 1b 24           vider.*0(.&.$
```

Finally is the **negHints.hintName** field, the value of which is the string "not_defined_in_RFC4178@please_ignore".

```
00000130                                         6e 6f 74               not
00000140  5f 64 65 66 69 6e 65 64 5f 69 6e 5f 52 46 43 34  _defined_in_RFC4
00000150  31 37 38 40 70 6c 65 61 73 65 5f 69 67 6e 6f 72  178@please_ignor
00000160  65                                               e
```

5 Security

5.1 Security Considerations for Implementers

Implementers of the SPNEGO Protocol Extensions should be aware of the correct use of the hint data that the server sends, as specified in section 3.3.5.2.

5.2 Index of Security Parameters

Security parameter	Section
GSS context parameters	NegTokenInit Variation for Server-Initiation (section 3.3.5.2)

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Windows 2000 operating system
- Windows XP operating system
- Windows Server 2003 operating system
- Windows Vista operating system
- Windows Server 2008 operating system
- Windows 7 operating system
- Windows Server 2008 R2 operating system
- Windows 8 operating system
- Windows Server 2012 operating system
- Windows 8.1 operating system
- Windows Server 2012 R2 operating system

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

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

<1> Section 1.4: Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008 do not support NegoEX.

<2> Section 1.4: Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008 do not support PKU2U [PKU2U-DRAFT].

<3> Section 1.5: By default, the Kerberos protocol and NTLM are available in Windows. The interface for authentication protocols in Windows is open and extensible; other protocols may be installed on a specific system by third parties; and other protocols may be added as defaults in future versions of Windows.

<4> Section 2.2.1: Windows generates the NegTokenInit2 message.

<5> Section 2.2.1: In Windows 2000, Windows XP, and Windows Server 2003, the negHints.hintName field contained the name of the name of the server principal, which is the service principal on the server in the form user-name@domain-name. The name is expressed in ANSI encoding, which uses an **OEM code page** that the local system defines. For two parties to use this extension, the OEM code page must be agreed upon out-of-band of this protocol.

<6> Section 3.1.1: Windows exposes this logical parameter (FragmentToFit) to applications through the SSPI interface on Windows.

<7> Section 3.1.5.1: Windows 2000, Windows Server 2003, and Windows XP do not process the **mechListMIC** field. No **mechListMIC** value is produced when either the client or server completes the exchange. If a **mechListMIC** value is supplied to either the client or server, it is ignored. If the initiator in the GSS exchange has the last GSS token, the server returns a NegTokenResp token that has the **negState** field set to accept_complete and no **mechListMIC** field.

On Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2, if AES Kerberos ciphers are negotiated by Kerberos, the signature in the SPNEGO **mechListMIC** field MUST be processed by the recipient.

<8> Section 3.1.5.2: Windows versions offer and accept two distinct OIDs as identifiers for the Kerberos authentication mechanism.

Windows 2000 incorrectly encoded the OID for the Kerberos protocol in the supportedMech field. Rather than the OID { iso(1) member-body(2) United States(840) mit(113554) infosys(1) gssapi(2) krb5(2) }, an implementation error truncated the values at 16 bits. Therefore, the OID became { iso(1) member-body(2) United States(840) ???(48018) infosys(1) gssapi(2) krb5 (2) }.

Windows version	Offers/receives standard OID	Offers/receives truncated OID
Windows 2000		X
Windows XP	X	X
Windows Server 2003	X	X
Windows Vista	X	X
Windows Server 2008	X	X
Windows 7	X	X
Windows Server 2008 R2	X	X
Windows 8	X	X
Windows Server 2012	X	X
Windows 8.1	X	X
Windows Server 2012 R2	X	X

Windows clients will fail if the accepter accepts the preferred mechanism token (1.2.840.48018.1.2.2), and produces a response token with the supportedMech being the standard Kerberos OID (1.2.840.113554.1.2.2).

<9> Section 3.2.5.2: In Windows 2000, Windows XP, and Windows Server 2003, the **negHints.hintName** field contains the name of the name of the server principal, which is the service principal on the server in the form user-name@domain-name.

<10> Section 3.3.5: Windows 2000, Windows Server 2003, and Windows Vista do not support non-complaint implementations of [RFC4178] that send a **supportedMech** field in a subsequent NegTokenResp message.

7 Change Tracking

This section identifies changes that were made to the [MS-SPNG] protocol document between the January 2013 and August 2013 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.

- 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
3.3.5 Message Processing Events and Sequencing Rules	67816 Added information about the supportedMech field in subsequent NegTokenResp messages.	Y	Content updated.
6 Appendix A: Product Behavior	Modified this section to include references to Windows Server 2012 operating system, Windows 8.1 operating system, and Windows Server 2012 R2 operating system.	Y	Content updated.

8 Index

N

O

P

R

S

T

V