[MS-CONMGMT]:

Connection Management Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft 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. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **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
4/4/2008	0.1		Initial version
4/25/2008	0.2		Revised and edited the technical content
6/27/2008	1.0		Revised and edited the technical content
8/15/2008	1.01		Revised and edited the technical content
12/12/2008	2.0		Revised and edited the technical content
2/13/2009	2.01		Revised and edited the technical content
3/13/2009	2.02		Revised and edited the technical content
7/13/2009	2.03	Major	Revised and edited the technical content
8/28/2009	2.04	Editorial	Revised and edited the technical content
11/6/2009	2.05	Editorial	Revised and edited the technical content
2/19/2010	2.06	Editorial	Revised and edited the technical content
3/31/2010	2.07	Major	Updated and revised the technical content
4/30/2010	2.08	Editorial	Revised and edited the technical content
6/7/2010	2.09	Editorial	Revised and edited the technical content
6/29/2010	2.10	Editorial	Changed language and formatting in the technical content.
7/23/2010	2.10	No Change	No changes to the meaning, language, or formatting of the technical content.
9/27/2010	3.0	Major	Significantly changed the technical content.
11/15/2010	3.0	No Change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	3.0	No Change	No changes to the meaning, language, or formatting of the technical content.
3/18/2011	3.0	No Change	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	3.0	No Change	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	3.1	Minor	Clarified the meaning of the technical content.
4/11/2012	3.1	No Change	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	3.1	No Change	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	3.1.1	Editorial	Changed language and formatting in the technical content.
2/11/2013	3.1.1	No Change	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
7/30/2013	3.1.1	No Change	No changes to the meaning, language, or formatting of the technical content.
11/18/2013	3.2	Minor	Clarified the meaning of the technical content.
2/10/2014	3.2	No Change	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	3.2	No Change	No changes to the meaning, language, or formatting of the technical content.
7/31/2014	3.2	No Change	No changes to the meaning, language, or formatting of the technical content.
10/30/2014	3.2	No Change	No changes to the meaning, language, or formatting of the technical content.
9/4/2015	3.2	No Change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document specifies the Connection Management Protocol that can be used for a protocol client to automatically discover the address of its Session Initiation Protocol (SIP) outbound proxy, and for maintaining a persistent, reliable, in-order transport between the protocol client and the proxy.

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 [RFC2119]. Sections 1.5 and 1.9 are also normative but do not contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are specific to this document:

address-of-record: A **Session Initiation Protocol (SIP) URI** that specifies a domain with a location service that can map the URI to another URI for a user, as described in [RFC3261].

Augmented Backus-Naur Form (ABNF): A modified version of Backus-Naur Form (BNF), commonly used by Internet specifications. ABNF notation balances compactness and simplicity with reasonable representational power. ABNF differs from standard BNF in its definitions and uses of naming rules, repetition, alternatives, order-independence, and value ranges. For more information, see [RFC5234].

autodiscovery: An ability to discover a first hop Session Initiation Protocol (SIP) proxy without explicitly configuring the proxy name.

certificate: A certificate is a collection of attributes (1) and extensions that can be stored persistently. The set of attributes in a certificate can vary depending on the intended usage of the certificate. A certificate securely binds a public key to the entity that holds the corresponding private key. A certificate is commonly used for authentication (2) and secure exchange of information on open networks, such as the Internet, extranets, and intranets. Certificates are digitally signed by the issuing **certification authority (CA)** and can be issued for a user, a computer, or a service. The most widely accepted format for certificates is defined by the ITU-T X.509 version 3 international standards. For more information about attributes and extensions, see [RFC3280] and [X509] sections 7 and 8.

certification authority (CA): A third party that issues public key **certificates**. Certificates serve to bind public keys to a user identity. Each user and certification authority (CA) can decide whether to trust another user or CA for a specific purpose, and whether this trust should be transitive. For more information, see [RFC3280].

domain: A set of users and computers sharing a common namespace and management infrastructure. At least one computer member of the set must act as a domain controller (DC) and host a member list that identifies all members of the domain, as well as optionally hosting the Active Directory service. The domain controller provides authentication (2) of members, creating a unit of trust for its members. Each domain has an identifier that is shared among its members. For more information, see [MS-AUTHSOD] section 1.1.1.5 and [MS-ADTS].

Domain Name System (DNS): A hierarchical, distributed database that contains mappings of domain names (1) to various types of data, such as IP addresses. DNS enables the location of computers and services by user-friendly names, and it also enables the discovery of other information stored in the database.

Dynamic Host Configuration Protocol (DHCP): A protocol that provides a framework for passing configuration information to hosts on a TCP/IP network, as described in [RFC2131].

endpoint: A device that is connected to a computer network.

fully qualified domain name (FQDN): An unambiguous domain name (2) that gives an absolute location in the **Domain Name System's (DNS)** hierarchy tree, as defined in [RFC1035] section 3.1 and [RFC2181] section 11.

keepalive message: A protocol message that is sent between a protocol client and a protocol server to help ensure that a connection is considered active by all **endpoints**. Inactive connections are considered idle and are likely to be closed by either **endpoint** to conserve resources.

outbound proxy: A network node that acts as a proxy for outbound traffic between a protocol client and a protocol server.

REGISTER: A **Session Initiation Protocol (SIP)** method that is used by an SIP client to register the client address with an SIP server.

root certificate: A self-signed **certificate** that identifies the public key of a root **certification authority (CA)** and has been trusted to terminate a certificate chain.

Session Initiation Protocol (SIP): An application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants. **SIP** is defined in [RFC3261].

SIP message: The data that is exchanged between **Session Initiation Protocol (SIP)** elements as part of the protocol. An SIP message is either a request or a response.

SIP registrar: A **Session Initiation Protocol (SIP)** server that accepts REGISTER requests and places the information that it receives from those requests into the location service for the domain that it handles.

SIP request: A **Session Initiation Protocol (SIP)** message that is sent from a user agent client (UAC) to a **user agent server (UAS)** to call a specific operation.

SIP response: A **Session Initiation Protocol (SIP)** message that is sent from a **user agent server (UAS)** to a user agent client (UAC) to indicate the status of a request from the UAC to the UAS.

SIP transaction: A **SIP transaction** occurs between a UAC and a **UAS**. The **SIP transaction** comprises all messages from the first request sent from the UAC to the **UAS** up to a final response (non-1xx) sent from the **UAS** to the UAC. If the request is INVITE, and the final response is a non-2xx, the **SIP transaction** also includes an ACK to the response. The ACK for a 2xx response to an INVITE request is a separate **SIP transaction**.

Transmission Control Protocol (TCP): A protocol used with the Internet Protocol (IP) to send data in the form of message units between computers over the Internet. TCP handles keeping track of the individual units of data (called packets) that a message is divided into for efficient routing through the Internet.

Transport Layer Security (TLS): A security protocol that supports confidentiality and integrity of messages in client and server applications communicating over open networks. **TLS** supports server and, optionally, client authentication by using X.509 certificates (as specified in [X509]). **TLS** is standardized in the IETF TLS working group. See [RFC4346].

Uniform Resource Identifier (URI): A string that identifies a resource. The URI is an addressing mechanism defined in Internet Engineering Task Force (IETF) Uniform Resource Identifier (URI): Generic Syntax [RFC3986].

user agent server (UAS): A logical entity that generates a response to a **Session Initiation Protocol (SIP)** request. The response either accepts, rejects, or redirects the request. The role of the UAS lasts only for the duration of that transaction. If a process responds to a request, it acts as a UAS for that transaction. If it initiates a request later, it assumes the role of a user agent client (UAC) for that transaction.

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

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.

[MS-SIPAE] Microsoft Corporation, "Session Initiation Protocol (SIP) Authentication Extensions".

[MS-SIPCOMP] Microsoft Corporation, "Session Initiation Protocol (SIP) Compression Protocol".

[MS-SIPREGE] Microsoft Corporation, "Session Initiation Protocol (SIP) Registration Extensions".

[RFC1035] Mockapetris, P., "Domain Names - Implementation and Specification", STD 13, RFC 1035, November 1987, http://www.ietf.org/rfc/rfc1035.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

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997, http://www.ietf.org/rfc/rfc2131.txt

[RFC2132] Alexander, S., and Droms, R., "DHCP Options and BOOTP Vendor Extensions", RFC 2132, March 1997, http://www.ietf.org/rfc/rfc2132.txt

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

[RFC2459] Housley, R., Ford, W., Polk, W., and Solo, D., "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", RFC 2459, January 1999, http://www.rfc-editor.org/rfc/rfc2459.txt

[RFC2782] Gulbrandsen, A., Vixie, P., and Esibov, L., "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000, http://www.ietf.org/rfc/rfc2782.txt

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and Schooler, E., "SIP: Session Initiation Protocol", RFC 3261, June 2002, http://www.ietf.org/rfc/rfc3261.txt

[RFC3361] Schulzrinne, H., "Dynamic Host Configuration Protocol (DHCP-for-IPv4) Option for Session Initiation Protocol (SIP) Servers", August 2002, http://www.rfc-editor.org/rfc/rfc3361.txt

[RFC5234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, http://www.rfc-editor.org/rfc/rfc5234.txt

1.2.2 Informative References

[RFC793] Postel, J., Ed., "Transmission Control Protocol: DARPA Internet Program Protocol Specification", RFC 793, September 1981, http://www.rfc-editor.org/rfc/rfc793.txt

1.3 Overview

This document specifies a proprietary extension to the **Session Initiation Protocol (SIP)** to support connection management.

This protocol defines a mechanism for the protocol client to automatically discover its SIP **outbound proxy**. This protocol also defines the **certificate** requirement for the **Transport Layer Security (TLS)** channel from the protocol client to the outbound proxy. It defines a mechanism to negotiate the keep-alive capability between the protocol client and outbound proxy using **keepalive messages**. The keep-alive negotiation is conducted with **SIP messages**. This protocol also defines the actual mechanism for keep-alive negotiation by sending keepalive messages on the established connection.

Keep-alive negotiation refers to a mechanism that keeps a **Transmission Control Protocol (TCP)** connection from timing out because of inactivity. A keep-alive mechanism negotiates between a protocol client and an outbound proxy by using a custom header that specifies the proposed or supported keep-alive capabilities in **SIP requests**.

1.4 Relationship to Other Protocols

The Connection Management Protocol depends on the following protocols:

[RFC1035] for resolving names of network resources.

[RFC2782] and [RFC3361] for automatically discovering the SIP outbound proxy.

[RFC793] for establishing persistent, reliable transport.

1.5 Prerequisites/Preconditions

The SIP outbound proxy needs to obtain a valid certificate if the SIP outbound proxy implementation supports TLS. The protocol client needs to obtain the **root certificate** from a trusted **certification authority (CA)** to verify the certificate presented by the SIP outbound proxy.

1.6 Applicability Statement

This protocol is applicable to all protocol clients that are not explicitly configured to connect to the SIP outbound proxy with a specific address and port and using a specific transport.

1.7 Versioning and Capability Negotiation

The **autodiscovery** mechanism does not negotiate versioning or any capabilities. After the persistent, reliable, in-order transport has been established, the protocol client can request negotiation of a keep-alive mechanism to keep the persistent transport from being disconnected because of inactivity. The negotiation is conducted using a custom **Ms-Keep-Alive** header field in SIP requests.

The syntax of the **Ms-Keep-Alive** header field is specified in section 2, and its use is specified in section 3.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

This protocol uses **Domain Name System (DNS)** SRV, as specified in the format section in [RFC2782], and DNS A, as specified in [RFC1035] section 3.4.1, and SIP server **Dynamic Host Configuration Protocol (DHCP)** discovery<1>, as specified in [RFC3361] section 3, for automatic discovery of a SIP outbound proxy, and does not define any new message format for auto-discovery of SIP outbound proxy nodes.

This protocol uses a custom header field in SIP requests to support the keep-alive negotiation. The name of the new header field is **Ms-Keep-Alive** and the new header field can be used to specify proposed and supported keep-alive capabilities to keep the persistent, reliable, in-order transport from being disconnected because of inactivity.

All SIP traffic MUST be transported over TCP. TLS on the established TCP connection for added security is optional.

2.2 Message Syntax

SIP server discovery using DNS uses [RFC1035] and [RFC2782]. SIP server discovery using DHCP<2> uses [RFC3361]. The keep-alive protocol relies on the SIP message format, as specified in [RFC3261] section 7. All of the message syntax specified in this document is described in words and in **Augmented Backus-Naur Form (ABNF)**, as defined in [RFC5234].

2.2.1 Ms-Keep-Alive Header Field Syntax

This protocol extends the definition of **message-header** in [RFC3261] section 25 as follows. The **Ms-Keep-Alive** header field is the only field added to the list, along with the details for this field at the bottom.

```
message-header  =  (Accept

                /  Accept-Encoding
                /  Accept-Language
                /  Alert-Info
                /  Allow
                /  Authentication-Info
                /  Authorization
                /  Call-ID
                /  Call-Info
                /  Contact
                /  Content-Disposition
                /  Content-Encoding
                /  Content-Language
                /  Content-Length
                /  Content-Type
                /  CSeq
                /  Date
                /  Error-Info
                /  Expires
                /  From
                /  In-Reply-To
                /  Max-Forwards
                /  MIME-Version
                /  Min-Expires
                /  Ms-Keep-Alive
                /  Organization
                /  Priority
                /  Proxy-Authenticate
```

```
                  /  Proxy-Authorization
                  /  Proxy-Require
                  /  Record-Route
                  /  Reply-To
                  /  Require
                  /  Retry-After
                  /  Route
                  /  Server
                  /  Subject
                  /  Supported
                  /  Timestamp
                  /  To
                  /  Unsupported
                  /  User-Agent
                  /  Via
                  /  Warning
                  /  WWW-Authenticate
                  /  extension-header) CRLF
Ms-Keep-Alive   = "ms-keep-alive" HCOLON ms-keep-alive-value
ms-keep-alive-value = ms-keep-alive-role *(SEMI ms-keep-alive-capability) [ SEMI ms-keep-
alive-timeout ] *(SEMI generic-param)
ms-keep-alive-role = "UAC" / "UAS"
ms-keep-alive-capability = ms-keep-alive-mechanism EQUAL BOOLEAN
BOOLEAN = "yes" / "no"
ms-keep-alive-mechanism = "hop-hop" / "end-end" / "tcp" / token
ms-keep-alive-timeout = "timeout" EQUAL 1*DIGIT
```

The **Ms-Keep-Alive** header field SHOULD be present in a SIP request and in the corresponding **SIP response** for negotiating the keep-alive mechanism that is to be used on the TCP connection on which the SIP request is sent. The **Ms-Keep-Alive** header field MUST NOT appear more than once in a SIP request or SIP response.

The **Ms-Keep-Alive** header field MUST contain the **ms-keep-alive-role** parameter. This parameter identifies the role of the sender in the keep-alive negotiation. The value MUST be one of the following:

- "UAC": The sender is the initiator of the keep-alive negotiation.
- "UAS"**:** The sender is the responder to the keep-alive negotiation.

ms-keep-alive-capability: Specifies the keep-alive capability of the involved party. This value has two parts.

- **ms-keep-alive-mechanism**: Specifies the keep-alive mechanism. The value of **ms-keep-alive-mechanism** MUST be one of the following:
  - "hop-hop": The hop-by-hop keep-alive mechanism is specified in section 3.4.
  - "end-end": Reserved for future use.
  - "tcp": Reserved for future use.
  - Any **token** value as specified in [RFC3261] section 25: Reserved for future use.
- **BOOLEAN:** Indicates whether the specified keep-alive mechanism is supported by the sending party. This value MUST be either "yes" or "no". If any mechanism other than "hop-hop" is present, the value of **BOOLEAN** MUST be "no".

The **Ms-Keep-Alive** header field MUST have an **ms-keep-alive-timeout** parameter if the **UAS** accepts the keepalive message sent for keep-alive negotiation. The parameter value MUST be an unsigned integer that indicates the time, in seconds, that the connection will be kept alive.

generic-param: Reserved for future use.

2.2.2 keep-alive Message Syntax

The keepalive message for the hop-by-hop keep-alive mechanism is composed entirely of a **double-CRLF** with the following ABNF, as defined in [RFC5234], code:

```
double-CRLF = CR LF CR LF
CR = 0x0d
LF = 0x0a
```

The keepalive message MUST be sent on a connection on which the hop-by-hop keep-alive mechanism has been successfully negotiated. The hop-by-hop mechanism is specified in section 3.4.

3 Protocol Details

3.1 SIP Client Details - SIP Outbound Proxy Autodiscovery

This section specifies the protocol client behavior for automatically discovering its SIP outbound proxy. There is no requirement on the SIP server in for SIP outbound proxy discovery. This section only applies to protocol clients.

3.1.1 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.

The protocol client SHOULD maintain a list to store the returned DNS SRV records and DHCP entries from the queries.

3.1.2 Timers

The protocol client MUST maintain a DHCP discovery timer with a recommended timeout value of 5 seconds if it supports SIP server DHCP discovery.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

To automatically discover the address of the SIP outbound proxy, the protocol client MUST first obtain an **address-of-record**. The address-of-record is obtained from user input or from any offline storage. The **host** part in the address-of-record, which is defined in [RFC3261] section 6, MUST be used as the **domain** for the autodiscovery mechanism. As an example, in the address-of-record of "sip:alice@contoso.com", the **host** part is "contoso.com" and the domain for autodiscovery is also "contoso.com".

3.1.5 Message Processing Events and Sequencing Rules

After the domain is obtained, as specified in section 3.1.4, the protocol client MUST query the following DNS SRV entries in parallel for the transport associated with the queried DNS SRV entry.

- _sipinternaltls._tcp.<*domain*> - for the associated transport, TLS
- _sipinternal._tcp. <*domain*> - for the associated transport, TCP
- _sip._tls. <*domain*> - for the associated transport, TLS
- _sip._tcp.<*domain*> - for the associated transport, TCP

Replace <*domain*> with the domain obtained from the SIP **URI**.

For example, the following DNS SRV entries are queried for "sip:alice@contoso.com":

- _sipinternaltls._tcp.contoso.com

- _sipinternal._tcp.contoso.com
- _sip._tls.contoso.com
- _sip._tcp.contoso.com

For each DNS SRV query, the protocol client MUST sort the returned records by the priority of the DNS SRV record. A query is complete when a DNS SRV response is received. The response contains zero or more DNS SRV records.

In addition, the protocol client SHOULD<3> perform SIP server DHCP discovery by issuing a DHCP INFORM message, as specified in [RFC2131] section 4, with the DHCP Option 120 request, as specified in [RFC3361] section 3, for SIP server discovery, with the following restrictions and exception. The client SHOULD include "MS-UC-Client" as the DHCP Option 60 Vendor Class Identifier for the Option 120 request, as specified in [RFC2132] section 9.13. The DHCP INFORM packet MUST be sent to the local broadcast address "255.255.255.255". The protocol client SHOULD also send the DHCP INFORM packet to the corresponding DHCP on any local network interface where DHCP is enabled. The protocol client SHOULD ignore any IP addresses returned. The SIP server DHCP discovery is done once the protocol client received a DHCPACK, as defined in [RFC2131], with two results, and the client MUST cancel the DHCP discovery timer. If two results are received, the results are ordered randomly. Because DHCP results do not include port information, the protocol client MUST use the default TLS port, 5061, when trying to connect to the entries with TLS, and use the default TCP port, 5060, when trying to connect to the entries with TCP.

Once the protocol client completes the DNS SRV queries and SIP server DHCP discovery, the protocol client MUST group the records returned in the following sequence.

- _sipinternaltls._tcp.<*domain*> - for TLS
- DHCP results – for TLS
- _sipinternal._tcp. <*domain*> - for TCP
- DHCP results – for TCP
- _sip._tls. <*domain*> - for TLS
- _sip._tcp.<*domain*> - for TCP

If both _sipinternaltls._tcp.<*domain*> and _sip._tls.<*domain*> queries are completed before the _sip._tcp.<*domain*> query is completed, the protocol client MUST add both sets of returned records to the result. The protocol client SHOULD include the records from _sipinternal._tcp.<*domain*> in the result if _sipinternal._tcp.<*domain*> has already completed, and the protocol client SHOULD NOT include any records from the pending _sip._tcp.<*domain*> in the result. If the _sip._tcp.<*domain*> query is completed before one of the other queries, the protocol client SHOULD wait for all queries to complete and include records from all queries in the result.

The protocol client SHOULD ignore records from the _sipinternaltls._tcp.<*domain*> and _sip._tls.<*domain*> queries whose domain or subdomain parts do not match <*domain*>. For example, the protocol client ignores "server1.fakecontoso.com", but the protocol client accepts "server1.contoso.com" or "server1.subdomain.contoso.com". The protocol client MUST use TLS to connect to the address in the valid records that are returned from the _sipinternaltls._tcp.<*domain*> and _sip._tls.<*domain*> queries. The protocol client MUST use TCP to connect to the address in the records returned from the _sipinternal._tcp.<*domain*> and _sip._tcp.<*domain*> queries.

After getting the result, the protocol client SHOULD append the following entries to the result, if they are not already present, to produce the final result:

- sipinternal.<domain>:443 - for TLS
- sipinternal.<domain> - for TCP

- sip.<domain> :443- for TLS
- sip.<domain> - for TCP
- sipexternal.<domain>:443 for TLS
- sipexternal.<domain> for TCP

The protocol client SHOULD try to connect to the entries in the result sequentially. Before the protocol client tries to connect to the current entry, the protocol client SHOULD perform a new DNS A lookup for the entry. The client SHOULD try to connect to the IP addresses returned from the A lookup sequentially. If the protocol client encounters a DNS A lookup failure, or if the protocol client encounters TCP connection failure for all IP addresses for the current entry, the protocol client SHOULD try the next entry. For all other failures, or if all entries in the final result are exhausted, the protocol client MUST treat the autodiscovery attempt as a failure.

3.1.6 Timer Events

When the DHCP discovery timer fires, the SIP server DHCP discovery is considered to be completed with no results.

3.1.7 Other Local Events

None.

3.2 SIP Client Details - TLS Certificate Requirement

The protocol client MUST use the method specified in [RFC2246] section 7 to perform key exchange with, and authenticate the identity of, the outbound proxy by a certificate in TLS. This section specifies the certificate requirement for the protocol client to the outbound proxy TLS channel.

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

None.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 SIP Server Details - TLS Certificate Requirement

For the purpose of authenticating the outbound proxy computer, the certificate presented by the outbound proxy during TLS negotiation MUST have a subject name, as defined in [RFC2459] section 4.1.2.6, of the **fully qualified domain name (FQDN)** (the unambiguous domain name that gives the absolute location) of the outbound proxy.

3.3.1 Abstract Data Model

None.

3.3.2 Timers

None.

3.3.3 Initialization

None.

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

None.

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

3.4 keep-alive Details

This section describes the negotiation and the hop-by-hop keep-alive mechanism. The keep-alive mechanism serves two purposes. First, it serves to keep the connection between the protocol client and the outbound proxy alive by keeping network components operating below the TCP layer between the protocol client and its first hop SIP proxy from timing out and disconnecting the TCP connection because of inactivity. In addition, if the first hop SIP proxy is also the **SIP registrar** (registrar) for the protocol client, the registrar uses the lack of periodic keepalive message the protocol client **endpoint** for inactivity. The hop-by-hop keep-alive mechanism does not forward the keepalive message across multiple SIP entities. The outbound proxy MUST NOT behave as if every connection originating from the protocol client endpoints has keep-alive enabled. The keep-alive mechanism is enabled or disabled for each individual connection.

3.4.1 Abstract Data Model

None.

3.4.2 Timers

The outbound proxy MUST maintain an expiry timer.

The outbound proxy SHOULD<4> define a time-out value for keeping the connection alive. If the outbound proxy accepts the keepalive message SIP request, the timer SHOULD<5> be set to the time-out value plus a grace period of at least a **SIP transaction** (transaction) timeout, and the outbound proxy MUST send the SIP response with an **Ms-Keep-Alive** header field that contains the time-out value in **ms-keep-alive-timeout**. The expiry timer MUST be reset to the time-out value plus a grace period whenever any traffic is received on the connection.

Protocol clients MUST maintain a refresh timer.

When the protocol client retrieves the time-out value from the **ms-keep-alive-timeout** parameter in the **Ms-Keep-Alive** header field, it SHOULD<6> set the refresh timer to two-thirds of the timeout value. The protocol client refresh timer is reset to the time-out value if the protocol client sends any data on the connection.

3.4.3 Initialization

None.

3.4.4 Higher-Layer Triggered Events

The protocol client MUST always initiate the hop-by-hop keep-alive negotiation. The outbound proxy MUST NOT initiate hop-by-hop keep-alive negotiation.

3.4.5 Message Processing Events and Sequencing Rules

The keep-alive negotiation exchanges SIP requests and SIP responses to communicate the keep-alive negotiation capabilities between the protocol client SIP endpoint and the outbound proxy. The keep-alive negotiation MUST begin immediately after the transport, including a compression negotiation, as specified in [MS-SIPCOMP] section 3, has been successfully established.

3.4.5.1 Initiating keep-alive Negotiation

After compression negotiation, as specified in [MS-SIPCOMP] section 3, is complete, the keep-alive negotiation can be conducted on any SIP request that is sent from the protocol client to the outbound proxy.

The protocol client MUST include an **Ms-Keep-Alive** header field in the SIP request. The **Ms-Keep-Alive** header field is constructed as specified in section 2.2.1, with **ms-keep-alive-role** set to "UAC" and **ms-keep-alive-capability** specified as "hop-hop" and enabled. The "end-end" and "tcp" values of **ms-keep-alive-capability** are currently not supported, and the client SHOULD NOT specify these capabilities.

For an example of a protocol client-initiated **REGISTER** request that also includes the keep-alive negotiation, see section 4.1.

3.4.5.2 Responding to a keep-alive Request

If the outbound proxy receives a SIP request that contains more than one **Ms-Keep-Alive** header field, the outbound proxy MUST treat the first **Ms-Keep-Alive** header field as the only **Ms-Keep-Alive** header field in the SIP request and ignore any additional **Ms-Keep-Alive** header fields. When the outbound proxy receives a SIP request that contains an **Ms-Keep-Alive** header field from a connection, the outbound proxy MUST inspect the **ms-keep-alive-role** parameter in the **Ms-Keep-Alive** header field. The outbound proxy SHOULD proceed further with the keep-alive negotiation only if **ms-keep-alive-role** is "UAC". The outbound proxy then MUST inspect the set of **ms-keep-alive-capability** values and SHOULD proceed further only if the outbound proxy supports the negotiation mechanism specified and the **BOOLEAN** field is set to "yes". At present, only the behavior of "hop-hop" is defined. The "end-end" and "tcp" values are not supported and the outbound proxy MUST ignore these capabilities.

If the outbound proxy does not generate a successful response for the request that initiated the keep-alive negotiation, a failure response is sent from the outbound proxy to the client with an error code greater than or equal to 400. The client and the outbound proxy MUST treat the keep-alive as a failure, and the client MUST NOT send the keep-alive message. If the server generated a successful response, the negotiation can proceed.

If the outbound proxy accepts the keep-alive mechanism, it MUST construct an **Ms-Keep-Alive** header field, as specified in section 2.2.1, and it MUST insert the header field in the successful SIP response before the SIP response is sent.

The **ms-keep-alive-role** parameter in the **Ms-Keep-Alive** header field MUST be set to "UAS" to indicate that the **Ms-Keep-Alive** header field is a negotiation SIP response and not a mirrored copy of the header field in the SIP request. For each **ms-keep-alive-capability** in the **Ms-Keep-Alive** header field from the SIP request that the outbound proxy supports, the outbound proxy SHOULD add the same **ms-keep-alive-capability** with a Boolean value of "yes" to the **Ms-Keep-Alive** header field in the SIP response. At present, only the ms-keep-alive-mechanism of "hop-hop" is defined. The outbound proxy MUST NOT include "end-end" or "tcp" in the **ms-keep-alive-capability** in the response.

The outbound proxy MUST also insert an **ms-keep-alive-timeout.** with the desired timeout value in seconds.

For an example of an outbound proxy-side REGISTER SIP response that also includes the keep-alive negotiation, see section 4.2.

3.4.5.3 Processing the SIP Response to a keep-alive Request

If the protocol client receives a failure SIP response to the SIP request that initiates the keep-alive negotiation, the protocol client MUST treat the keep-alive negotiation as a failure. If the keep-alive negotiation failed, the protocol client MUST NOT send the keep-alive message.

If the protocol client receives a successful SIP response to the SIP request, it inspects the **Ms-Keep-Alive** header field. If the SIP response contains more than one **Ms-Keep-Alive** header field, the protocol client MUST ignore all the **Ms-Keep-Alive** header fields in the SIP response. If the header field is not present, the protocol client MUST also treat the keep-alive negotiation as a failure and the protocol client MUST NOT send the keepalive message.

The protocol client inspects the set of supported **ms-keep-alive-capability** values inside the **Ms-Keep-Alive** header field. If there is no intersection between the set of **ms-keep-alive-capability** values supported by the protocol client and the set of **ms-keep-alive-capability** values present in the **Ms-Keep-Alive** header field, the keep-alive negotiation has failed. If the intersection is not empty, the protocol client MUST then choose one of the **ms-keep-alive-capability** values in the intersection as its keep-alive mechanism. The protocol client MUST retrieve the unsigned integer value from the **ms-keep-alive-timeout** parameter and reset its refresh timer. At this point, the keep-alive negotiation has succeeded.

3.4.5.4 Sending Periodic Hop-by-Hop keep-alive Message

When the protocol client's refresh timer expires on a keep-alive enabled connection, the protocol client MUST send a keepalive message constructed as specified in section 2.2.2.

3.4.6 Timer Events

Section 3.4.5.4 covers the case for sending a periodic hop-by-hop keepalive message when the protocol client's refresh timer fires. The outbound proxy expiry timer fires on a connection after a period of inactivity equal to the expiry timer. Extended protocol client endpoint inactivity can be caused by an application crash or other reasons. When the protocol client endpoint becomes inactive, the outbound proxy SHOULD remove the protocol client endpoint and its registration binding associated with the connection if the outbound proxy is the SIP registrar for the protocol client endpoint. At any point in time after the protocol client endpoint becomes inactive, the outbound proxy SHOULD close the connection to the protocol client endpoint if the outbound proxy is the SIP registrar for the protocol client endpoint. The SIP registrar MUST NOT send a NOTIFY message when the SIP registrar removes the protocol client endpoint, which is specified in [MS-SIPREGE] section 3.2.2.4, because the protocol client endpoint is not expected to respond. If the outbound proxy is not a SIP registrar for the protocol client endpoint, the outbound proxy does not need to take any action when the timer fires.

3.4.7 Other Local Events

None.

3.5 Outbound Proxy Connection Management Details

This section specifies the outbound proxy side connection management details in relation to other SIP extensions. There is no requirement on the protocol client for outbound proxy connection management. This section only applies to the outbound proxy.

3.5.1 Abstract Data Model

None.

3.5.2 Timers

The outbound proxy MUST keep a connection timer on the connection. This timer is used for closing the connection if the protocol client has not successfully completed a transaction. The connection timer MUST be set when the connection is established. The timer is reset to the original time-out value if a provisional SIP response, as specified in [RFC3261] section 8.2.6.1, is sent to the protocol client on the connection. The timer is cancelled only when a successful SIP response is sent to the protocol client on the connection. Outbound proxy implementations SHOULD use a timeout period of 32 seconds and it SHOULD close the connection at any point in time after the timeout period elapses.

In addition to the connection timer, the outbound proxy MUST also keep an idle timer. Outbound proxy implementations SHOULD use an idle timer value of 15 minutes and 32 seconds. If there is no traffic on the connection for the time-out period, the outbound proxy SHOULD close the connection at any point in time after time-out period elapses. The timer is first set when the connection is established. The timer is reset to the original idle timer value if traffic is sent or received on the connection.

3.5.3 Initialization

None.

3.5.4 Higher-Layer Triggered Events

None.

3.5.5 Message Processing Events and Sequencing Rules

When the outbound proxy successfully establishes a connection to the protocol client endpoint and authenticates the protocol client endpoint using the mechanism defined in [MS-SIPAE] section 3 on the same connection, the outbound proxy MUST close any existing connection authenticated using [MS-SIPAE] and its associated states, including any security association established by using the mechanism defined in [MS-SIPAE], for the same protocol client endpoint.

3.5.6 Timer Events

If the connection timer fires and the client endpoint has not been authenticated using [MS-SIPAE] section 3 on the connection, the outbound proxy MUST close the connection and release all states associated with the connection to prevent a denial of service attack.

If the idle timer fires, the outbound proxy SHOULD close the connection and release all states associated with the connection. This is to remove any stale state in the case where the protocol client has crashed.

3.5.7 Other Local Events

None.

4 Protocol Examples

4.1 Protocol Client Request for the keep-alive Negotiation

The following is a sample REGISTER SIP request for the keep-alive negotiation.

```
REGISTER sip:contoso.com SIP/2.0
Via: SIP/2.0/TLS 10.56.65.232:12345
Max-Forwards: 70
From: <sip:alice@contoso.com>;tag=cf6792e59e;epid=99ad5894fe
To: <sip:alice@contoso.com>
Call-ID: 63f9d742e7374b3cae3930824bed57ee
CSeq: 1 REGISTER
Contact: <sip:10.56.65.232:49729;transport=tls;ms-opaque=b26b785992>;methods="INVITE,
MESSAGE, INFO, OPTIONS, BYE, CANCEL, NOTIFY, ACK, REFER,
BENOTIFY";proxy=replace;+sip.instance="<urn:uuid:6A4F8F80-9C64-5FE8-93D1-FE43A25CD7FF>"
User-Agent: SIPSTACK/1.0 APPLICATION/1.0 (SIP Application)
ms-keep-alive: UAC;hop-hop=yes
Event: registration
Content-Length: 0
```

Note that this example is for illustration purposes only. Actual registration would include additional headers required for authentication as specified in [MS-SIPAE].

4.2 Outbound Proxy Response for the keep-alive Negotiation

The following is a sample SIP response to a REGISTER SIP request for the keep-alive negotiation.

```
SIP/2.0 200 OK
ms-keep-alive: UAS; tcp=no; hop-hop=yes; end-end=no; timeout=300
From: <sip:alice@contoso.com>;tag=cf6792e59e;epid=99ad5894fe
To: <sip: alice@contoso.com>;tag=5B9D8DF714B02667F171A8E1AA4E971A
Call-ID: 63f9d742e7374b3cae3930824bed57eeCSeq: 1
REGISTERVia: SIP/2.0/TLS10.56.65.232:49729;ms-received-port=49729;ms-received-cid=2D9D00
Contact: <sip:10.56.65.232:49729;transport=tls;ms-opaque=b26b785992;ms-received-
cid=2D9D00>;expires=7200;+sip.instance="<urn:uuid:6a4f8f80-9c64-5fe8-93d1-
fe43a25cd7ff>";gruu=sip:alice@contoso.com;opaque=user:epid:gI9PamSc6F-T0f5DolzX_wAA;gruu
Expires: 7200presence-state: register-action="added"
Allow-Events: vnd-microsoft-provisioning,vnd-microsoft-roaming-contacts,vnd-microsoft-
roaming-ACL,presence,presence.wpending,vnd-microsoft-roaming-self,vnd-microsoft-provisioning-
v2Supported: adhoclistServer: SIPSERVER/3.0Supported: msrtc-event-categories
Content-Length: 0
```

Note that this example is for illustration purposes only. Actual registration would include additional headers required for authentication as specified in [MS-SIPAE].

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

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.

- Microsoft Office Communications Server 2007
- Microsoft Office Communications Server 2007 R2
- Microsoft Office Communicator 2007
- Microsoft Office Communicator 2007 R2
- Microsoft Lync Server 2010
- Microsoft Lync 2010
- Microsoft Lync Server 2013
- Microsoft Lync Client 2013/Skype for Business
- Microsoft Skype for Business 2016
- Microsoft Skype for Business Server 2015

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 2.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: SIP server DHCP discovery is not supported.

<2> Section 2.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: SIP server DHCP discovery is not supported.

<3> Section 3.1.5: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<4> Section 3.4.2: A time-out value of 300 seconds is recommended.

<5> Section 3.4.2: A grace period is set to Timer B or Timer F from [RFC3261], 32 seconds by default.

<6> Section 3.4.2: All products other than Office Communications Server 2007, Office Communicator 2007: This behavior has been updated to support features as described by the MSDN Knowledgebase Article #967673, "Description of the Communicator 2007 R2 hotfix rollup package: April 2009".

7 Change Tracking

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

8 Index

N

O

P

R

S

T

V