[MS-EAPE]:
EAP Extensions 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.** Portions of this document were contributed by The Technical Committee (www.thetc.org) and are reproduced with permission. Other portions are 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 Open Specification Promise or the Community Promise. 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.

Revision Summary

Date	Revision History	Revision Class	Comments
06/17/2011	0.1	New	This document was created by The Technical Committee.

Contents

1 Introduction

This document does not specify a new protocol. It specifies an extension to the Extensible Authentication Protocol [RFC3748] as it relates to Protected Extensible Authentication Protocol specified by [MS-PEAP] and Network Access Protection as specified by [MS-NAPSO]. Its main purpose is to introduce and describe the interactions with the NAP Agent. In particular this document clarifies the details of PEAP Phase 1 authentication triggered by the Clients attempt to access the protected network and the interfaces used to obtain a Statement of Health from the NAP Agent. This layer is referred to as the Outer EAP layer because within the PEAP protocol stack given by [MS-PEAP] section 1.4, it is one layer below it, and in effect indicates the desire to proceed with PEAP/TLS tunnel negotiation.

In order to understand this protocol the reader must be familiar with [MS-NAPSO], [MS-PEAP], [MS-SOH] and [MS-WSH].

1.1 Glossary

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

Active Directory Domain Services (AD DS)
Coordinated Universal Time (UTC)
Dynamic Host Configuration Protocol (DHCP)
Extensible Authentication Protocol (EAP)
globally unique identifier (GUID)
health registration authority (HRA)
Internet Protocol security (IPsec)
little-endian
Network Access Policy
Network Access Protection (NAP)
network access server (NAS)
RADIUS attribute
RADIUS client
RADIUS server
Remote Access Service (RAS) server
security identifier (SID)
statement of health (SoH)
statement of health response (SoHR)
Unicode

The following terms are specific to this document:

Outer EAP: The EAP layer indicating the desire to proceed with PEAP/TLS negotiation. The payload of this layer de-capsulated and then encapsulated in the RADIUS protocol by the Authenticator and sent to the Authentication Server [RFC3579]. Outer EAP is the EAPE Layer specified by this document.

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.

[IANA-EAP] Internet Assigned Numbers Authority, "Extensible Authentication Protocol (EAP) Registry", October 2006, http://www.iana.org/assignments/eap-numbers

[IANA-ENT] Internet Assigned Numbers Authority, "Private Enterprise Numbers", January 2007, http://www.iana.org/assignments/enterprise-numbers

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

[MS-NAPSO] Microsoft Corporation, "Network Access Protection System Overview".

[MS-SOH] Microsoft Corporation, "Statement of Health for Network Access Protection (NAP) Protocol Specification".

[RFC2104] Krawczyk, H., Bellare, M., and Canetti, R., "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997, http://www.ietf.org/rfc/rfc2104.txt

[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

[RFC2246] Dierks, T., and Allen, C., "The TLS Protocol Version 1.0", RFC 2246, January 1999, http://www.ietf.org/rfc/rfc2246.txt

[RFC2548] Zorn, G., "Microsoft Vendor-Specific RADIUS Attributes", RFC 2548, March 1999, http://www.ietf.org/rfc/rfc2548.txt

[RFC2865] Rigney, C., Willens, S., Rubens, A., and Simpson, W., "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000, http://www.ietf.org/rfc/rfc2865.txt

[RFC3174] Eastlake III, D., and Jones, P., "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001, http://www.ietf.org/rfc/rfc3174.txt

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., et al., "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004, http://www.ietf.org/rfc/rfc3748.txt

[RFC5216] Simon, D., Aboda, B., and Hurst, R., "The EAP-TLS Authentication Protocol", RFC 5216, March 2008, http://www.ietf.org/rfc/rfc5216.txt

[RFC5246] Dierks, T., and Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008, http://www.ietf.org/rfc/rfc5246.txt

[RFC5280] Cooper, D., Santesson, S., Farrell, S., et al., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008, http://www.ietf.org/rfc/rfc5280.txt

1.2.2 Informative References

[IEEE802.1X] Institute of Electrical and Electronics Engineers, "IEEE Standard for Local and Metropolitan Area Networks - Port-Based Network Access Control", December 2004, http://ieeexplore.ieee.org/iel5/9828/30983/01438730.pdf

[MS-CHAP] Microsoft Corporation, "Extensible Authentication Protocol Method for Microsoft Challenge Handshake Authentication Protocol (CHAP) Specification".

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

[MS-GPWL] Microsoft Corporation, "Group Policy: Wireless/Wired Protocol Extension".

[RFC1661] Simpson, W., Ed., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994, http://www.ietf.org/rfc/rfc1661.txt

[RFC1750] Eastlake III, D., Crocker, S., and Schiller, J., "Randomness Recommendations for Security", RFC 1750, December 1994, http://www.ietf.org/rfc/rfc1750.txt

[RFC4017] Stanley, D., Walker, J., and Aboba, B., "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs", RFC 4017, March 2005, http://www.ietf.org/rfc/rfc4017.txt

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005, http://www.ietf.org/rfc/rfc4306.txt

1.3 Protocol Overview (Synopsis)

The Outer EAP layer terminates at the PEAP Server and specifies PEAP as its authentication method which in turn will initiate a TLS session which goes through the NAP Capable Pass-through Proxy and terminates at the PEAP Server. Its main purpose is to complete PEAP Phase 1 authentication between PEAP Client and Server but as a side effect it supports enforcement of Network Access Protection. PEAP will establish a secure channel in which PEAP Phase 2 can initiate EAP to specify a higher layer EAP authentication method such as MS-CHAPv2 [MS-CHAP]. Outer EAP payloads are passed from the NAP Capable Client to the NAP Capable Pass-through Proxy over PPP or 8021.X, and are forwarded to the NAP Capable Server via the NAP Capable Pass-through Proxy as shown in Figure 1.

This Outer EAP layer specified by this document is encapsulated by the transport layer. On a NAP Capable Pass-through Proxy the lower layer over the transport layer will be a Remote Dial-In User Service protocol [RADIUS] implementation that may support Microsoft® specific attributes as specified by [MS-RNAP].

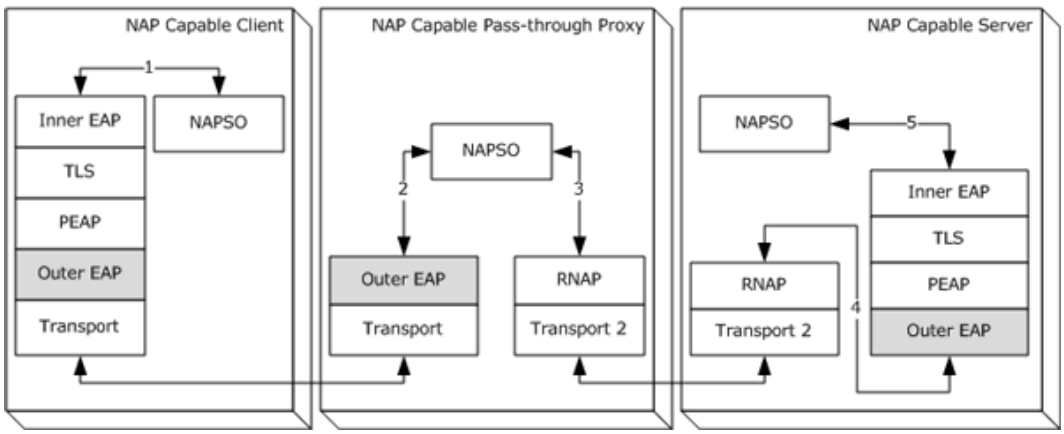

Figure 1: Typical deployment on NAP capable machines

The roles shown here are NAP Capable. A NAP capable EAP client sends EAP messages which are received at the NAP Capable Pass-through Proxy and with the usage of NAPSO defined interfaces passes them to RADIUS/RNAP Client, on the NAP Capable Pass-through Proxy which encapsulates them into RADIUS Messages using the EAP-Message Attribute. The RADIUS Client on the NAP Capable Pass-through Proxy will forward the messages to the RADIUS Server, on the NAP Capable Server where the RADIUS/RNAP Layer will de-capsulate the RADIUS message and pass the EAP Message content to the Outer EAP Layer specified by this document.

The NAPSO interfaces required are bidirectional and labeled from 1-5. This document specifies the usage of NAPSO interfaces 2 and 4.

1.4 Relationship to Other Protocols

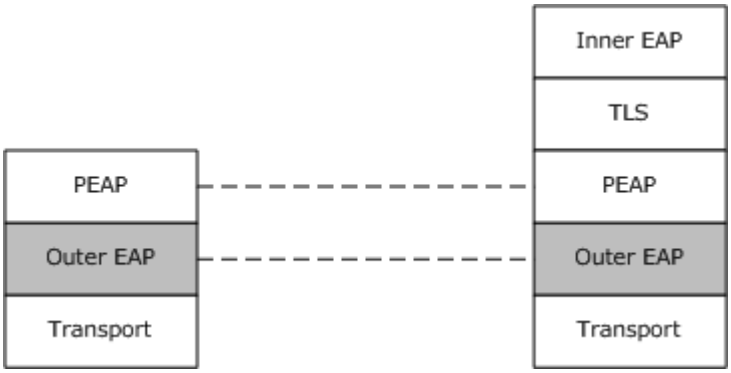

Figure 2: Phase 1 and Phase 2 standalone stack layering

1.5 Prerequisites/Preconditions

Outer EAP or EAPE has all the prerequisites specified in [MS-PEAP] section 1.5 and [MS-RNAP] section 1.5. Furthermore, it MUST comply with the requirements placed on it by [MS-NAPSO].

1.6 Applicability Statement

EAP is an extensible authentication protocol which defines many authentication methods. This document specifies PEAP as its authentication method and is applicable where Network Access Protection as specified by [MS-NAPSO] is implemented.

1.7 Versioning and Capability Negotiation

EAP is an extensible protocol with no versioning or capabilities negotiation of its own.

1.8 Vendor-Extensible Fields

This document does not specify any new vendor-extensible fields but makes use of EAP's expanded types to specify PEAP as its authentication method. Section 5.7 of [RFC3748] specifies the vendor extensible packet format.

1.9 Standards Assignments

Parameter	Value	Reference
PEAP EAP Method	25	[IANA-EAP]
EAP **Type-Length-Value (TLV)** extensions method type (also known as MS-Authentication-TLV)	33	[IANA-EAP]
Vendor-ID for Microsoft® (SMI Private Enterprise Code)	311	[IANA-ENT]

2 Messages

2.1 Transport

Protocols that carry EAP (for example, PPP [RFC1661], IEEE802.1x [IEEE802.1X], and RADIUS [RFC2865]) provide the transport of the associated messages, as specified in [RFC3748] section 3.

2.2 Message Syntax

EAPE does not introduce any new messages. Messages used by EAPE are defined in [RFC3748], [MS-PEAP], [RFC2865], [MS-RNAP], [MS-SOH]. EAPE introduces interfaces that facilitate Network Access Protection.

3 Common Details

3.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

currentState: Initialized when the server starts the EAP authentication and remains valid until the authentication is done. At any point in time, **currentState** can have one of the following integer values, each of which represents a possible state of the EAPE Layer.

- EAP_AUTH_IN_PROGRESS

- EAP_AUTH_SUCCESS

- EAP_AUTH_FAILURE

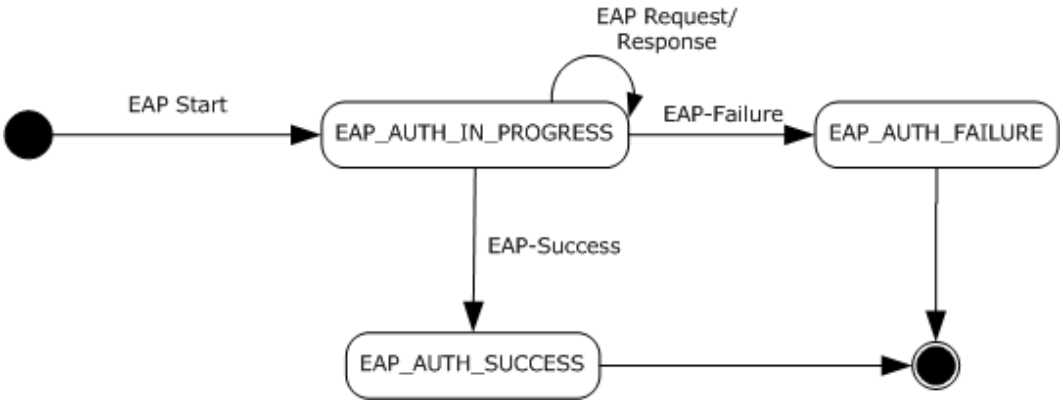

Figure 3: EAP State Machine

3.2 Timers

No other timers other than the EAP timers, as specified in [RFC3748] section 4.3.

3.3 Initialization

Initialization is specified in sections 4.3, 5.3, 6.3

3.4 Higher-Layer Triggered Events

None.

3.4.1 Abstract Interface with PEAP

OnEAPPacket: An abstract interface for EAP Messages. The interface is defined as follows.

```
OnEAPPacket([in] EAPPacket);
```

```
struct EAPPacket
{
Byte Code;
Byte Identifier;
Word Length;
Byte Type;
Byte[] Type-Data;
}
```

EAPPacket: As specified by [RFC3748] sections 4, 4.1 and 4.2.

3.5 Message Processing Events and Sequencing Rules

3.5.1 General Packet Validation

EAPE MUST validate all fields of EAPE header and only supports PEAP as the inner authentication method.

3.6 Timer Events

None

3.7 Other Local Events

None

4 EAPE Client Details

4.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

As specified in [RFC3748] and section 3.1.1

4.2 Timers

As specified in [RFC3748] section 4.3.

4.3 Initialization

EAPE is initialized when the user selects an inner EAP Authentication Method such as PEAP.

4.4 Higher-Layer Triggered Events

None.

4.5 Message Processing Events and Sequencing Rules

4.5.1 Link Establishment

After the transport layer establishes the link, the device acting as an EAPE Client represented by the NAP Capable Client in Figure 1, will sent and EAP-Start message.

4.5.2 Validating an EAP Packet

When the client receives an EAP Packet it must validate the Code, Identifier, and Type as specified in [RFC3748].

4.5.3 Receiving an EAP-Request

When a client receives an EAP-Request message it will validate the packet as given in section 4.5.2 and if the packet is a valid EAP packet it will process the packet based on its Type. If the EAP Type is set to 25, then raise the event **OnPEAPPacket** and consequently PEAP will call the abstract interface with PEAP specified in section 4.1.4.1.

A NAP Capable Client that sent a SoH during the Phase 2 authentication will receive a SoHR which the inner EAP specified in [MS-CEAP] will pass to the NAPSO interface specified by [MS-NAPSO], section 5.

4.5.4 Receiving an EAP-Success

When EAPE receives an EAP-Success, the EAPE client has been authenticated by the inner authentication method and EAPE Phase 2 has been completed.

4.5.5 Receiving an EAP-Failure

When EAPE receives an EAP-Failure, the EAPE client has failed authentication and will not be allowed on the network.

4.6 Timer Events

As specified in [RFC3748] section 4.3.

4.7 Other Local Events

None

5 EAPE Server Details

5.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

As specified in [RFC3748] and section 3.1.1

5.2 Timers

As specified in [RFC3748] section 4.3

5.3 Initialization

As specified in [RFC3748] section 5.1

5.4 Higher-Layer Triggered Events

None.

5.4.1 Sending an message to NAP Capable RADIUS Server

The CEAP server invokes this interface (TBD) to forward EAP Messages, on a NAP Capable Server to the RNAP server.

TBD

The EAPE server then encapsulates the CEAP packet (with the embedded SOHR message) and forwards it to the RNAP server by invoking the RNAP-EAPE abstract interface SendRadiusEAPReply as described in [MS-RNAP] section 3.2.4.3.

The interface is defined as follows.

SendRadiusEAPReply:

```
HRESULT SendRadiusEAPReply (
    [in] RMsgType radiusMsgType,
    [in] GUID24 HCEP-RAS-Correlation-Id,
    [in] String EAPMsgBlob,
    [in] int EAPMsgBlobLen,
    [in] DWORD quarantineState,
    [in] DWORD extendedQuarantineState,
    [in] IPv4Filter ipv4Filter,
    [in] IPv6Filter ipv6Filter,
    [in] DWORD quarantineSessionTimeout,
    [in] DWORD quarantineGraceTime,
    [in] IPv4Address[] ipv4RemediationServers,
    [in] IPv6Address[] ipv6RemediationServers,
```

```
    [in] String dhcpQuarantineUserClass,
    [in] DWORD rdgDeviceRedirection,
    [in] DWORD afwZone,
    [in] DWORD afwProtectionLevel
);
```

The following table shows the mapping of the incoming CEAP parameters to the parameters of the **SendRadiusEAPReply** interface.

TBD.

5.5 Message Processing Events and Sequencing Rules

5.5.1 Receiving a message from NAP Capable RADIUS Server

ProcessEAPRequest: An abstract interface for forwarding EAP Messages, on a NAP Capable Server from the RNAP server to the EAPE Layer. The interface is defined as follows.

```
HRESULT ProcessEAPRequest (
[in] String eapBlob,
[in] DWORD eapBlobLength,
[in] String eapBlobSignature,
[in] DWORD eapBlobSigLength,
[in] DWORD NotQuarantineCapable,
[in] String ClientName,
[in] String MachineName,
[in] DWORD ClientIPv4Address,
[in] ByteArray ClientIPv6Address,
[in] SID SecurityIdentifier,
[in] DWORD CallingStationID,
[in] DWORD NetworkAccessServerType,
[in] DWORD TunnelType,
[in] DWORD NASPortType,
[in] DWORD NASIPv4Address,
[in] ByteArray NASIPv6Address,
[in] String HCAPLocationGroup,
[in] String HCAPUserGroups,
[in] String HCAPUserName
);
```

Unless otherwise specified, all multibyte elements are in host-byte order.

eapBlob is a binary blob to be de-encrypted. It contains the SOH.

eapBlobLength the length of the eapBlob.

eapBlobSignature contains the hash signature of the eapBlob.

eapBlobSigLength the length of the eapBlobSignature.

The rest of the parameters are simply passed up to the CEAP layer to be processed by the NAPSO layer.

NotQuarantineCapable signifies whether the client is NAP aware.

ClientName the value of the MS-RAS-Client-Name attribute.

MachineName the value of the MS-Machine-Name attribute.

ClientIPv4Address the value of the MS-UserIPv4-Address attribute.

ClientIPv6Address the value of the MS-UserIPv6-Address attribute.

DhcpServiceClass the value of the MS-Service-Class attribute.

NASIdentifier the value of the NAS-Identifier attribute.

SecurityIdentity the value of the MS-User-Security-Identity attribute.

CallingStationID the value of the Calling-Station-ID attribute.

NetworkAccessServerType the value of MS-Network-Access-Server-Type attribute.

TunnelType the value of the Tunnel-Type attribute.

NASPortType the value of the NAS-Port attribute.

NASIPv4Address the value of the NAS-IP-Address attribute.

NASIpv6Address the value of NAS-IPv6-Address attribute.

HCAPLocationGroup the value of the HCAP-Location-Group attribute.

HHCAPUserGroup the value of HCAP-User-Groups attribute.

Remarks:

Note that the correlation ID is included in the SoH as specified in [MS-SOH]. The authentication is independent of the SoH. It is handled by the underlying RADIUS protocol implementation and is outside the scope of this document.

The EAPE server then forwards the EAP blob and other attributes to the CEAP layer by invoking the EAPE-CEAP interface (TBD).

5.6 Timer Events

As specified in [RFC3748] section 4.3

5.7 Other Local Events

6 EAPE Pass-through Proxy Details

6.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

As specified in [RFC3748] and section 3.1.1

6.2 Timers

As specified in [RFC3748] section 4.3

6.3 Initialization

As specified in [RFC3748] section 5.1

6.4 Higher-Layer Triggered Events

None.

6.4.1 Interface with NAPSO

The interface with NAPSO on the NAP Capable Pass-through Proxy allows the EAPE Layer to pass messages to the RADIUS Layer. The result is the exchange of EAPE messages from the client to the server.

6.4.1.1 Forward a Message to the EAPE Client

SendEAPReply: An abstract interface for forwarding an EAP Message, on a NAP Capable Pass-through Proxy, to the EAPE Client. The interface is defined as follows.

```
HRESULT SendEAPReply (
    [in] EAPMsgType eapMsgType,
    [in] String EAPMsgBlob,
    [in] int EAPMsgBlobLen,
    [in] IPv4Filter ipv4Filter,
    [in] IPv6Filter ipv6Filter,
    [in] IPv4Address[] ipv4RemediationServers,
    [in] IPv6Address[] ipv6RemediationServers,
);
```

Unless otherwise specified, all multibyte elements are in host-byte order.

The following abstract type definitions are common to several parameters.

```
enum LateBoundFlags
{
    NoReplacement = 0x00,
    ReplaceSourceAddress = 0x01,
    ReplaceDestinationAddress = 0x04,
```

```
    ReplaceSourceMaskOrPrefix = 0x10,
    ReplaceDestinationMaskOrPrefix = 0x20,
}

enum ForwardAction
{
    Forward = 0,
    Drop = 1
}

struct IPv4Address
{
    Byte[4] bytes;
}

struct IPv6Address
{
    Byte[16] bytes;
}
```

The remainder of this section describes the individual parameters of the SendRadiusAccessAccept abstract interface.

eapMsgType: An enum describing the RADIUS message type to send.

```
enum EAPMsgType
{
    EAP-Request = 1,
    EAP-Response = 2,
}
```

EAPMsgBlob: Opaque binary data that specifies the EAP message to be put into the EAP-Message attribute. The RADIUS EAP-Message attribute is type number 79 and is described in [RFC3579] section 3.1

EAPMsgBlobLen: The length of the EAPMsgBlob parameter.

ipv4Filter: An object of an IPv4Filter abstract type that specifies the network access scope of the endpoint. The IPv4Filter abstract type is defined as follows.

```
struct Filter_IPv4
{
    IPv4Address sourceAddress;
    IPv4Address sourceMask;
    IPv4Address destinationAddress;
    IPv4Address destinationMask;
    DWORD protocol;
    DWORD lateBoundFlags; // Bitwise combination of LateBoundFlags values
}

struct FilterSet
{
    ForwardAction forwardAction;
    Filter_IPv4[] filters;
}
```

```
enum InfoType_IPv4
{
    InputFilter = 0xffff0001,
    OutputFilter = 0xffff0002,
    SiteToSiteConnection = 0xffff0009
}

srtuct FilterSetEntry_IPv4
{
    InfoType_IPv4 infoType;
    FilterSet_IPv4[] filterSets;
}

struct IPv4Filter
{
    FilterSetEntry_IPv4[] filterEntries;
}
```

ipv6Filter: An object of an IPv6Filter abstract type that specifies the network access scope of the endpoint. The IPv6Filter abstract type is defined as follows.

```
struct Filter_IPv6
{
    IPv6Address sourceAddress;
    DWORD sourcePrefixLength;
    IPv6Address destinationAddress;
    DWORD destinationPrefixLength;
    DWORD protocol;
    DWORD lateBoundFlags; // Bitwise combination of LateBoundFlags values
}

struct FilterSet_IPv6
{
    ForwardAction forwardAction;
    Filter_IPv6[] filters;
}

enum InfoType_IPv6
{
    InputFilter = 0xffff0011,
    OutputFilter = 0xffff0012
}

srtuct FilterSetEntry_IPv6
{
    InfoType_IPv6 infoType;
    FilterSet_IPv6[] filterSets;
}

struct IPv6Filter
{
    FilterSetEntry_IPv6[] filterEntries;
}
```

ipv4RemediationServers: An array of objects of the IPv4Address abstract type (defined earlier in this section) that specifies the addresses of available IPv4 remediation servers.

ipv6RemediationServers: An array of objects of the IPv6Address abstract type (defined earlier in this section) that specifies the addresses of available IPv6 remediation servers.

6.4.1.2 Forward a Message to EAPE Server

SendEAPRequest: An abstract interface for forwarding an EAP Message on a NAP Capable Pass-through Proxy. The interface is defined as follows.

```
HRESULT SendEAPRequest (
    [in] EMsgType eapMsgType,
    [in] String EAPMsgBlob,
    [in] int EAPMsgBlobLen,
    [in] IPv4Filter ipv4Filter,
    [in] IPv6Filter ipv6Filter,
    [in] IPv4Address[] ipv4RemediationServers,
    [in] IPv6Address[] ipv6RemediationServers,
);
```

Unless otherwise specified, all multibyte elements are in host-byte order.

The following abstract type definitions are common to several parameters.

```
enum LateBoundFlags
{
    NoReplacement = 0x00,
    ReplaceSourceAddress = 0x01,
    ReplaceDestinationAddress = 0x04,
    ReplaceSourceMaskOrPrefix = 0x10,
    ReplaceDestinationMaskOrPrefix = 0x20,
}

enum ForwardAction
{
    Forward = 0,
    Drop = 1
}

struct IPv4Address
{
    Byte[4] bytes;
}

struct IPv6Address
{
    Byte[16] bytes;
}
```

The remainder of this section describes the individual parameters of the SendRadiusAccessAccept abstract interface.

eapMsgType: An enum describing the RADIUS message type to send.

```
enum EMsgType
{
    EAP-Request = 1,
    EAP-Response = 2,
```

```
}
```

EAPMsgBlob: Opaque binary data that specifies the EAP message to be put into the EAP-Message attribute. The RADIUS EAP-Message attribute is type number 79 and is described in [RFC3579] section 3.1

EAPMsgBlobLen: The length of the EAPMsgBlob parameter.

ipv4Filter: An object of an IPv4Filter abstract type that specifies the network access scope of the endpoint. The IPv4Filter abstract type is defined as follows.

```
struct Filter_IPv4
{
    IPv4Address sourceAddress;
    IPv4Address sourceMask;
    IPv4Address destinationAddress;
    IPv4Address destinationMask;
    DWORD protocol;
    DWORD lateBoundFlags; // Bitwise combination of LateBoundFlags values
}

struct FilterSet
{
    ForwardAction forwardAction;
    Filter_IPv4[] filters;
}

enum InfoType_IPv4
{
    InputFilter = 0xffff0001,
    OutputFilter = 0xffff0002,
    SiteToSiteConnection = 0xffff0009
}

srtuct FilterSetEntry_IPv4
{
    InfoType_IPv4 infoType;
    FilterSet_IPv4[] filterSets;
}

struct IPv4Filter
{
    FilterSetEntry_IPv4[] filterEntries;
}
```

ipv6Filter: An object of an IPv6Filter abstract type that specifies the network access scope of the endpoint. The IPv6Filter abstract type is defined as follows.

```
struct Filter_IPv6
{
    IPv6Address sourceAddress;
    DWORD sourcePrefixLength;
    IPv6Address destinationAddress;
    DWORD destinationPrefixLength;
    DWORD protocol;
    DWORD lateBoundFlags; // Bitwise combination of LateBoundFlags values
```

```
}

struct FilterSet_IPv6
{
    ForwardAction forwardAction;
    Filter_IPv6[] filters;
}

enum InfoType_IPv6
{
    InputFilter = 0xffff0011,
    OutputFilter = 0xffff0012
}

srtuct FilterSetEntry_IPv6
{
    InfoType_IPv6 infoType;
    FilterSet_IPv6[] filterSets;
}

struct IPv6Filter
{
    FilterSetEntry_IPv6[] filterEntries;
}
```

ipv4RemediationServers: An array of objects of the IPv4Address abstract type (defined earlier in this section) that specifies the addresses of available IPv4 remediation servers.

ipv6RemediationServers: An array of objects of the IPv6Address abstract type (defined earlier in this section) that specifies the addresses of available IPv6 remediation servers.

6.5 Message Processing Events and Sequencing Rules

6.5.1 Link Establishment

After the transport layer established the link, the device acting as a Pass-Through Authenticator will send the EAP Client and EAP-Request/Identity message specified in [RFC3748] section 5.

6.5.2 Receiving an EAP-Response

When the EAPE Layer on the Pass-through Proxy receives an EAP-Response message it will call the interface with NAPSO 2, shown in Figure 1, named **SendEAPRequest** and use EAPMsgBlob, and EAPMsgBlobLen fields to pass the unprocessed message to NAPSO, which will facilitate the exchange of messages from the Client to the Server, through the use of NAP capable RADIUS Client.

EAPMsgBlob MUST contain the EAP Message. The EAPMsgBlobLen will be the length of the message. EAPMsgBlob MUST include the original EAPE header which it was received with on the proxy. NasServerType MUST be set to "EAPE". All other fields are not used.

6.5.3 Receiving a RADIUS Access-Challenge

Upon receipt of a RADIUS Access-Challenge containing an EAP-Message the RADIUS Client Layer on the Pass-Through Proxy will pass it to the interface with NAPSO 3, shown in Figure 1. Upon receipt of this message destined for the Client the NAPSO component will use the interface with NAPSO 2, from Figure 1, to pass the message to the EAPE Layer by calling **SendEAPReply**.

EAPMsgBlob will contain the EAP Message with the EAPE header and the EAPMsgBlobLen will be the length of the message. All other fields are not used.

6.6 Timer Events

None

6.7 Other Local Events

7 Protocol Examples

See [MS-PEAP].

8 Security

8.1 Security Considerations for Implementers

See [MS-PEAP].

9 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:

- Microsoft Windows NT® operating system
- Microsoft 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

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.

10 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

11 Index