[MS-DPSP]:
Digest Protocol Extensions

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

Date	Revision History	Revision Class	Comments
06/01/2007	1.3.1	Editorial	Revised and edited the technical content.
07/03/2007	1.4	Minor	Updated technical content.
07/20/2007	1.4.1	Editorial	Revised and edited the technical content.
08/10/2007	1.4.2	Editorial	Revised and edited the technical content.
09/28/2007	1.4.3	Editorial	Revised and edited the technical content.
10/23/2007	1.4.4	Editorial	Revised and edited the technical content.
11/30/2007	2.0	Major	Updated Abstract Data Model sections; clarified Windows behavior.
01/25/2008	2.0.1	Editorial	Revised and edited the technical content.

Table of Contents

1 Introduction

The Windows implementation of the Digest Authentication Protocol contains variations from the Digest Authentication standard specified in [RFC2617] and [RFC2831].

Digest authentication supports client authentication to servers (based on the user's name and password) and server authentication to the client.

Windows implements the digest access authentication for HTTP/1.1 as specified in [RFC2617]. Windows also implements digest authentication as a Simple Authentication and Security Layer (SASL) mechanism, as specified in [RFC2831].

Higher-Layer protocols such as Lightweight Directory Access Protocol (LDAP) ([RFC2251]) employ digest authentication as an SASL mechanism. The Windows implementation is compliant with digest authentication, as specified in [RFC2617] and [RFC2831].

This protocol is also how Windows implements optional fields and behaviors (specified by keywords such as MAY or SHOULD) and how Windows implements support for older clients and servers that exhibit nonconforming behavior to [RFC2617] and [RFC2831].

1.1 Glossary

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

Keyed Hash
Nonce

The following terms are specific to 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.

[FIPS140] National Institute of Standards and Technology, "Federal Information Processing Standards Publication 140-2: Security Requirements for Cryptographic Modules", December 2002, http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

[MS-DTYP] Microsoft Corporation, "Windows Data Types", January 2007.

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[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

[RFC2222] Myers, J., "Simple Authentication and Security Layer (SASL)", RFC 2222, October 1997, http://www.ietf.org/rfc/rfc2222.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

[RFC2252] Wahl, M., Coulbeck, A., Howes, T., and Kille, S., "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions", RFC 2252, December 1997, http://www.ietf.org/rfc/rfc2252.txt

[RFC2616] Fielding, R., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and Stewart, L., "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999, http://www.ietf.org/rfc/rfc2617.txt

[RFC2829] Wahl, M., Alvestrand, H., Hodges, J., and Morgan, R., "Authentication Methods for LDAP", RFC 2829, May 2000, http://www.ietf.org/rfc/rfc2829.txt

[RFC2831] Leach, P. and Newman, C., "Using Digest Authentication as a SASL Mechanism", RFC 2831, May 2000, http://www.ietf.org/rfc/rfc2831.txt

1.2.2 Informative References

[MS-APDS] Microsoft Corporation, "Authentication Protocol Domain Support Specification", June 2007.

[MS-KILE] Microsoft Corporation, "Kerberos Protocol Extensions", January 2007.

[RFC2069] Franks, J., et al., "An Extension to HTTP: Digest Access Authentication", RFC 2069, January 1997, http://www.ietf.org/rfc/rfc2069.txt

[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

1.3 Protocol Overview (Synopsis)

The digest authentication mechanism [RFC2617] [RFC2831] performs authentication between a client and a server based on a user name and a password. The digest authentication protocol can authenticate the client to the server, and optionally the server to a client; the latter is termed mutual authentication.

In the digest authentication mechanism, the client is presented with a **nonce**, a randomly generated value sent by the server to the client. The client proves knowledge of the password by computing a **keyed hash** over parameters sent by the server and parameters generated locally by the client.

Once the server has validated the keyed hash by performing the same computation, it can authenticate itself to the client by computing another keyed hash and returning it to the client. Because the correct keyed hash results can only be created by someone who knows the password, the two parties are assured of the other knowing the password.

Subsequent client-to-server messages can be authenticated by a keyed hash. Replay protection is provided via an ordered nonce count, as specified in [RFC2831]. Digest authentication as a SASL mechanism expands the digest access authentication protocol by also supporting integrity and confidentiality of messages sent between the client and server. For more information on SASL, see [RFC2831]. For more information on digest access authentication, see [RFC2617].

1.4 Relationship to Other Protocols

The Digest Authentication Protocol was originally specified as a native authentication method for HTTP/1.1, as specified in [RFC2616], to serve as an improvement on the HTTP basic authentication. The popularity of digest authentication grew, and it was covered as an SASL [RFC2222] mechanism by the specification in [RFC2831]. Once made into an SASL mechanism, the Digest Authentication Protocol became available for other protocols, such as LDAP, as specified in [RFC2251].<1>

1.5 Prerequisites/Preconditions

Digest Protocol Extensions assumes the following:

1. Prior to the start of digest authentication, the client and the server have access to the user's password (shared knowledge between them).

2. On the server and the client, a source of cryptographically useful random numbers is available for generating a nonce.<2>

1.6 Applicability Statement

The digest authentication mechanism is used in environments that require users to authenticate to servers to access secure resources. Note that Kerberos (for more information, see [MS-KILE]) and public key–based authentication offer stronger security guarantees both in terms of initial authentication and subsequent confidentiality and integrity of client/server traffic.

The digest authentication mechanism can be used in environments where these stronger mechanisms are not available and can serve for interoperability purposes with multiple vendors, browsers, Web servers, and directory services.<3>

1.7 Versioning and Capability Negotiation

Neither the Digest Authentication Protocol nor the Digest Protocol Extensions have any versioning capability. The Digest Authentication Protocol does have support for negotiating what cryptographic algorithms to use. This is specified in [RFC2831] section 2.1.2.

1.8 Vendor-Extensible Fields

The Digest Protocol Extensions introduce no additional vendor-extensible fields beyond those specified in [RFC2831] and [RFC2617].<4>

1.9 Standards Assignments

There are no standards assignments in the Digest Protocol Extensions beyond what is specified in [RFC2617] and [RFC2831].

2 Messages

2.1 Transport

The elements of the Digest Protocol Extensions messages are embedded directly in HTTP/1.1 messages [RFC2616] when digest authentication is used within HTTP, as specified in [RFC2617]. As a result, Digest Protocol Extensions uses the HTTP/1.1 transport, as specified in [RFC2616].

As specified in [RFC2831], Digest Protocol Extensions messages are carried via SASL [RFC2222] in SASL-aware protocols such as LDAP.

2.2 Message Syntax

The message syntax of the Digest Protocol Extensions is as specified in [RFC2617] and [RFC2831].<5>

The extension to this protocol consists of the use of the capability specified in [RFC2617] to add a new directive via the auth-param in the digest-challenge message ([RFC2831] section 2.1.1). The server sends: charset=utf-8 in the digest-challenge message. This indicates that the server can process UTF-8 encoded strings and that the client might use [UNICODE] encoding for the **username** field and in the password if it can also process UTF-8. Clients will use [UNICODE] encoding when it is offered by the server to allow authentication with a region's supported character sets.<6>

3 Protocol Details

The following sections specify details of the Digest Protocol Extensions, including abstract data models and message processing rules that are common for both the client and the server. The variations are as specified in [RFC2617] and [RFC2831].

3.1 Common Details

3.1.1 Abstract Data Model

The abstract data model follows what is specified in [RFC2617] and [RFC2831]. In addition, there is the following variable:

Integrity: When this value is set, it indicates that the higher-layer protocol requires integrity in addition to simple authentication. This corresponds to the auth-int option, as specified in [RFC2617] sections 3.2.1 and 3.2.2. If this is not set, only auth is specified as the requested quality of protection.

3.1.2 Timers

There are no common timers specified in the Digest Protocol Extensions.

3.1.3 Initialization

The random-number generator is initialized for nonce and key generation.<7>

3.1.4 Higher-Layer Trigger Events

Digest Protocol Extensions are triggered by a higher-layer application protocol, such as when HTTP or LDAP create a connection and require authentication. The higher-layer protocol determines if the **Integrity** option is enabled for a particular connection.<8>

3.1.5 Message Processing and Sequencing

3.1.5.1 Authentication-Info

Specifying fields in the Authentication-Info message sent on the third leg of the Digest Protocol Extensions from the server to the client SHOULD be supported, as specified in [RFC2617] section 3.2.3.<9>

3.1.5.2 Realm Directive

The realm directive is optional; if not present, the client SHOULD solicit it from the user or be able to compute a default, as specified in [RFC2831] section 2.1.1.<10>

3.1.5.3 Subsequent Authentication

As specified in [RFC2831] section 2.2, the client performs "subsequent authentication" on the following conditions:<11>

- The client has previously authenticated to the server.
- Both the client and server possess the values of username, realm, nonce, nonce-count, cnonce, and qop that the client used in that authentication.

- The SASL profile for a protocol permits an initial client response.

3.1.5.4 Opaque Directive

The opaque directive is a string of data that SHOULD be returned by the client unchanged in the authorization header, as specified in [RFC2617] section 3.2.1.<12>

3.1.6 Timer Events

There are no common timer events specified in the Digest Protocol Extensions.

3.1.7 Other Local Events

There are no additional local events specified in the Digest Protocol Extensions.

3.2 Client Details

3.2.1 Abstract Data Model

The client must maintain a nonce for ongoing communications with the server, as specified in [RFC2617] section 3.2.2 and [RFC2831] section 2.1.2. The client maintains the state of the nonce by keeping track of the nonce count, which is incremented each time the client sends a message to the server.<13>

3.2.2 Timers

There are no additional client-specific timers specified in the Digest Protocol Extensions.

3.2.3 Initialization

For details on initialization for the client, see section 3.1.3.

3.2.4 Higher-Layer Trigger Events

For details on higher-layer trigger events for the client, see section 3.1.4.

3.2.5 Message Processing and Sequencing

3.2.5.1 Client Nonce Generation

It is highly recommended that when the **cnonce** is generated by the client, it contains at least 64 bits of entropy, as specified in [RFC2831] section 2.1.2.<14>

3.2.5.2 Qop Directive

The qop directive, as specified in [RFC2617] section 3.2.2, is optional to preserve backward compatibility with minimal implementation of digest access authentication, as specified in [RFC2069]. [RFC2617] specifies that the qop directive SHOULD be used by the client (if the server indicates that qop is supported) by providing a qop directive in the WWW-Authenticate header field.<15>

3.2.6 Timer Events

There are no additional client-specific timer events specified in the Digest Protocol Extensions.

3.2.7 Other Local Events

There are no additional client-specific local events specified in the Digest Protocol Extensions.

3.3 Server Details

3.3.1 Abstract Data Model

The server must associate state with each authenticated connection, as specified in [RFC2617]. Specifically, the server must associate a client and a server nonce with each connection, along with a nonce count ([RFC2617] section 3.2.2 and [RFC2831] section 2.1.2) for ongoing communications. The server may keep all this information longer than an active connection, depending on the length of time that is allotted for subsequent authentication, as specified in [RFC2617] section 3.3.

3.3.2 Timers

There are no additional server-specific timers specified in the Digest Protocol Extensions.

3.3.3 Initialization

For details on initialization for the server, see section 3.1.3.

3.3.4 Higher-Layer Trigger Events

For details on the higher-layer trigger events for the server, see section 3.1.4.

3.3.5 Message Processing and Sequencing

3.3.5.1 Server Nonce Generation

It is recommended that the nonce computed by the server contain at least 64 bits of entropy, as specified in [RFC2831] section 2.1.1.<16>

3.3.5.2 Qop-Options Directive

The qop-options directive, as specified in [RFC2617] section 3.2.1, is optional; but it is used for backward compatibility with digest access authentication, as specified in [RFC2069]. The qop-options directive SHOULD be used by all implementations compliant with this version of the digest authentication mechanism.<17>

3.3.5.3 Realm Directive for the Digest Challenge

The realm directive is required if the server provides any realms in the digest challenge; in which case it may appear exactly once, and its value SHOULD be one of those realms, as specified in [RFC2831] section 2.1.2.<18>

3.3.5.4 Principal Name Validation

Digest-Uri indicates the principal name of the service that the client is attempting to connect with, as specified in [RFC2831] section 2.1.2. Servers SHOULD check that the supplied value is correct.<19>

3.3.5.5 Host Name

The Digest Protocol Extensions do not make use of the host field.

3.3.6 Timer Events

There are no additional server-specific timer events specified in the Digest Protocol Extensions.

3.3.7 Other Local Events

There are no additional server-specific local events specified in the Digest Protocol Extensions.

4 Protocol Examples

The following diagram and procedural steps describe a common scenario to illustrate the function of the Digest Protocol Extensions.

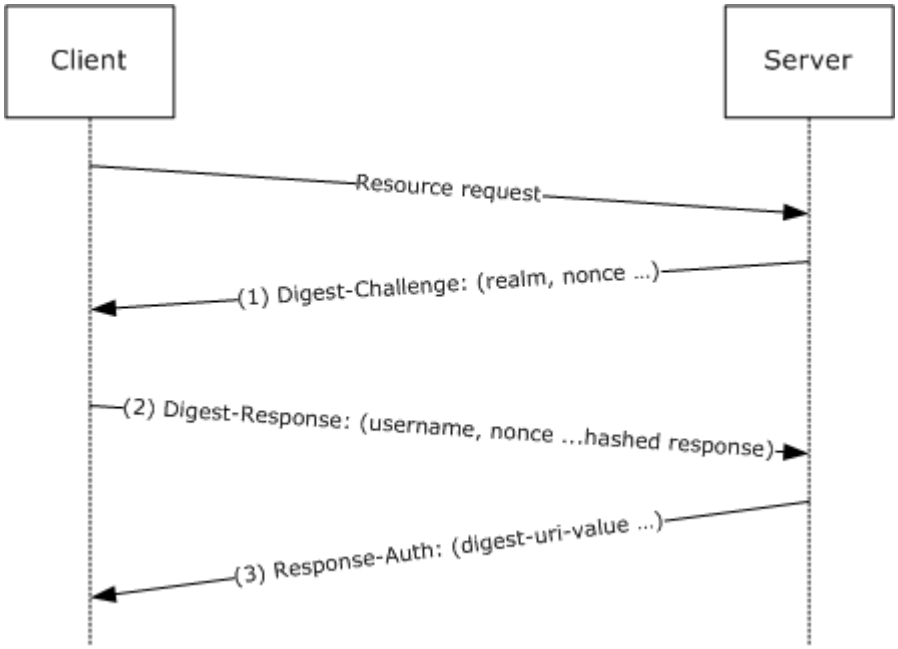

Figure 1: Common use scenario for Digest Protocol Extensions

1. After the client attempts to access a protected resource (for example, ResourceA) on the server, the server returns a digest-challenge message to the client. Among other fields, the digest-challenge message includes a randomly generated nonce and the domain name of the server (via the **realm** field). A sample digest Challenge is shown here:

```
qop="auth",algorithm=MD5-sess,
nonce="91c121b4f47ec601a281ceefaa5d6f2b096897ea0797fdd2ea72bfeec7fd
a64433a98d4ae57186a1",
charset=utf-8,realm="TestDomain"
```

2. The client obtains the user name (for example, User123) and password for the user and constructs a response to the server's challenge. In the digest-response, the client proves knowledge of the user's password by performing a keyed hash over the user name, nonce, and other fields (the password is fed into the hash). A sample digest-response is shown here:

```
username="User123",
realm="TestDomain",
qop="auth",
algorithm="MD5-sess",
uri="/ResourceA",
nonce="91c121b4f47ec601a281ceefaa5d6f2b096897ea0797fdd2ea72bfeec7fd
a64433a98d4ae57186a1",
nc=00000001,
cnonce="579c5e7723ad0ef8eeb0b7427379bdd4",
```

```
response="641b92d2d8af170329ce308832a4df13"
```

3. The server validates the digest-response message by looking up the user's password by using the user name that the client sent, recomputing the keyed hash over fields from the digest-response message, and then comparing the resulting hash value to the Response directive value sent by the client. If the values match, the client's digest-response message is valid; otherwise, the authentication request fails. The server further checks that the client sent the expected nonce and nonce-count values (and not an old, replayed value). After the digest response is validated, the client is authenticated to the server. For mutual authentication, the server has the option to send a keyed hash over the URI that the client requested and return it to the client in the Response-Auth message. Note that sending the Response-Auth message only applies to digest authentication when used as an SASL mechanism, as specified in [RFC2831]. For further information, see section 3.1.5.1.

5 Security

5.1 Security Considerations for Implementers

Kerberos (for more information, see [MS-KILE]) and public key-based authentication offer stronger security guarantees both in terms of initial authentication and in subsequent confidentiality and integrity of client-server traffic. The digest authentication mechanism can be used in environments where these stronger mechanisms are not available.

5.2 Index of Security Parameters

There are no security parameters specified in the Digest Protocol Extensions.

6 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows NT
- Windows 2000
- Windows XP
- Windows Server 2003
- Windows Vista

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

<1> Section 1.4: In Windows domain environments, validating the digest authentication exchange can be performed at a different computer (for more information, see [MS-APDS]). This does not affect the actual implementation of the digest authentication between the client and the server. For more information on digest authentication validation, see [RFC2617] section 3.2.2.

<2> Section 1.5: The random number generator is FIPS-140-compliant, as specified in [FIPS140] .

<3> Section 1.6: In Windows environments, the Digest Authentication Protocol can be used for authentication during HTTP traffic between browsers and Internet Information Services (IIS). In addition, LDAP clients and servers in an Active Directory domain can make use of the SASL mechanism for digest authentication.

By default, IIS does not offer digest authentication as an authentication method, but it can be configured to do so. When configured, IIS offers digest authentication as specified in [RFC2616] section 14.47. Internet Explorer uses digest authentication, but only if the preferred methods such as the SPNEGO Protocol Extensions or the NTLM Authentication Protocol are not available.

Windows NT IIS does not support digest authentication.

Active Directory's LDAP server offers digest authentication by default as one of several authentication mechanisms for Active Directory. This is exposed by the supported SASL Mechanism attribute, as specified in [RFC2829]. All native use of LDAP within Windows uses SPNEGO authentication, and not digest authentication.

<4> Section 1.8: Windows uses the capability specified in [RFC2617] to add a new directive via the auth-param in the digest-challenge message ([RFC2831] section 2.1.1). The server sends: charset=utf-8 in the **auth-param** field. This indicates that the server can process UTF-8 encoded strings and that the client might use Unicode encoding for the **username** field and in the password if it can also process UTF-8. Windows clients will use [UNICODE] encoding when it is offered by the server to allow authentication with a region's supported character sets.

<5> Section 2.2: There are two instances in the Digest Authentication Protocol where the Windows XP and Windows Server 2003 implementations allow a non-RFC compliant message syntax.

The first instance in which the Windows XP and Windows Server 2003 implementations allow non-RFC compliant message syntax pertains to the processing of the **username** field. Windows, when

acting as a server, first attempts to decode the **username** field of the digest-response message ([RFC2617] section 3.2.2) using the RFC-compliant character escaping ([RFC2616] section 2.2).

However, if no account with that name can be found, or if the keyed hash computed by the server does not match the keyed hash sent by the client, the server attempts to decode the original username by using backslash '\' characters that are not properly escaped (treated as if the client specified '\\'), and then retries. Windows NTand Windows XP incorrectly escape the backslash '\' character in the account name. If a backslash ('\') character is presented by the client, the string is treated as "NetbiosDomainName\\AccountName".

Unlike other authentication protocols, digest authentication does not support the diacritical folding that is applied by Active Directory. This is because digest authentication uses the username as submitted by the client for creating the key. If two user names map to the same account for digest authentication, authentication fails for any version of the name other than the "true" value associated with the account.

The second instance in which the Windows XP and Windows Server 2003 implementations allow non-RFC compliant message syntax pertains to additional flexibility in processing the QOP field. Digest authentication supports sending a variety of options, as specified in [RFC2617], including the Quality of Protection (QOP) field. As specified in [RFC2617] section 3.2.1, the values of the QOP operand must be quoted; however, as specified in [RFC2617] section 3.2.2 (the reply), the selected value does not need to be quoted.

For single QOP values, Windows does not require the QOP value to be quoted when it is received either at the client or the server. Therefore, it allows the operand to be sent with or without quotations from either the client or server. That is, on receipt of the message, Windows treats 'qop="value"' the same as 'qop=value'.

Windows does require that a multiple value QOP field be correctly quoted; that is, it must comply with the RFC and be presented as 'qop="value1,value2"'. Windows XP, Windows Server 2003, and Windows Vista versions of Windows have a setting to force single QOP value strings to be emitted with quotes (with the default setting to use quotes). This is for compatibility with Windows 2000.

<6> Section 2.2: The auth-param extension applies to Microsoft implementations starting with Windows XP and Windows Server 2003.

<7> Section 3.1.3: The random number generator for keys and nonces is initialized by other components, but complies with what is specified in [FIPS140].

<8> Section 3.1.4: For details on where exactly digest authentication is used in the Windows environment, see section 1.6.

<9> Section 3.1.5.1: The Windows 2000, Windows XP, and Windows Server 2003 digest implementations for HTTP do not support the Authentication-Info message. If a third-party server generates an Authentication-Info message, it will be ignored on the Windows client. However, the third leg of the Digest Protocol Extensions as the SASL mechanism is supported, as specified in [RFC2831] section 2.1.3.

<10> Section 3.1.5.2: The realm directive is used in Windows 2000, Windows XP, and Windows Server 2003. The realm is set to the DNS domain name of the server by default. This can be overridden at the server by configuration options. The user is prompted and has the chance to override the realm value sent by the server (that is, the user can enter a realm other than the one sent by the server).

<11> Section 3.1.5.3: Windows servers accept subsequent authentication for the client connections they have cached. The lifetime of the digest session is controlled by the calling

application. The Windows digest component does not implement any expiration for the digest session because none is specified in [RFC2617] or [RFC2831].

<12> Section 3.1.5.4: For Windows NT and Windows 2000, the opaque value is returned from the client to the server only during the initial authentication step. They do not return the opaque value for subsequent authentication ([RFC2617] section 3.3). Windows Server 2003 complies with [RFC2617] in its use of the opaque directive. Windows XP and Windows Vista return the opaque value for both initial and subsequent authentications.

<13> Section 3.2.1: The Windows digest implementation ensures that if the nonce count value wraps, the authentication fails.

<14> Section 3.2.5.1: For Windows XP, the **cnonce** contains 128 bits of entropy by using a random-number generator, as specified in [FIPS140].

<15> Section 3.2.5.2: Windows 2000, Windows XP, and Windows Server 2003: Windows servers indicate support for different qops, and clients select accordingly, using the qop values as specified in [RFC2617] section 3.2.1 (subject to exceptions specified in section 2.2) and in [RFC2831] section 2.1.1. The value of this directive depends on the quality of protection requested by the calling application. For information on Windows usage of the qop directive, see section 2.2.

<16> Section 3.3.5.1: In Windows Server 2003, the nonce contains 128 bits of entropy by using a random-number generator as specified in [FIPS140].

<17> Section 3.3.5.2: Windows 2000, Windows XP, and Windows Server 2003: The qop-options directive is supported as specified in [RFC2617].

<18> Section 3.3.5.3: Windows 2000, Windows XP, and Windows Server 2003 handle the realm directive exactly as specified in this section.

<19> Section 3.3.5.4: The digest implementation does not do any validation of the digest-uri value. The application that calls the digest authentication validates the principal name specified by the digest-uri. The digest authentication implementation returns the principal name to the calling application.

7 Index

S

T

V

W