[MS-TURN]:

Traversal Using Relay NAT (TURN) Extensions

Tools. The Open Specifications documentation does 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 documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

Release: September 15, 2017

Revision Summary

Date	Revision History	Revision Class	Comments
4/4/2008	0.1	New	Initial version
4/25/2008	0.2	Minor	Updated the technical content.
6/27/2008	1.0	Major	Updated and revised the technical content.
8/15/2008	1.01	Minor	Revised and edited the technical content.
12/12/2008	2.0	Major	Updated and revised the technical content.
2/13/2009	2.01	Minor	Revised and edited the technical content.
3/13/2009	2.02	Minor	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	None	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	None	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	3.0	None	No changes to the meaning, language, or formatting of the technical content.
3/18/2011	3.0	None	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	3.0	None	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	4.0	Major	Significantly changed the technical content.
4/11/2012	4.0	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	4.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	4.1	Minor	Clarified the meaning of the technical content.
2/11/2013	4.1	None	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
7/30/2013	4.1	None	No changes to the meaning, language, or formatting of the technical content.
11/18/2013	4.1	None	No changes to the meaning, language, or formatting of the technical content.
2/10/2014	4.1	None	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	5.0	Major	Significantly changed the technical content.
7/31/2014	5.1	Minor	Clarified the meaning of the technical content.
10/30/2014	5.1	None	No changes to the meaning, language, or formatting of the technical content.
3/30/2015	6.0	Major	Significantly changed the technical content.
6/30/2015	6.0	None	No changes to the meaning, language, or formatting of the technical content.
9/4/2015	6.0	None	No changes to the meaning, language, or formatting of the technical content.
7/1/2016	7.0	Major	Significantly changed the technical content.
9/14/2016	7.0	None	No changes to the meaning, language, or formatting of the technical content.
12/15/2016	8.0	Major	Significantly changed the technical content.
3/28/2017	8.1	Minor	Clarified the meaning of the technical content.
6/20/2017	8.2	Minor	Clarified the meaning of the technical content.
8/1/2017	9.0	Major	Significantly changed the technical content.
9/15/2017	10.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction 6
1.1 Glossary 6
1.2 References 8
1.2.1 Normative References 8
1.2.2 Informative References 9
1.3 Overview 10
1.4 Relationship to Other Protocols 13
1.5 Prerequisites/Preconditions 13
1.6 Applicability Statement 14
1.7 Versioning and Capability Negotiation 14
1.8 Vendor-Extensible Fields 14
1.9 Standards Assignments 14

2 Messages 15
2.1 Transport 15
2.1.1 Pseudo-TLS over TCP 15
2.1.2 TLS over TCP 17
2.1.3 HTTPS over TCP 18
2.1.4 TCP 18
2.1.5 UDP 18
2.2 Message Syntax 18
2.2.1 Message Header 18
2.2.2 Message Attribute 19
2.2.2.1 Mapped Address 21
2.2.2.2 Username 21
2.2.2.3 Message Integrity 22
2.2.2.4 Error Code 23
2.2.2.5 Unknown Attributes 24
2.2.2.6 Lifetime 24
2.2.2.7 Alternate Server 25
2.2.2.8 Magic Cookie 26
2.2.2.9 Bandwidth 26
2.2.2.10 Destination Address 26
2.2.2.11 Remote Address 27
2.2.2.12 Data 28
2.2.2.13 Nonce 28
2.2.2.14 Realm 28
2.2.2.15 Requested Address Family 29
2.2.2.16 XOR Mapped Address 29
2.2.2.17 MS-Version Attribute 30
2.2.2.18 MS-Alternate Host Name 31
2.2.2.19 MS-Sequence Number Attribute 32
2.2.2.20 MS-Service Quality Attribute 33
2.2.2.21 MS-Alternate Mapped Address 33
2.2.2.22 Multiplexed TURN Session ID 34
2.2.3 Multiplexed TURN 34

3 Protocol Details 36
3.1 Common Details 36
3.1.1 Abstract Data Model 36
3.1.2 Timers 36
3.1.3 Initialization 36
3.1.4 Higher-Layer Triggered Events 36
3.1.5 Message Processing Events and Sequencing Rules 36
3.1.6 Timer Events 36

3.1.7 Other Local Events ... 36
3.1.8 Forming Outbound TURN Messages ... 36
3.1.9 Forming Raw Data ... 36
3.1.10 Verifying Inbound TURN Messages ... 36
3.1.11 Message Authentication ... 37
3.1.12 Digest Challenge Extension ... 37
3.2 Client Details ... 37
3.2.1 Abstract Data Model ... 37
3.2.2 Timers ... 37
3.2.3 Initialization ... 38
3.2.4 Higher-Layer Triggered Events ... 38
3.2.4.1 Allocating Public Transport Addresses ... 38
3.2.4.2 Sending TURN Encapsulated Data to the Peer ... 38
3.2.4.3 Set the Peer as the Active Destination ... 38
3.2.4.4 Tearing Down an Allocation ... 39
3.2.4.5 Sending Non-TURN Data to the Peer ... 39
3.2.4.6 Sending Multiplexed TURN Encapsulated Data to the Peer ... 39
3.2.5 Message Processing Events and Sequencing Rules ... 39
3.2.5.1 Receiving Allocate Response Messages ... 39
3.2.5.2 Receiving Allocate Error Response Messages ... 40
3.2.5.3 Receiving Set Active Destination Response Messages ... 41
3.2.5.4 Receiving Set Active Destination Error Response Messages ... 41
3.2.5.5 Receiving Data Indication Messages ... 42
3.2.5.6 Receiving Non-TURN Data from the Server ... 42
3.2.6 Timer Events ... 42
3.2.7 Other Local Events ... 42
3.3 Server Details ... 42
3.3.1 Abstract Data Model ... 42
3.3.2 Timers ... 42
3.3.3 Initialization ... 43
3.3.4 Higher-Layer Triggered Events ... 43
3.3.5 Message Processing Events and Sequencing Rules ... 43
3.3.5.1 Receiving Allocate Request Messages ... 43
3.3.5.2 Receiving Send Request Messages ... 45
3.3.5.3 Receiving Set Active Destination Request Messages ... 45
3.3.5.4 Receiving Data and Connections on the Allocated Transport Address ... 46
3.3.5.5 Receiving Non-TURN Data from the Client ... 46
3.3.5.6 Receiving Multiplexed TURN Encapsulated Data from the Client ... 46
3.3.6 Timer Events ... 46
3.3.7 Other Local Events ... 46

4 Protocol Examples ... 47

5 Security ... 51
5.1 Security Considerations for Implementers ... 51
5.2 Index of Security Parameters ... 51

6 Appendix A: Product Behavior ... 52

7 Change Tracking ... 56

8 Index ... 57

1 Introduction

This protocol specifies proprietary extensions to the Traversal Using Relay NAT (TURN) protocol. TURN is an Internet Engineering Task Force (IETF) draft proposal designed to provide a mechanism to enable a user behind a network address translation (NAT) to acquire a transport address from a public server and to use the allocated transport address to receive data from a selected peer.

This protocol is used as part of the Interactive Connectivity Establishment (ICE) Extensions protocol, as described in [MS-ICE] and [MS-ICE2].

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

200 OK: A response to indicate that the request has succeeded.

allocated transport address: A transport address that is allocated by a Traversal Using Relay NAT (TURN) server in response to an Allocate request from a TURN client. The TURN server obtains the transport address from a network interface that is connected to the Internet. The transport address has the same transport protocol over which the Allocate request was received; a request that is received over TCP returns a TCP allocated transport address. Also referred to as an allocated address.

authentication: The act of proving an identity to a server while providing key material that binds the identity to subsequent communications.

Coordinated Universal Time (UTC): A high-precision atomic time standard that approximately tracks Universal Time (UT). It is the basis for legal, civil time all over the Earth. Time zones around the world are expressed as positive and negative offsets from UTC. In this role, it is also referred to as Zulu time (Z) and Greenwich Mean Time (GMT). In these specifications, all references to UTC refer to the time at UTC-0 (or GMT).

digest: The fixed-length output string from a one-way hash function that takes a variable-length input string and is probabilistically unique for every different input string. Also, a cryptographic checksum of a data (octet) stream.

error response message: A Traversal Using Relay NAT (TURN) message that is sent from a protocol server to a protocol client in response to a request message. It is sent when an error occurs during processing of a request message.

Hash-based Message Authentication Code (HMAC): A mechanism for message **authentication** using cryptographic hash functions. HMAC can be used with any iterative cryptographic hash function (for example, **MD5** and **SHA-1**) in combination with a secret shared key. The cryptographic strength of HMAC depends on the properties of the underlying hash function.

Hypertext Transfer Protocol Secure (HTTPS): An extension of HTTP that securely encrypts and decrypts web page requests. In some older protocols, "Hypertext Transfer Protocol over Secure Sockets Layer" is still used (Secure Sockets Layer has been deprecated). For more information, see [SSL3] and [RFC5246].

Interactive Connectivity Establishment (ICE): A methodology that was established by the Internet Engineering Task Force (IETF) to facilitate the traversal of network address translation (NAT) by media.

Internet Protocol version 4 (IPv4): An Internet protocol that has 32-bit source and destination addresses. IPv4 is the predecessor of IPv6.

Internet Protocol version 6 (IPv6): A revised version of the Internet Protocol (IP) designed to address growth on the Internet. Improvements include a 128-bit IP address size, expanded routing capabilities, and support for **authentication** and privacy.

INVITE: A **Session Initiation Protocol (SIP)** method that is used to invite a user or a service to participate in a session.

long-term credentials: A set of user-authentication credentials that consist of a user name and password, and are used by a protocol client to authenticate with a protocol server.

MD5: A one-way, 128-bit hashing scheme that was developed by RSA Data Security, Inc., as described in [RFC1321].

Multiplexed TURN: An extension to the TURN protocol that enables a TURN server to multiplex traffic from multiple TURN clients over the same UDP or TCP port.

network address translation (NAT): The process of converting between IP addresses used within an intranet, or other private network, and Internet IP addresses.

nonce: A number that is used only once. This is typically implemented as a random number large enough that the probability of number reuse is extremely small. A nonce is used in authentication protocols to prevent replay attacks. For more information, see [RFC2617].

protocol client: An endpoint that initiates a protocol.

public address: An IPv4 or IPv6 address that is on the Internet.

reflexive transport address: A **transport address** that is given to a protocol client and identifies the public address of that client as seen by a protocol server. The address is communicated to the protocol client through the XOR MAPPED ADDRESS attribute in an allocate response message.

request message: A **Traversal Using Relay NAT (TURN)** message that is sent from a protocol client to a protocol server.

response message: A Traversal Using Relay NAT (TURN) message that is sent from a protocol server to a protocol client in response to a request message. It is sent when the request message is handled successfully by the protocol server.

Secure Sockets Layer (SSL): A security protocol that supports confidentiality and integrity of messages in client and server applications that communicate over open networks. SSL uses two keys to encrypt data-a public key known to everyone and a private or secret key known only to the recipient of the message. SSL supports server and, optionally, client **authentication** using X.509 certificates. For more information, see [X509]. The SSL protocol is precursor to **Transport Layer Security (TLS)**. The TLS version 1.0 specification is based on SSL version 3.0 [SSL3].

Session Description Protocol (SDP): A protocol that is used for session announcement, session invitation, and other forms of multimedia session initiation. For more information see [MS-SDP] and [RFC3264].

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].

SHA-1: An algorithm that generates a 160-bit hash value from an arbitrary amount of input data, as described in [RFC3174]. SHA-1 is used with the Digital Signature Algorithm (DSA) in the Digital Signature Standard (DSS), in addition to other algorithms and standards.

SHA-256: An algorithm that generates a 256-bit hash value from an arbitrary amount of input data, as described in [FIPS180-2].

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.

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 address: A 3-tuple that consists of a port, an IPv4 or IPV6 address, and a transport protocol of User Datagram Protocol (UDP) or Transmission Control Protocol (TCP).

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.

Traversal Using Relay NAT (TURN): A protocol that is used to allocate a public IP address and port on a globally reachable server for the purpose of relaying media from one endpoint to another endpoint.

TURN client: An endpoint that generates **Traversal Using Relay NAT (TURN)** request messages.

TURN server: An endpoint that receives **Traversal Using Relay NAT (TURN)** request messages and sends TURN response messages. The protocol server acts as a data relay, receiving data on the public address that is allocated to a protocol client and forwarding that data to the client.

type-length-value (TLV): A method of organizing data that involves a Type code (16-bit), a specified length of a Value field (16-bit), and the data in the Value field (variable).

User Datagram Protocol (UDP): The connectionless protocol within TCP/IP that corresponds to the transport layer in the ISO/OSI reference model.

UTF-8: A byte-oriented standard for encoding Unicode characters, defined in the Unicode standard. Unless specified otherwise, this term refers to the UTF-8 encoding form specified in [UNICODE5.0.0/2007] section 3.9.

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.

[IETFDRAFT-STUN-02] Rosenberg, J., Huitema, C., and Mahy, R., "Simple Traversal of UDP Through Network Address Translators (NAT) (STUN)", draft-ietf-behave-rfc3489bis-02, July 2005, http://tools.ietf.org/html/draft-ietf-behave-rfc3489bis-02

[IETFDRAFT-TURN-08] Rosenberg, J., Mahy, R., and Huitema, C., "Traversal Using Relay NAT (TURN)", draft-rosenberg-midcom-turn-08, September 2005, http://tools.ietf.org/html/draft-rosenberg-midcom-turn-08

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992, http://www.ietf.org/rfc/rfc1321.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.rfc-editor.org/rfc/rfc2246.txt

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

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.rfc-editor.org/rfc/rfc2818.txt

[RFC3489] Rosenberg, J., Weinberger, J., Huitema, C., and Mahy, R., "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003, http://www.ietf.org/rfc/rfc3489.txt

[RFC4346] Dierks, T., and Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006, http://www.ietf.org/rfc/rfc4346.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

[RFC6156] Camarillo, G., Novo, O., and Perreault, S. Ed., "Traversal Using Relays around NAT (TURN) Extension for IPv6", April 2011, http://www.ietf.org/rfc/rfc6156.txt

1.2.2 Informative References

[MS-AVEDGEA] Microsoft Corporation, "Audio Video Edge Authentication Protocol".

[MS-ICE2] Microsoft Corporation, "Interactive Connectivity Establishment (ICE) Extensions 2.0".

[MS-ICE] Microsoft Corporation, "Interactive Connectivity Establishment (ICE) Extensions".

[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

[RFC4566] Handley, M., Jacobson, V., and Perkins, C., "SDP: Session Description Protocol", RFC 4566, July 2006, http://www.ietf.org/rfc/rfc4566.txt

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980, http://www.rfc-editor.org/rfc/rfc768.txt

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

[TURN-01] Rosenberg, J., Mahy, R., and Huitema, C., "Obtaining Relay Addresses from Simple Traversal of UDP Through NAT (STUN)", draft-ietf-behave-turn-01, February 2006, http://tools.ietf.org/wg/behave/draft-ietf-behave-turn/draft-ietf-behave-turn-01.txt

[TURN-05] Rosenberg, J., Mahy, R., Matthews, P., and Wing, D., "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", draft-ietf-behave-turn-05, November 2007, http://tools.ietf.org/wg/behave/draft-ietf-behave-turn/draft-ietf-behave-turn-05.txt

1.3 Overview

The **Traversal Using Relay NAT (TURN)** protocol, as described in [IETFDRAFT-TURN-08], enables a **TURN client** located on a private network behind one or more **network address translation (NAT)** to allocate a **transport address** from a **TURN server** that is sitting on the Internet. This **allocated transport address** can be used for receiving data from a peer. The peer itself could be on a private network behind a NAT or it could have a **public address**.

A typical deployment, supported by the TURN protocol and this extension, where a protocol client is behind a NAT and is communicating with a peer on the Internet, is shown in the following figure.

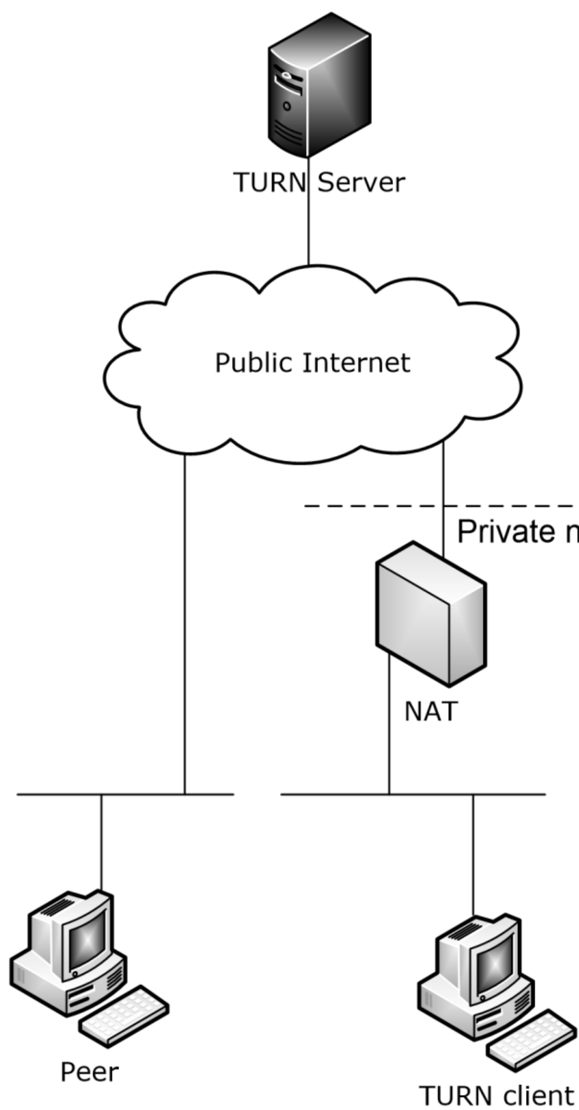

Figure 1: A TURN client communicating with a public peer

When a protocol client needs a public address to send data to or receive data from a peer, it sends an **Allocate** **request message** to the TURN server. This request is authenticated by the TURN server through a **digest** challenge mechanism. Once the TURN server has authenticated the **Allocate** request, it returns an allocated address to the protocol client in an **Allocate** **response message**.

At this point the allocated address has been reserved by the protocol client. It cannot be used to receive data from a peer until the protocol client attempts to send data to the peer by encapsulating the data in a **Send** request message. The **Send** request message serves two functions:

- The TURN server relays the data contained in the message to the peer identified by the **Destination** attribute.
- Permissions are set on the allocated address in a way that data arriving on the allocated address from the peer is relayed to the protocol client in a **Data Indication** message.

If the protocol client needs to communicate with more than one peer, it can send an additional **Send** request message to each peer.

Once the permissions have been set for a peer, any data received on the allocated address from that peer is relayed back to the protocol client encapsulated in a **Data Indication** message. This message includes the **Remote Address** attribute that identifies the peer that originated the data.

If the protocol client decides to communicate with a preferred peer, it can send a **Set Active Destination** request message to the TURN server. The TURN server acknowledges the protocol client's request by responding with a **Set Active Destination** response message. This allows the protocol client and TURN server to stop using **Send** request and **Data Indication** messages to encapsulate data flowing end-to-end for this peer, thus making the data communication channel more efficient. The results are that all data that the TURN server receives from the protocol client that is not a TURN control message is relayed directly to the active peer. All data that the TURN server receives on the allocated address from the active peer is relayed directly to the protocol client. If the TURN server receives data from a peer other than the active peer but for which it has permissions, as set by the protocol client through an earlier **Send** request message, the TURN server relays the data encapsulating it in a **Data Indication** message.

The basic flow of TURN messages between a protocol client and a TURN server is shown in the following figure.

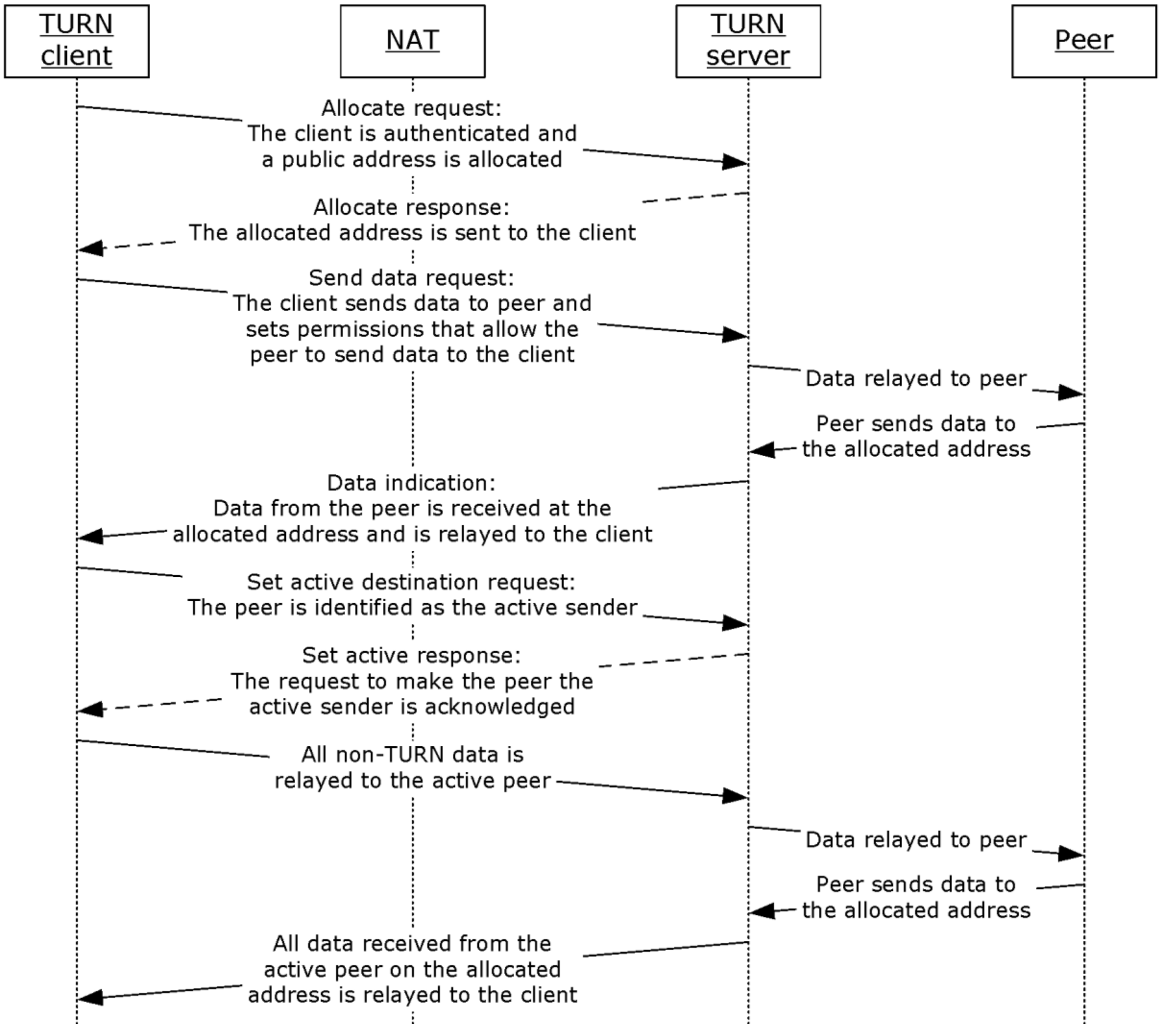

Figure 2: Basic flow of TURN messages

This protocol specifies proprietary extensions to the TURN protocol. These extensions include:

Authentication: This protocol does not use the **Shared Secret** request and **Shared Secret** response messages, as described in [IETFDRAFT-TURN-08] section 7.1, for **authentication** of the protocol client. Instead, this protocol uses **long-term credentials** and has extended the **Allocate** request and **Allocate** **error response message** processing, as described in [IETFDRAFT-TURN-08] section 7.2, to incorporate a digest challenge mechanism. This is specified in section 3.1.12. This extension is from the TURN draft version described in [TURN-05].

TCP Framing Header: A header has been added to this protocol for stream-based transports, so that TURN control messages are uniquely identifiable from end-to-end data. This is specified in section 2.1.4. This extension is from the TURN draft version described in [TURN-01].

Pseudo-TLS Session Establishment: This protocol includes a pseudo **Transport Layer Security (TLS)** **Client Hello** and **Server Hello** message exchange at the beginning of a **Transmission Control Protocol (TCP)** session to allow session establishment over TCP port 443, where a proxy or firewall might inspect the initial traffic for TLS packets. This is specified in section 2.1.1.

Client Versioning: An attribute has been added to this protocol to request messages to allow a protocol client to identify the protocol version it is using. This is specified in section 2.2.2.17.

Request Message Sequencing: An attribute that contains a sequence number has been added to the request messages in this protocol. This attribute helps prevent replay attacks and is specified in section 2.2.2.19.

No Support for Send Response or Send Error Response Messages: Support has been dropped for any response to the **Send** request message, as described in [IETFDRAFT-TURN-08] section 7.3, to streamline the **Send** request phase of a TURN session. This extension is from the TURN draft version described in [TURN-01].

XOR Mapped Address: This protocol uses the **XOR-MAPPED-ADDRESS** attribute from [IETFDRAFT-STUN-02] section 10.2.12 to inform the protocol client of the protocol client's **reflexive transport address**. This keeps intrusive NATs from rewriting the binary encoded IP address and port. This is specified in section 2.2.2.16.

Alternate Server Attribute: This protocol extends the use of the **Alternate Server** attribute and includes it in the **Allocate** error response message, error response code of **401 Unauthorized**, to convey the IP address of the TURN server to the protocol client. In some deployments, for example, a pool of TURN servers behind a load balancer that presents a virtual IP address, the protocol client needs to know the direct address of the TURN server with which it established a TURN session. This is specified in section 2.2.2.7.

Service Quality: An attribute has been added to this protocol to convey information about the data stream that the protocol client is intending to transfer over an allocated port. This is specified in section 2.2.2.20.

Request a specific IP Address Family: This protocol provides support for the **Requested Address Family** attribute from [RFC6156] section 4.1.1. This attribute is used in an **Allocate** request message to identify the IP address family to be allocated by the TURN server. This is specified in section 2.2.2.15.

Request both an IPv4 and IPv6 allocated addresses: This protocol provides support for allocating both IPv4 and IPv6 addresses from a TURN server configured to use both IPv4 and IPv6 networks.

HMAC SHA-256 support: This protocol provides support for using HMAC SHA-256 with the Message Integrity attribute specified in section 2.2.2.3. The use of HMAC SHA-256 provides enhanced security for TURN messages.

1.4 Relationship to Other Protocols

This protocol does not introduce any new protocol relationships beyond those described in [IETFDRAFT-TURN-08]. The **TURN** protocol, as described in [IETFDRAFT-TURN-08], is used to provide network connectivity and relies on either **User Datagram Protocol (UDP)**, as described in [RFC768], or **TCP**, as described in [RFC793], as a transport.

1.5 Prerequisites/Preconditions

It is assumed that the protocol client and **TURN server** have an **Internet Protocol version 4 (IPv4)** or **Internet Protocol version 6 (IPv6)** address with either **UDP** or **TCP** connectivity and that the protocol client knows the IPv4 or IPv6 address and port of the TURN server and a peer that it wants to communicate with. The TURN server is assumed to be ready to receive datagrams, in the case of UDP, or incoming connections, in the case of TCP, on the configured port.

It is also assumed that the TURN client has **long-term credentials** that it can use to authenticate with the TURN server. These credentials are acquired by communicating with a protocol TURN server that has implemented the protocol described in [MS-AVEDGEA].

1.6 Applicability Statement

This protocol does not change the applicability of the **TURN** protocol as it is described in [IETFDRAFT-TURN-08] section 4.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- **Supported Transports:** This protocol can be implemented over either **TCP** or **UDP** running on either **IPv4** or **IPv6**, as discussed in section 2.2.1.
- **Protocol Versions:** This protocol specifies a mechanism by which the protocol client and **TURN server** can explicitly indicate what version of the protocol is supported. The protocol client does this by including the **MS-Version** attribute in an **Allocate** **request message**. The TURN server does this by including the **MS-Version** attribute in an **Allocate** **response message**. The **MS-Version** attribute is specified in section 2.2.2.17.
- **Security and Authentication Methods:** This protocol supports **authentication** through **long-term credentials** supplied in the **Allocate** request message. This is specified in section 3.1.12.
- **Capability Negotiation:** This protocol does not have any capability negotiation constraints.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

This protocol uses the standard **UDP** and **TCP** ports from [IETFDRAFT-STUN-02]. It has no additional standard assignments.

Parameter	Value	Reference
UDP Port	443	[IETFDRAFT-STUN-02]
TCP Port	443	[IETFDRAFT-STUN-02]

2 Messages

2.1 Transport

This protocol can use either **UDP** or **TCP** running on either **IPv4** or **IPv6**<1> as a transport protocol. All message formats are specified as a UDP datagram and do not require any additional framing when sent over UDP. Transport over TCP requires additional framing, as specified in section 2.1.1 and section 2.1.4.

2.1.1 Pseudo-TLS over TCP

When **TCP** is used as a transport, the **TURN server** is deployed to listen on port 443, the **Secure Sockets Layer (SSL)**/**TLS** port. If a protocol client is attempting to communicate with a TURN server deployed in this fashion, it sends a pseudo-TLS message to the TURN server to begin the session. The pseudo-TLS messages are useful if a firewall or Web proxy, doing packet inspection for TLS messages, is sitting between the protocol client and TURN server. The TURN server MUST support pseudo-TLS.

The protocol client begins the exchange by sending the pseudo-TLS **ClientHello** message. If the protocol client sends this message, it MUST be the first message and the protocol client MUST NOT send any additional messages until the TURN server has responded with a pseudo-TLS **ServerHello** message followed by a pseudo-TLS **ServerHelloDone** message. If the TURN server receives a pseudo-TLS **ClientHello** message, it MUST respond with a **ServerHello** followed by a **ServerHelloDone** message. The **ServerHello** and **ServerHelloDone** messages MUST be sent in the same TLS record. These messages appear next in this protocol.

The **ClientHello**, **ServerHello**, and **ServerHelloDone** messages passed in the exchange are known as **Handshake** messages within the TLS record protocol. The TLS record protocol is described in [RFC2246] section 6, while **Handshake** messages are described in [RFC2246] section 7.3.

Pseudo-TLS record containing ClientHello message

<table>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>1
0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>2
0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>3
0</th><th>1</th></tr>
<tr><td colspan="8">Content Type</td><td colspan="8">Record Version Major</td><td colspan="8">Record Version Minor</td><td colspan="8">Record Length ...</td></tr>
<tr><td colspan="8">...</td><td colspan="8">Handshake Type</td><td colspan="16">Handshake Length ...</td></tr>
<tr><td colspan="8">...</td><td colspan="8">Handshake Version Major</td><td colspan="8">Handshake Version Minor</td><td colspan="8">Time Stamp ...</td></tr>
<tr><td colspan="24">...</td><td colspan="8">Random Value
(28 bytes)</td></tr>
<tr><td colspan="32">Random Value ...</td></tr>
<tr><td colspan="8">...</td><td colspan="8">Session ID Length</td><td colspan="16">Cipher Suites Length</td></tr>
<tr><td colspan="16">Cipher Suites</td><td colspan="8">Compression Methods Length</td><td colspan="8">Compression Methods</td></tr>
</table>

Content Type (1 byte): The **Record Layer** protocol type. This field MUST be set to "0x16" for the **Handshake**.

Record Version Major (1 byte): The **Major** version of TLS for this record. This field MUST be set to "0x03" (TLS 1.0).

Record Version Minor (1 byte): The **Minor** version of TLS for this record. This field MUST be set to "0x01" (TLS 1.0).

Record Length (2 bytes): The length of the TLS record. This field MUST be set to "0x00 0x2D".

Handshake Type (1 byte): The **Handshake** message type. This field MUST be set to "0x01" for a **ClientHello** message.

Handshake Length (3 bytes): The length of the **Handshake** message. This field MUST be set to "0x00 0x00 0x29".

Handshake Version Major (1 byte): The **Major** version of TLS for the message. This field MUST be set to "0x03" (TLS 1.0).

Handshake Version Minor (1 byte): The **Minor** version of TLS for the message. This field MUST be set to "0x01" (TLS 1.0).

Time Stamp (4 bytes): The current time and date in seconds since midnight starting January 1, 1970, **Coordinated Universal Time (UTC)**, ignoring leap seconds. The protocol client SHOULD fill this field with the correct time. The TURN server SHOULD ignore this field.

Random Value (28 bytes): 28 bytes of randomly generated data.

Session ID Length (1 byte): The length of the session ID vector. This field MUST be set to "0x00".

Cipher Suites Length (2 bytes): The length of the cipher suite vector. This field MUST be set to "0x00 0x02".

Cipher Suites (2 bytes): The cipher suite the protocol client is requesting. This field MUST be set to "0x00 0x18".

Compression Methods Length (1 byte): The length of the compression method vector. This field MUST be set to "0x01".

Compression Methods (1 byte): The compression methods that the protocol client is requesting. This field MUST be set to "0x00".

Pseudo-TLS record containing ServerHello and ServerHelloDone messages

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="8">Content Type</td><td colspan="8">Record Version Major</td><td colspan="8">Record Version Minor</td><td colspan="8">Record Length …</td></tr>
<tr><td colspan="8">…</td><td colspan="8">Handshake Type</td><td colspan="16">Handshake Length …</td></tr>
<tr><td colspan="8">…</td><td colspan="8">Handshake Version Major</td><td colspan="8">Handshake Version Minor</td><td colspan="8">Time Stamp …</td></tr>
<tr><td colspan="24">…</td><td colspan="8">Random Value
(28 bytes)</td></tr>
<tr><td colspan="32">Random Value …</td></tr>
<tr><td colspan="8">…</td><td colspan="8">Session ID Length</td><td colspan="16">Session ID (32 bytes)</td></tr>
</table>

...			
Cipher Suites		Compression Methods	Handshake Type
Handshake Length			

Content Type (1 byte): The **Record Layer** protocol type. This field MUST be set to "0x16" for the **Handshake**.

Record Version Major (1 byte): The **Major** version of TLS for this record. This field MUST be set to "0x03" (TLS 1.0).

Record Version Minor (1 byte): The **Minor** version of TLS for this record. This field MUST be set to "0x01" (TLS 1.0).

Record Length (2 bytes): The length of the TLS record. This field MUST be set to "0x00 0x4E".

Handshake Type (1 byte): The **Handshake** message type. This field MUST be set to "0x02" for a Server Hello message.

Handshake Length (3 bytes): The length of the **Handshake** message. This field MUST be set to "0x00 0x00 0x46".

Handshake Version Major (1 byte): The **Major** version of TLS for the message. This field MUST be set to "0x03" (TLS 1.0).

Handshake Version Minor (1 byte): The **Minor** version of TLS for the message. This field MUST be set to "0x01" (TLS 1.0).

Time Stamp (4 bytes): The current time and date in seconds since midnight starting January 1, 1970, UTC, ignoring leap seconds. The TURN server SHOULD fill this field with the correct time. The protocol client SHOULD ignore this field.

Random Value (28 bytes): 28 bytes of randomly generated data.

Session ID Length (1 byte): The length of the session ID vector. This field MUST be set to "0x20".

Session ID (32 bytes): 32 bytes used to identify the TLS session. The TURN server does not track the TLS session id, so the protocol client SHOULD ignore this field.

Cipher Suites (2 bytes): The cipher suite the TURN server has selected. This field MUST be set to "0x00 0x18".

Compression Methods (1 byte): The compression method that the TURN server has selected. This field MUST be set to "0x00".

Handshake Type (1 byte): The **Handshake** message type. This field MUST be set to "0x0E" for a **ServerHelloDone** message.

Handshake Length (3 bytes): The length of the **Handshake** message. This field MUST be set to "0x00 0x00 0x00".

2.1.2 TLS over TCP

When **TCP** is used as a transport, the **TURN server** is deployed to listen on port 443, the **Secure Sockets Layer (SSL)**/**TLS** port. If a protocol client is attempting to communicate with a TURN server deployed in this fashion, it MAY choose to use the standard Transport Layer Security (TLS) instead of the pseudo-TLS protocol described in section 2.1.1. In this case, cryptographic parameters negotiation is carried out in accordance with [RFC2246], [RFC4346] or [RFC5246], after which the communication proceeds as described in section 2.1.4.

2.1.3 HTTPS over TCP

When **TCP** is used as a transport, the **TURN server** is deployed to listen on port 443, the **Secure Sockets Layer (SSL)**/**TLS** port. If a protocol client is attempting to communicate with a TURN server deployed in this fashion, it MAY choose to encapsulate all the protocol traffic into **HTTPS** requests in accordance with [RFC2818] and [RFC2616].

2.1.4 TCP

When **TCP** is used as a transport for this protocol, it requires an additional framing so that the **TURN** control messages can be identified within the TCP data stream. This additional framing consists of a header followed by the TURN datagram. This framing header MUST be used for all TURN messages and data sent to the **TURN server**. The framing header MUST NOT be used for the pseudo-TLS session establishment messages.

TCP Framing Header

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Type								Reserved								Length															

Type (1 byte): The data contained in this frame is a TURN control message or end-to-end data. This MUST be set to "0x02" to identify a TURN control message or it MUST be set to "0x03" to identify end-to-end data.

Reserved (1 byte): Not used and MUST be set to zero.

Length (2 bytes): The number of bytes of the frame following immediately after the **Length** field itself.

2.1.5 UDP

When **UDP** is used as a transport for this protocol no transport specific framing is required.

2.2 Message Syntax

2.2.1 Message Header

All **TURN** messages consist of a 20 byte TURN header followed by 1 or more TURN attributes. The TURN attributes are **type-length-value (TLV)** encoded. The TURN message header is the same as the message header specified in [IETFDRAFT-STUN-02] section 10.1. All TURN messages begin with any necessary transport specific framing, as specified in section 2.1, followed by this header.

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**1 0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**2 0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**3 0**	**1**
00		Message Type														Message Length															
Transaction ID (16 bytes)																															

Message Type (16 bits): The type of TURN message. The most significant two bits of this field MUST be set to zero so that TURN packets can be differentiated from other protocols. The TURN message types are specified in [IETFDRAFT-TURN-08] section 9.1. The TURN message types supported in this extension:

- "0x0003": Allocate request
- "0x0103": Allocate response
- "0x0113": Allocate error response
- "0x0004": Send request
- "0x0115": Data Indication
- "0x0006": Set Active Destination request
- "0x0106": Set Active Destination response
- "0x0116": Set Active Destination error response

The following TURN message types are not supported by this extension and the **TURN server** MUST NOT send them:

- "0x0104": Send request response
- "0x0114": Send request error response

In addition, this extension does not support the shared secret **authentication** mechanism. The following shared secret messages, specified in [RFC3489] section 11.1, MUST NOT be used by either the protocol client or TURN server:

- "0x0002": Shared Secret request
- "0x0102": Shared Secret response
- "0x0112": Shared Secret error response

Message Length (16 bits): The length, in bytes, of the message. This length does not include the 20 byte header.

Transaction ID (16 bytes): A 128 bit identifier used to uniquely identify the TURN transaction. A **Transaction ID**, created by the protocol client and used in a **request message**, is echoed from the TURN server back to the protocol client in the subsequent response or **error response message**. The protocol client MUST choose a new **Transaction ID** for each new transaction. A new **Transaction ID** SHOULD be uniformly and randomly distributed between 0 and 2^128 -1. If the protocol client is retransmitting a request message, it MUST use the same **Transaction ID** as it used in the original request message.

2.2.2 Message Attribute

After the **TURN** header, all TURN messages consist of 1 or more attributes. All attributes MUST be **TLV** encoded and have the same format as specified in [IETFDRAFT-STUN-02] section 10.2. The **Magic Cookie** attribute MUST be the first attribute in all TURN messages.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Attribute Type																Attribute Length															

Value (variable)
...

Attribute Type (2 bytes): The TURN attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2.

The following mandatory attribute types are supported in this extension. Any other attributes from the mandatory attribute space MUST generate an error response with an error response code of **Unknown Attribute**.

- "0x0001": Mapped Address
- "0x0006": Username
- "0x0008": Message Integrity
- "0x0009: Error Code
- "0x000A": Unknown Attributes
- "0x000D": Lifetime
- "0x000E": Alternate Server
- "0x000F": Magic Cookie
- "0x0010": Bandwidth
- "0x0011": Destination Address
- "0x0012": Remote Address
- "0x0013": Data
- "0x0014": Nonce
- "0x0015": Realm
- "0x0017": Requested Address Family<2>

The following optional attributes are also supported in this extension. Any other attributes from the optional attribute space SHOULD be ignored.

- "0x8008": MS-Version
- "0x8020": XOR Mapped Address
- “0x8032”: MS-Alternate Host Name
- "0x8050": MS-Sequence Number
- "0x8055": MS-Service Quality<3>
- "0x8090": MS-Alternate Mapped Address<4>
- "0x8095": Multiplexed TURN Session ID<5>

Attribute Length (2 bytes): The length of bytes of the **Value** data following the **Attribute Length** field itself.

Value (variable): Variable-length field that contains information dependent on the attribute type.

2.2.2.1 Mapped Address

This section follows the product behavior as described in product behavior note <6>.

The **Mapped Address** attribute is specified in [IETFDRAFT-STUN-02] section 10.2.1. This attribute is used to identify the public **transport address** allocated by the **TURN server** on behalf of the protocol client.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Attribute Type																Attribute Length															
Reserved								Family								Port															
IP Address																															

Attribute Type (2 bytes): The TURN attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x0001".

Attribute Length (2 bytes): Set to "0x0008" (8) for an **IPv4** address or "0x0014" (20) for an **IPv6** address.

Reserved (1 byte): The first 8 bits are used for alignment purposes and are ignored.

Family (1 byte): The address family of the Address. It MUST have the value "0x01" for an IPv4 address or "0x02" for an IPv6 address.

Port (2 bytes): A network byte ordered representation of the mapped port.

IP Address (4 or 16 bytes): The IPv4 or IPv6 mapped address.

If the address is IPv4 (**Family** is set to "0x01") this is the network byte ordered 32-bit (4 byte) IPv4 address.

If the address is IPv6 (**Family** is set to "0x02") this is the network byte ordered 128-bit (16 byte) IPv6 address.

2.2.2.2 Username

The **Username** attribute is specified in [IETFDRAFT-STUN-02] section 10.2.6. This attribute is used to identify the user name part of the protocol client's **long-term credentials** with the **TURN server**. The TURN server MUST know how to validate this user name and it MUST be able to retrieve the password associated with this user name. If the TURN server does not know the user name, it MUST fail the authenticated request with an appropriate **error response message** that includes an **Error Code** attribute with an error response value of **436 Unknown User**.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Attribute Type																Attribute Length															
Username																															

...

Attribute Type (2 bytes): The TURN attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x0006".

Attribute Length (2 bytes): The length of **Username** in bytes.

Username (variable): The value of the opaque, variable length **Username**.

2.2.2.3 Message Integrity

This section follows the product behavior as described in product behavior note <7>.

The **Message Integrity** attribute is specified in [IETFDRAFT-STUN-02] section 10.2.8. This attribute is used by the protocol in all authenticated **request messages** and **response messages**. This attribute MUST be the last attribute in a **TURN** message.

There are two possible algorithms that can be used to create the **Hash-based Message Authentication Code (HMAC)**, HMAC **SHA-1** or HMAC **SHA-256**. The protocol client and **TURN server** indicate support for the two algorithms through the value of the **MS-Version** attribute. If the **MS-Version** attribute is absent or if the version is less than "0x03" (03) only HMAC SHA-1 is supported. If the version is equal to or greater than "0x03" (03) both HMAC SHA-1 and HMAC SHA-256 are supported. If both the protocol client and TURN server advertise a version equal to or greater than "0x03" (03) the HMAC SHA-256 algorithm MUST be used. If either the protocol client or the TURN server advertise a version less than "0x03" (03) the HMAC SHA-1 algorithm MUST be used.

The algorithm for using HMAC SHA-1 to create the hash value is specified as part of [IETFDRAFT-TURN-08] section 7.1. This algorithm is used to create the hash for outbound messages and to verify the hash of inbound messages. The algorithm summary is as follows:

- The text used as input to the HMAC is the TURN message, including the TURN message header, up to and including the attribute preceding the **Message Integrity** attribute. The text is padded with zeroes to be a multiple of 64 bytes.
- As shown in the following example, the key used in the HMAC is the 128-bit **digest** resulting from using the **MD5** message digest algorithm, as specified in [RFC1321], on the concatenation of the long-term user name, the value of the **Realm** attribute and the long-term password, each separated by a ":".

```
Key = MD5(Username || ":" || Realm || ":" || password)
Hash = HMAC-SHA1(Key, Text)
```

The algorithm for using HMAC SHA-256 to create the hash value is shown in the following summary. This algorithm is used to create the hash for outbound messages and to verify the hash of inbound messages. The algorithm summary is as follows:

- The text used as input to the HMAC is the TURN message, including the TURN message header, up to and including the attribute preceding the **Message Integrity** attribute. The text is padded with zeroes to be a multiple of 64 bytes.
- As shown in the following example, the key used in the HMAC is the 256-bit hash resulting from using the following two step key derivation procedure.
  - The first step produces the initial key, K, from the 256-bit hash output from the HMAC SHA-256 algorithm using the value of the **Nonce** attribute as the key and the long-term password as text.

- The second step produces the final key, Key, from the 256-bit hash output from the HMAC SHA-256 algorithm using the initial key, K, as the key and the following concatenated fields as the text:
  - 8-bit value set to "0x01" (1)
  - ASCII encoded binary string "TURN"
  - 8-bit value set "0x00" (0)
  - Value of the **Username** attribute
  - Value of the **Realm** attribute
  - 32-bit value set to (0x00000100) (256) in network byte order

```
K = HMAC-SHA256(Nonce, password)
Key = HMAC-SHA256(K, 0x01:8 || "TURN" || 0x00:8 || Username || Realm || 0x00000100:32)
Hash = HMAC-SHA256(Key, Text)
```

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>20</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>30</td><td>1</td></tr>
<tr><td colspan="16">Attribute Type</td><td colspan="16">Attribute Length</td></tr>
<tr><td colspan="32">HMAC Hash</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

Attribute Type (2 bytes): The TURN attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x0008".

Attribute Length (2 bytes): This field contains the length, in bytes, of the **HMAC Hash** field. If the HMAC hash algorithm used is HMAC SHA-1 this is set to "0x0014" (20). If the HMAC hash algorithm used is HMAC SHA-256 this is set to "0x0020" (32).

HMAC Hash (20 or 32 bytes): The output of the HMAC hash algorithm. If the HMAC hash algorithm used is HMAC SHA-1 this will be the 20 byte hash output. If the HMAC hash algorithm used is HMAC SHA-256 this will be the 32 byte hash output.

2.2.2.4 Error Code

The **Error Code** attribute is specified in [IETFDRAFT-STUN-02] section 10.2.9. For error response codes and suggested text for the associated **Reason Phrase**, see [IETFDRAFT-STUN-02] section 10.2.9 and [IETFDRAFT-TURN-08] section 9.2.10.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>20</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>30</td><td>1</td></tr>
<tr><td colspan="16">Attribute Type</td><td colspan="16">Attribute Length</td></tr>
<tr><td colspan="24">Reserved</td><td colspan="3">Class</td><td colspan="1">A</td><td colspan="4">B</td></tr>
</table>

...

Attribute Type (2 bytes): The **TURN** attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x0009".

Attribute Length (2 bytes): This field contains the length, in bytes, of the following fields.

Reserved (3 bytes): Set to zero.

Class (3 bits): The 100s digit of the response code. The value MUST be between 1 and 6, as specified in [IETFDRAFT-STUN-02] section 10.2.9. The supported **Error Class** values are specified in [IETFDRAFT-STUN-02] section 10.2.9 and [IETFDRAFT-TURN-08] section 9.2.10.

A - *Number (1 bit): The response code modulo 100. The value MUST be between 0 and 99, as specified in [IETFDRAFT-STUN-02] section 10.2.9. The supported **Error Numbers** are specified in [IETFDRAFT-STUN-02] section 10.2.9 and [IETFDRAFT-TURN-08] section 9.2.10.

B - Reason Phrase (variable): Textual description of the error that has occurred. The phrase is encoded in **UTF-8**, as specified in [IETFDRAFT-STUN-02] section 10.2.9. Recommended reason phrases for various errors are specified in [IETFDRAFT-STUN-02] section 10.2.9 and [IETFDRAFT-TURN-08] section 9.2.10.

2.2.2.5 Unknown Attributes

The **Unknown Attributes** attribute is specified in [IETFDRAFT-STUN-02] section 10.2.10. This attribute is present only in an **error response message** that contains an error code of 420. The attribute contains a list of 16-bit values, each representing the mandatory attribute type that was not understood by the **TURN server**.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Attribute Type																Attribute Length															
Attribute 1 Type																Attribute 2 Type															

Attribute Type (2 bytes): The **TURN** attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x000A".

Attribute Length (2 bytes): This field contains the length, in bytes, of the following fields.

Attribute 1 Type (2 bytes): Type of first attribute.

Attribute 2 Type (2 bytes): Type of second attribute.

2.2.2.6 Lifetime

The **Lifetime** attribute is specified in [IETFDRAFT-TURN-08] section 9.2.1.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Attribute Type																Attribute Length															
Lifetime																															

Attribute Type (2 bytes): The **TURN** attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x000D".

Attribute Length (2 bytes): This field contains the length, in bytes, of the **Lifetime** field. Set to "0x0004" (4).

Lifetime (4 bytes): Number of seconds the **TURN server** maintains an allocated address in the absence of data traffic from the protocol client to the TURN server. If the value is zero in a subsequent **Allocate** **request message**, the TURN session associated with this protocol client MUST be torn down.

2.2.2.7 Alternate Server

This section follows the product behavior as described in product behavior note <8>.

The **Alternate Server** attribute is specified in [IETFDRAFT-TURN-08] section 9.2.2. The alternate **TURN server** is used in two **error response messages**:

- An error response with an error code of **401 Unauthorized**. In this case, the value of the **Alternate Server** attribute SHOULD be the public **transport address** of the TURN server from which the response originated. If the transport is **UDP**, the protocol client MUST use the transport address from the **Alternate Server** attribute as the destination for the next **Allocate** **request message**.
- An error response with an error code of **300 Try Alternate**, which occurs when the TURN server does not have resources to satisfy an **Allocate** request. In this case the value of the **Alternate Server** attribute is another TURN server that had available resources for the **Allocate** request.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="16">Attribute Type</td><td colspan="16">Attribute Length</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="8">Family</td><td colspan="16">Port</td></tr>
<tr><td colspan="32">IP Address</td></tr>
</table>

Attribute Type (2 bytes): The **TURN** attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x000E".

Attribute Length (2 bytes): This field contains the length, in bytes, of the following fields. Set to "0x0008" (8) for an **IPv4** address or "0x0014" (20) for an **IPv6** address.

Reserved (1 byte): The first 8 bits are used for alignment purposes and are ignored.

Family (1 byte): The address family of the Address. It MUST have the value "0x01" for an IPv4 address or "0x02" for an IPv6 address.

Port (2 bytes): A network-byte-ordered representation of the TURN server port.

IP Address (4 bytes or 16 bytes): The IPv4 or IPv6 address of the TURN server.

If the address is IPv4 (**Family** is set to "0x01") this is the network byte ordered 32-bit (4 byte) IPv4 address.

If the address is IPv6 (**Family** is set to "0x02") this is the network byte ordered 128-bit (16 byte) IPv6 address.

2.2.2.8 Magic Cookie

The **Magic Cookie** attribute is specified in [IETFDRAFT-TURN-08] section 9.2.3. This attribute MUST be the first attribute following the **TURN** message header in all TURN messages. It is used to disambiguate TURN messages from data traffic.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="16">Attribute Type</td><td colspan="16">Attribute Length</td></tr>
<tr><td colspan="32">Magic Cookie</td></tr>
</table>

Attribute Type (2 bytes): The TURN attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x000F".

Attribute Length (2 bytes): This field contains the length, in bytes, of the **Magic Cookie** field. Set to "0x0004" (4).

Magic Cookie (4 bytes): Set to "0x72c64bc6".

2.2.2.9 Bandwidth

The **Bandwidth** attribute is specified in [IETFDRAFT-TURN-08] section 9.2.4.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="16">Attribute Type</td><td colspan="16">Attribute Length</td></tr>
<tr><td colspan="32">Bandwidth</td></tr>
</table>

Attribute Type (2 bytes): The **TURN** attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x0010".

Attribute Length (2 bytes): This field contains the length, in bytes, of the **Bandwidth** field. Set to "0x0004" (4).

Bandwidth (4 bytes): The **Bandwidth** value represents the peak bandwidth, in kilobits per second.

2.2.2.10 Destination Address

This section follows the product behavior as described in product behavior note <9>.

The **Destination Address** attribute is specified in [IETFDRAFT-TURN-08] section 9.2.5.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="16">Attribute Type</td><td colspan="16">Attribute Length</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="8">Family</td><td colspan="16">Port</td></tr>
<tr><td colspan="32">IP Address</td></tr>
</table>

Attribute Type (2 bytes): The **TURN** attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x0011".

Attribute Length (2 bytes): Length of the following fields. Set to "0x0008" (8) for an **IPv4** address or "0x0014" (20) for an **IPv6** address.

Reserved (1 byte): The first 8 bits are used for alignment purposes and are ignored.

Family (1 byte): The address family of the IP Address. It MUST have the value "0x01" for an IPv4 address or "0x02" for an IPv6 address. If the value is anything other than 0x01 or 0x02, the attribute MUST be silently ignored.

Port (2 bytes): A network byte ordered representation of the mapped port.

IP Address (4 bytes or 16 bytes): The IPv4 or IPv6 destination address.

If the address is IPv4 (**Family** is set to "0x01") this is the network byte ordered 32-bit (4 byte) IPv4 address.

If the address is IPv6 (**Family** is set to "0x02") this is the network byte ordered 128-bit (16 byte) IPv6 address.

2.2.2.11 Remote Address

This section follows the product behavior as described in product behavior note <10>.

The **Remote Address** attribute is specified in [IETFDRAFT-TURN-08] section 9.2.6.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Attribute Type																Attribute Length															
Reserved								Family								Port															
IP Address																															

Attribute Type (2 bytes): The **TURN** attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x0012".

Attribute Length (2 bytes): Length of the following fields. Set to "0x0008" (8) for an **IPv4** address or "0x0014" (20) for an **IPv6** address.

Reserved (1 byte): The first 8 bits are used for alignment purposes and are ignored.

Family (1 byte): The address family of the IP Address. It MUST have the value "0x01" for an IPv4 address or "0x02" for an IPv6 address. If the value is anything other than 0x01 or 0x02, the attribute MUST be silently ignored.

Port (2 bytes): A network-byte-ordered representation of the mapped port.

IP Address (4 bytes or 16 bytes): The IPv4 or IPv6 remote address.

If the address is IPv4 (**Family** is set to "0x01") this is the network byte ordered 32-bit (4 byte) IPv4 address.

If the address is IPv6 (**Family** is set to "0x02") this is the network byte ordered 128-bit (16 byte) IPv6 address.

2.2.2.12 Data

The **Data** attribute is specified in [IETFDRAFT-TURN-08] section 9.2.7.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="16">Attribute Type</td><td colspan="16">Attribute Length</td></tr>
<tr><td colspan="32">Data</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

Attribute Type (2 bytes): The **TURN** attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x0013".

Attribute Length (2 bytes): This field contains the length, in bytes, of the **Data** field.

Data (variable): Raw data that is to be relayed between a protocol client and a peer.

2.2.2.13 Nonce

The **Nonce** attribute is specified in [IETFDRAFT-TURN-08] section 9.2.8. The value of the **Nonce** attribute is used for replay protection and SHOULD be encoded by the **TURN server** in such a way as to indicate duration of validity or the protocol client identity for which it is valid. This protocol uses the attribute in the digest challenge extension specified in section 3.1.12.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="16">Attribute Type</td><td colspan="16">Attribute Length</td></tr>
<tr><td colspan="32">Nonce</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

Attribute Type (2 bytes): The **TURN** attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x0014".

Attribute Length (2 bytes): The length of bytes of the **Nonce** field. The **Nonce** field length MUST NOT exceed 128 bytes.

Nonce (variable): Variable length of data used as a **nonce** value. This nonce value length MUST NOT exceed 128 bytes.

2.2.2.14 Realm

The **Realm** attribute is specified in [IETFDRAFT-TURN-08] section 9.2.9. The value of the **Realm** attribute SHOULD be the domain name of the provider of the **TURN server**. This protocol uses the attribute in the digest challenge extension specified in section 3.1.12. If the protocol client includes this attribute, the TURN server SHOULD use the specified **Realm** value in the digest challenge extension. If the protocol client does not include this attribute in the **request message**, the TURN server uses a default **Realm** value. The TURN server MUST include this attribute in the associated response and the **Realm** value MUST be the value that the TURN server used in the digest challenge extension.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Attribute Type																Attribute Length															
Realm																															
...																															

Attribute Type (2 bytes): The **TURN** attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x0015".

Attribute Length (2 bytes): The length of bytes of the **Realm** field. The **Realm** length MUST NOT exceed 128 bytes.

Realm (variable): Variable length of data used as the **Realm** value. **Realm** MUST NOT exceed 128 bytes.

2.2.2.15 Requested Address Family

This section follows the behavior described in product behavior note<11>

The **Requested Address Family** attribute is specified in [RFC6156] section 4.1.1. It is used by the protocol client to request an allocation of a specific IP address family type from the **TURN server**. This attribute SHOULD be included in the **Allocate request message** when the protocol client wants either an **IPv4** or an **IPv6** address to be allocated. The absence of this attribute in the **Allocate** request message indicates that the protocol client wants both an IPv4 and an IPv6 address to be allocated if the TURN server is so configured. If the protocol client wants both an IPv4 and an IPv6 address to be allocated it SHOULD NOT include this attribute in the **Allocate** request message.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Attribute Type																Attribute Length															
Family								Reserved																							

Attribute Type (2 bytes): The **TURN** attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x0017".

Attribute Length (2 bytes): The length of bytes of following fields. Set to "0x0004" (4).

Family (1 byte): The address family of the attribute. There are two values defined for this field specified in [IETFDRAFT-STUN-02] section 10.2.1: "0x01" for IPv4 addresses and 0x02 for IPv6 addresses.

Reserved (3 bytes): The 24 bits in the Reserved field MUST be set to zero by the client and MUST be ignored by the server.

2.2.2.16 XOR Mapped Address

This section follows the product behavior as described in the product behavior note <12>

The **XOR Mapped Address** attribute is specified in [IETFDRAFT-STUN-02] section 10.2.12. This protocol uses the **XOR Mapped Address** attribute to indicate to the protocol client its **reflexive transport address**. The protocol client can use this to help identify the type of **NAT** it is behind.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Attribute Type																Attribute Length															
Reserved								Family								X-Port															
X-IP Address																															

Attribute Type (2 bytes): The **TURN** attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x8020".

Attribute Length (2 bytes): Length of the following fields. Set to "0x0008" (8) for an **IPv4** address or "0x0014" (20) for an **IPv6** address.

Reserved (1 byte): The first 8 bits are used for alignment purposes and are ignored.

Family (1 byte): The address family of the IP Address. It MUST have the value "0x01" for an IPv4 address or "0x02" for an IPv6 address. If the value is anything other than 0x01 or 0x02, the attribute MUST be silently ignored.

X-Port (2 bytes): A network byte ordered representation of the source port that the **Allocate request message** was received from. This value is created from the exclusive-or of the source port with the most significant 16 bits of the **Transaction ID**. If the source port was "0x1122" (network byte order) and the most significant 16 bits of the Transaction ID was "0x4455" (network byte order), the resulting **X-Port** is 0x1122 ^ 0x4455 = 0x5577.

X-IP Address (4 bytes or 16 bytes): The client's IPv4 or IPv6 address.

If the address family is IPv4 (**Family** is set to "0x01"), this is the client's network byte ordered 32-bit (4 byte) IPv4 address. This value is created from the exclusive-or of the IP address with the most significant 32 bits of the **Transaction ID** specified in section 2.2.1. If the IPv4 address was 0x11223344 and the most significant 32 bits of the **Transaction ID** (specified in section 2.2.1) was 0xaabbccdd, the resulting X-Address is "0x11223344 ^ 0xaabbccdd = 0xbb99ff99".

If the address family is IPv6 (**Family** is set to "0x02"), this is the client's network byte ordered 128-bit (16 byte) IPv6 address. This value is created from the exclusive-or of the IP address with the 128 bits of the **Transaction Id** specified in section 2.2.1. If the IPv6 address was 0x20010DB8112233445566778899AABBCC and the 128bit Transaction ID (specified in section 2.2.1) was 0x112233445566778899AABBCCDDEEFF00, the resulting X-Address is "0x20010DB8112233445566778899AABBCC ^ 0x112233445566778899AABBCCDDEEFF00 = 0x31233EFC444444CCCCCCCC44444444CC ".

2.2.2.17 MS-Version Attribute

The **MS-Version** attribute is used to convey the **TURN** protocol version. This attribute SHOULD be included in the **Allocate request message** from the protocol client. This attribute SHOULD be included in the **Allocate response message** from the **TURN server**<13>. The format of this attribute is as follows.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Attribute Type																Attribute Length															
Version																															

Attribute Type (2 bytes): The TURN attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x8008".

Attribute Length (2 bytes): The length of bytes of the **Version** field. Set to "0x0004" (4).

Version (4 bytes): This field contains the version of the TURN protocol in use.

The following versions are currently defined:

- "0x00000001" – Used by a protocol client implementing the **Interactive Connectivity Establishment (ICE)** protocol described in [MS-ICE].
- "0x00000002" – Used by a protocol client implementing the ICE protocol described in [MS-ICE2].<14>
- "0x00000003" – Used by a protocol client implementing the ICE protocol described in [MS-ICE2] along with support for **HMAC SHA-256** algorithm in the **Message Integrity** attribute. Used by a TURN server implementing support for the HMAC SHA-256 algorithm in the **Message Integrity** attribute.<15>
- "0x00000004" – Used by a protocol client implementing the ICE protocol described in [MS-ICE2] along with support for HMAC SHA-256 algorithm in the Message Integrity attribute and support for **IPv6** addresses. Used by a TURN server implementing support for the HMAC SHA-256 algorithm in the **Message Integrity** attribute along with support for IPv6 addresses.<16>
- "0x00000005" – Used by a protocol client and a TURN server implementing the **Multiplexed TURN** message format for **UDP** connections.
- "0x00000006" – Used by a protocol client and a TURN server implementing the Multiplexed TURN message format for both UDP and **TCP** connections.

2.2.2.18 MS-Alternate Host Name

The **MS-Alternate Host Name** attribute is used when the protocol connection is encapsulated in an HTTPS stream in accordance with section 2.1.3. In those cases, the **MS-Alternate Host Name** attribute MUST be present whenever the **Alternate Server** attribute specified in section 2.2.2.7 is present. The **TURN client** SHOULD use the value specified in the **MS-Alternate Host Name** attribute whenever a secure **HTTPS** connection with **TLS** certificate validation is required.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Attribute_Type																Attribute_Length															
Alternate_Host_Name (variable)																															
...																															
...																															

...

Attribute_Type (2 bytes): The **TURN** attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x8032".

Attribute_Length (2 bytes): The length of bytes of the **Alternate_Host_Name** field. The **Alternate_Host_Name** length MUST NOT exceed 128 bytes.

Alternate_Host_Name (variable): Variable length of data used as the **Alternate_Host_Name** value. The **Alternate_Host_Name** MUST NOT exceed 128 bytes.

2.2.2.19 MS-Sequence Number Attribute

The **MS-Sequence Number** attribute is used to provide sequence information for all authenticated **request messages** sent from the protocol client to the **TURN server**. This can help against replay attacks. The TURN server SHOULD include this attribute in the initial successfully authenticated **Allocate** response it sends to the protocol client. The **Connection ID** and initial **Sequence Number** are generated by the TURN server. The TURN server MUST use 20 bytes of random data for the **Connection ID**. The **Connection ID** SHOULD be unique per connection on the TURN server. The initial sequence number SHOULD be zero. If the TURN server includes this attribute in the **Allocate** response, the protocol client MUST include this attribute in all subsequent authenticated request messages. The protocol client MUST echo the **Connection ID** that it received from the TURN server in each request message. The protocol client MUST increment the sequence number monotonically for each request message it sends.

If the TURN server supports this attribute, it SHOULD use an algorithm that is tolerant of out-of-order packet reception and dropped packets.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>20</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>30</td><td>1</td></tr>
<tr><td colspan="16">Attribute Type</td><td colspan="16">Attribute Length</td></tr>
<tr><td colspan="32">Connection ID (20 bytes)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">Sequence Number</td></tr>
</table>

Attribute Type (2 bytes): The **TURN** attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x8050".

Attribute Length (2 bytes): The length of bytes of following fields. Set to "0x0018" (24).

Connection ID (20 bytes): A 20-byte connection identifier generated by the TURN server.

Sequence Number (4 bytes): A 32-bit sequence number that is monotonically incremented by the protocol client for each request message it sends to the TURN server.

2.2.2.20 MS-Service Quality Attribute

The **MS-Service Quality** attribute is used to convey information about the data stream that the protocol client is intending to transfer over an allocated port. The protocol client SHOULD<17> include this attribute as part of an **Allocate** **request message**. A **TURN server** SHOULD use the information in this attribute to make decisions about resource allocation, bandwidth prioritization, and data delivery methods. If the attribute is not present in the **Allocate** request message, the TURN server SHOULD assume that the data stream is audio with best effort delivery. The format of this attribute is as follows.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Attribute Type																Attribute Length															
Stream Type																Service Quality															

Attribute Type (2 bytes): The **TURN** attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x8055".

Attribute Length (2 bytes): The length of bytes of the following fields. Set to "0x0004" (4).

Stream Type (2 bytes): The type of data to be transferred over the allocated port.

The following stream types are supported in this extension. All other stream types are reserved for future use.

- "0x0001": Audio
- "0x0002": Video
- "0x0003": Supplemental Video
- "0x0004": Data

Service Quality (2 bytes): The service quality level required by the protocol client for the stream.

The following service quality levels are supported in this extension. All other service quality levels are reserved for future use.

- "0x0000": Best effort delivery.
- "0x0001": Reliable delivery.

2.2.2.21 MS-Alternate Mapped Address

This section follows the product behavior as described in product behavior note <18>.

The **MS-Alternate Mapped Address** attribute is identical to the **Mapped Address** attribute specified in section 2.2.2.1. This attribute is used to identify the public **IPv6 transport address** allocated by the **TURN server** if it is configured to support both **IPv4** and IPv6 and the protocol client requested allocation of both an IPv4 and IPv6 address.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Attribute Type																Attribute Length															

Reserved	Family	Port
IP Address		
...		

Attribute Type (2 bytes): The TURN attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x8090".

Attribute Length (2 bytes): Set to "0x0014" (20) for an IPv6 address.

Reserved (1 byte): The first 8 bits are used for alignment purposes and are ignored.

Family (1 byte): The address family of the Address. It MUST have the value of "0x02" for an IPv6 address.

Port (2 bytes): A network byte ordered representation of the mapped port.

IP Address (16 bytes): The network byte ordered 128-bit (16 bytes) IPv6 mapped address.

2.2.2.22 Multiplexed TURN Session ID

This attribute is populated by the **TURN server** and is included as a part of the Allocate Response TURN message. The **TURN client** MUST use the value provided when forming Multiplexed TURN messages (section 2.2.3).<19>

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Attribute_Type																Attribute_Length															
Session_ID																															
...																															

Attribute_Type (2 bytes): The **TURN** attributes are specified in [IETFDRAFT-STUN-02] section 10.2 and [IETFDRAFT-TURN-08] section 9.2. Set to "0x8095".

Attribute_Length (2 bytes): Set to "0x0008" (8).

Session_ID (8 bytes): The unique identifier for the given allocation and stream on the TURN server.

2.2.3 Multiplexed TURN

Multiplexed TURN messages are used to multiplex data streams from different clients over the same **UDP** or **TCP** port on a **TURN server**.<20>

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Channel_ID																Length															
MTURN_Session_ID																															

...
Data (variable)
...
...

Channel_ID (2 bytes): Always "0xFF10" for Multiplexed TURN packets.

Length (2 bytes): Number of bytes in the frame immediately following the **Length** field.

MTURN_Session_ID (8 bytes): Unique identifier for the given stream on the given TURN server.

Data (variable): Underlying data wrapped in this packet.

3 Protocol Details

3.1 Common Details

3.1.1 Abstract Data Model

None.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

None.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.1.8 Forming Outbound TURN Messages

A **TURN** message MUST begin with the transport specific header, as specified in section 2.1. The TURN message header MUST immediately follow the transport specific header, as specified in section 2.2. The **Magic Cookie** attribute, encoded as specified in section 2.2.2.8, MUST be the first attribute after the TURN header.

3.1.9 Forming Raw Data

All data sent between the protocol client and the **TURN server** that is not encapsulated in either a **Send** request or a **Data Indication** MUST begin with the transport specific header as specified in section 2.1.

3.1.10 Verifying Inbound TURN Messages

A **TURN** message received by either the protocol client or **TURN server** MUST begin with a properly formed transport specific header, as specific in section 2.1. The TURN message header MUST immediately follow the transport specific header, as specified in section 2.2.2. The **Magic Cookie** attribute, encoded as specified in section 2.2.2.8, MUST be the first attribute after the TURN message header. If any of these conditions are not met, the message is considered an improperly formed message and MUST be ignored. If the message transport is **TCP**, the connection SHOULD be disconnected.

3.1.11 Message Authentication

An authenticated **TURN** message MUST include a **Message Integrity** attribute as the last attribute of the message. This attribute and the algorithm used to authenticate the message are specified in section 2.2.2.3.

3.1.12 Digest Challenge Extension

This protocol does not use the **Shared Secret authentication** mechanism specified in [IETFDRAFT-TURN-08] sections 7.1 and 8.2. Instead, it uses long-term credentials that consist of a user name and password that are pre-configured on the protocol client. The **TURN server** MUST be able to verify the user name and discover the associated password. These credentials are used in place of the short-term shared secrets specified in [IETFDRAFT-TURN-08] section 7.2.2. The **Allocate** request and **Allocate error response messages** have been extended to use **long-term credentials** in a **digest** challenge and response exchange. These messages are used in the following procedure:

1. The protocol client MUST form an initial **Allocate request message**, as specified in section 3.2.4.1 and send it to the TURN server.
2. Upon reception of an **Allocate** request message, the TURN server does processing as specified in section 3.3.5.1 sending an Allocate error response message to the protocol client.
3. When the protocol client receives the **Allocate** error response message, it does processing as specified in section 3.2.5.2 sending a second **Allocate** request message to the TURN server.
4. Upon reception of the second **Allocate** request message, the TURN server does processing as specified in section 3.3.5.1 sending either an **Allocate** response message or an **Allocate** error response message to the protocol client.
5. If the protocol client receives an **Allocate** response message, it does processing as specified in section 3.2.5.1. If the protocol client receives an **Allocate** error response message it does processing as specified in section 3.2.5.2.

3.2 Client Details

3.2.1 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.

This protocol uses the abstract data model specified in [IETFDRAFT-TURN-08] section 8.

3.2.2 Timers

Retransmission Timer: This timer SHOULD be used by the protocol client for retransmission of the **Allocate** request and **Set Active Destination request messages** when the protocol client fails to receive a response from the **TURN server**. The protocol client SHOULD start this timer when the request message has been sent to the wire. The protocol client SHOULD retransmit these request messages at a fixed interval of 650 milliseconds. The protocol client SHOULD retransmit a maximum of nine times before assuming the transaction and **TURN** session are no longer valid.

3.2.3 Initialization

The protocol client MUST know the **transport address** of the **TURN server** and a peer with which it wants to communicate. The protocol client also MUST have **long-term credentials** that it can use to authenticate with the TURN server.

3.2.4 Higher-Layer Triggered Events

3.2.4.1 Allocating Public Transport Addresses

When a protocol client is ready to allocate public **transport address**es, it MUST follow the procedure as specified in this section. This procedure replaces [IETFDRAFT-TURN-08] section 8.2 and supplements section 8.3 and is explained in detail in section 3.1.12.

The protocol client MUST send an initial **Allocate** **request message** to the **TURN server**.

- The request MUST be formed as specified in section 3.1.8.
- The request SHOULD include the **MS-Version** attribute, as specified in section 2.2.2.17.
  - If the **MS-Version** attribute is included, the value has to be greater than "0x0004" (4) if Multiplexed TURN allocation is desired:<21>
    - A value of "0x0005" (5) or greater has to be used if Multiplexed TURN allocation for **UDP** protocol is desired.
    - A value of "0x0006" (6) or greater has to be used if Multiplexed TURN allocation for both UDP and **TCP** protocols is desired.
- The request SHOULD include the **MS-Service Quality** attribute, as specified in section 2.2.2.20.<22>
- The request MUST NOT include a **Message Integrity** attribute.

3.2.4.2 Sending TURN Encapsulated Data to the Peer

This section follows the product behavior as described in product behavior note<23>.

When the protocol client needs to send encapsulated data to a peer and set permissions with the **TURN server** to allow data from that peer to be relayed to the protocol client, it MUST follow the procedure specified in [IETFDRAFT-TURN-08] section 8.6 with the following exceptions:

- The **Send** **request message** MUST be formed as specified in section 3.1.8.
- The request SHOULD include the **MS-Version** attribute, as specified in section 2.2.2.17.
- The request MUST include the **MS-Sequence Number** attribute, as specified in section 2.2.2.19.
- The request MUST be authenticated using the procedure specified in section 3.1.11.

3.2.4.3 Set the Peer as the Active Destination

This section follows the product behavior as described in product behavior note<24>.

When the protocol client selects a peer that it wants to use as the destination for all non-TURN encapsulated data, it MUST follow the procedures in [IETFDRAFT-TURN-08] section 8.8, with the following exceptions:

- The **Set Active Destination** **request message** MUST be formed as specified in section 3.1.8.

- The request SHOULD include the **MS-Version** attribute, as specified in section 2.2.2.17.
- The request MUST include the **MS-Sequence Number** attribute, as specified in section 2.2.2.19.
- The request MUST be authenticated using the procedure specified in section 3.1.11.
- The protocol client SHOULD NOT implement the state computer from [IETFDRAFT-TURN-08] section 8.8 controlling the transition from one active peer to another. This mechanism has been removed from more recent versions of the draft.

3.2.4.4 Tearing Down an Allocation

When the protocol client is done with the allocated address, it MUST follow the procedure specified in [IETFDRAFT-TURN-08] section 8.9, with the following exceptions:

- The **Allocate request message** MUST be formed as specified in section 3.1.8.
- The request SHOULD include the **MS-Version** attribute, as specified in section 2.2.2.17.
- The request MUST include the **MS-Sequence Number** attribute, as specified in section 2.2.2.19.
- The request SHOULD<25> include the **MS-Service Quality** attribute, as specified in section 2.2.2.20.
- The request MUST be authenticated using the procedure specified in section 3.1.11.

3.2.4.5 Sending Non-TURN Data to the Peer

When the protocol client is sending data to a peer that has been set as the active destination with the **TURN server**, it MUST follow the procedure specified in [IETFDRAFT-TURN-08] section 8.10, with the exception that the data MUST be formed as specified in section 3.1.9.

3.2.4.6 Sending Multiplexed TURN Encapsulated Data to the Peer

When the **protocol client** needs to send **Multiplexed TURN** encapsulated data to a peer that has been set as the active destination with the **TURN server**, it MUST follow the procedure specified in section 3.2.4.5 with the following exceptions:<26>

- The data send MUST be wrapped in a packet, as specified in section 2.2.3.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Receiving Allocate Response Messages

This section follows the product behavior as described in product behavior note<27> .

When a protocol client receives an **Allocate response message**, it MUST follow the procedure specified in [IETFDRAFT-TURN-08] section 8.4, with the following exceptions:

- The response MUST be verified as specified in section 3.1.10.
- The response MUST be authenticated as specified in section 3.1.11.
- The response MUST include the **XOR Mapped Address** attribute, as specified in section 2.2.2.16.
- The response SHOULD include the **MS-Sequence Number** attribute, as specified in section 2.2.2.19.

- The response SHOULD include the **MS-Version** attribute, as specified in section 2.2.2.17.<28>

The public transport addresses allocated by the **TURN server** depend on the values of the **MS-Version** attribute, specified in section 2.2.2.17, and the **Requested Address Family** attribute, specified in section 2.2.2.15, in the associated **Allocate request message**.

- If the associated **Allocate** request message did not include the **MS-Version** attribute, or if it included the **MS-Version** attribute with a value equal to or less than "0x03" (3), the response SHOULD include a **Mapped Address** attribute, as specified in section 2.2.2.1. This attribute identifies the **IPv4** public transport address allocated by the TURN server.
- If the associated **Allocate** request message included the **MS-Version** attribute with a value equal to or greater than "0x04" (4):<29>
  - If the associated **Allocate** request message included the **Requested Address Family** attribute with the **Family** value set to "0x01" (1) the response MUST include a **Mapped Address** attribute, as specified in section 2.2.2.1. This attribute identifies the IPv4 public **transport address** allocated by the TURN server.
  - If the associated **Allocate** request message included the **Requested Address Family** attribute with the **Family** value set to "0x02" (2) the response MUST include a **Mapped Address** attribute, as specified in section 2.2.2.1. This attribute identifies the **IPv6** public transport address allocated by the TURN server.
  - If the associated **Allocate** request message did not include the **Requested Address Family** attribute:
    - If the TURN server was configured to support allocation of IPv4 addresses the response MUST include a **Mapped Address** attribute, as specified in section 2.2.2.1. This attribute identifies the IPv4 public transport address allocated by the TURN server.
    - If the TURN server was configured to support allocation of IPv6 addresses the response MUST include the **MS-Alternate Mapped Address** attribute, as specified in section 2.2.2.21. This attribute identifies the IPv6 public transport address allocated by the TURN server.
  - If the **MS-Version** attribute value was equal to or greater than "0x05" (5) and the **TURN client** is connected to the TURN server over **UDP**, the response MUST include a **Multiplexed TURN Session ID** attribute as specified in section 2.2.2.22.
  - If the **MS-Version** attribute value was equal to or greater than "0x06" (6) and the TURN client is connected to the TURN server over **TCP**, the response MUST include a **Multiplexed TURN Session ID** attribute as specified in section 2.2.2.22.

The protocol client can advertise the public transport addresses contained in the **Mapped Address** and **MS-Alternate Mapped Address** attributes as destination addresses to receive data over. The protocol client can use the transport address contained in the **XOR Mapped Address** to identify its **public address** as seen by the TURN server.

3.2.5.2 Receiving Allocate Error Response Messages

This section follows the product behavior as described in product behavior note<30>.

When a protocol client receives an **Allocate** error response, it MUST follow the procedure specified in [IETFDRAFT-TURN-08] section 8.4, with the exception that the response MUST be verified as specified in section 3.1.10.

If the error response code is 401, 431, 432, 434, 435, or 438, the protocol client SHOULD retry the **Allocate** request as follows:

- The request MUST be formed as specified in section 3.1.8.
- The request MUST include the **Username** attribute, as specified in section 2.2.2.2.
- The request MUST include the **Realm** attribute, as specified in section 2.2.2.14.
- The request MUST include the **Nonce** attribute, as specified in section 2.2.2.13. The **Nonce** value MUST be equal to what the **TURN server** sent in the previous 401 **error response message**.
- The request SHOULD<31> include the **MS-Service Quality** attribute, as specified in section 2.2.2.20.
- The request SHOULD include the **MS-Version** attribute, as specified in section 2.2.2.17.
- If the **Allocate** error response message from the TURN server included the **MS-Version** attribute<32>, as specified in section 2.2.2.17, and the version value was equal to or greater than "0x4" (4), the protocol client can request that the TURN server allocate an **IPv4**, **IPv6** or both IPv4 and IPv6 public **transport address**es.
  - If the protocol client is requesting allocation of an IPv4 public transport address it MUST include the **Requested Address Family** attribute, as specified in section 2.2.2.15, with a **Family** value of "0x01" (1).
  - If the protocol client is requesting allocation of an IPv6 public transport address it MUST include the **Requested Address Family** attribute, as specified in section 2.2.2.15, with a **Family** value of "0x02" (2).
  - If the protocol client is requesting allocation of both an IPv4 and an IPv6 public transport address it MUST NOT include the **Requested Address Family** attribute, as specified in section 2.2.2.15.
- The request MUST be authenticated as specified in section 3.1.11.

Processing for other error response codes MUST be done as specified in [IETFDRAFT-TURN-08] section 8.4.

3.2.5.3 Receiving Set Active Destination Response Messages

When a protocol client receives a **Set Active Destination** **response message**, it MUST follow the procedure specified in [IETFDRAFT-TURN-08] section 8.8, with the following exceptions:

- The response MUST be verified as specified in section 3.1.10.
- The response MUST be authenticated as specified in section 3.1.11.
- The protocol client SHOULD NOT implement the state computer from [IETFDRAFT-TURN-08] section 8.8 controlling the transition from one active peer to another. This mechanism has been removed from more recent versions of the draft. When the protocol client receives the **Set Active Destination** response message, it SHOULD assume that the **TURN server** has set the active destination.

3.2.5.4 Receiving Set Active Destination Error Response Messages

When a protocol client receives a **Set Active Destination** **error response message**, it MUST follow the procedure specified in [IETFDRAFT-TURN-08] section 8.8, with the following exceptions:

- The response MUST be verified as specified in section 3.1.10.
- The response MUST be authenticated as specified in section 3.1.11.

- The protocol client SHOULD NOT implement the state computer from [IETFDRAFT-TURN-08] section 8.8 controlling the transition from one active peer to another. This mechanism has been removed from more recent versions of the draft. When the protocol client receives the **Set Active Destination** error response message, it SHOULD assume that the **TURN server** has not set an active destination.

3.2.5.5 Receiving Data Indication Messages

This section follows the product behavior as described in product behavior note<33>

When a protocol client receives a **Data Indication** message, it MUST follow the procedure specified in [IETFDRAFT-TURN-08] section 8.7, with the exception that the indication MUST be verified as specified in section 3.1.10.

3.2.5.6 Receiving Non-TURN Data from the Server

Once the protocol client has set a peer as the active destination, it can receive non-TURN framed data from the **TURN server**. This data originates from the active peer and is relayed through the TURN server to the protocol client. When the protocol client receives this data, it MUST follow the procedure specified in [IETFDRAFT-TURN-08] section 8.10, with the exception that the data MUST be formed as specified in section 3.1.9.

3.2.6 Timer Events

Retransmission Timer Expiration: Upon expiration of the retransmission timer, the protocol client SHOULD retransmit the outstanding **request message** for which the timer was originally set. The protocol client SHOULD restart the timer when the retransmitted message has been sent to the wire. The protocol client SHOULD track the number of retransmit attempts it makes, and stop retransmitting after nine attempts. If the protocol client does not receive a response after nine attempts, it SHOULD consider the transaction to have failed.

3.2.7 Other Local Events

None.

3.3 Server Details

3.3.1 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.

This protocol uses the abstract data model specified in [IETFDRAFT-TURN-08] section 7.

3.3.2 Timers

Lifetime Timer: The lifetime timer MUST be implemented as specified in [IETFDRAFT-TURN-08] section 7.7.

3.3.3 Initialization

The **TURN server** MUST be initialized to receive **request messages** over **TCP** or **UDP**. It MUST be ready to receive messages on the default UDP TURN port 3478. It SHOULD be listening on TCP port 443.

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

3.3.5.1 Receiving Allocate Request Messages

This section follows the product behavior as described in product behavior note<34>.

Upon receipt of an **Allocate** **request message**, the **TURN server** does processing as specified in [IETFDRAFT-TURN-08] section 7.2, with the following exceptions:

- The TURN server MUST do basic message verification as specified in section 3.1.10.
- If the request does not include a **Message Integrity** attribute, the TURN server MUST respond with an **Allocate** **error response message** with an error response value of **401 Unauthorized**. The message MUST be formed as follows:
  - The response MUST be formed as specified in section 3.1.8.
  - The response MUST include an **Error Code** attribute with the appropriate error response code.
  - The response MUST include a **Realm** attribute, as specified in section 2.2.2.14.
  - The response MUST include a **Nonce** attribute, as specified in section 2.2.2.13.
  - The response SHOULD include the **Alternate Server** attribute, as specified in section 2.2.2.7.
  - The response SHOULD include the **MS-Version** attribute, as specified in section 2.2.2.17.<35>
  - The response MUST NOT include the **Message Integrity** attribute.
- If the request does include a **Message Integrity** attribute, it MUST be processed as follows:
- The request MUST include the **Username** attribute, as specified in section 2.2.2.2.
  - If the request does not include a **Username** attribute, the TURN server MUST respond with an **Allocate** error response, as specified in Step 2, with an error response code of **432 Missing Username**.
  - If the request includes a **Username** attribute, but the value of the attribute was not understood by the TURN server, the TURN server MUST respond with an **Allocate** error response, as specified in Step 2, with an error response code of **436 Unknown User**.
- The request MUST include the **Realm** attribute, as specified in section 2.2.2.14.
  - If the request does not include a **Realm** attribute, the TURN server MUST respond with an **Allocate** error response, as specified in Step 2, with an error response code of **434 Missing Realm**.
- The request MUST include the **Nonce** attribute, as specified in section 2.2.2.13.

  - If the request does not include a **Nonce** attribute, the TURN server MUST respond with an **Allocate** error response, as specified in Step 2, with an error response code of **435 Missing Nonce**.
  - If the request includes a **Nonce** attribute, but the value was not valid, the TURN server MUST respond with an **Allocate** error response, as specified in Step 2, with an error response code of **438 Stale Nonce**.
- The request SHOULD include the **MS-Version** attribute, as specified in section 2.2.2.17.
- If all of the required attributes are present and valid, the TURN server MUST authenticate the **Allocate** request message as specified in section 3.1.11.
- If **authentication** fails, the TURN server MUST respond with an **Allocate** error response, as specified in step 2, with an error response value of **431 Integrity Check Failure**.
- If authentication succeeds, the TURN server MUST attempt to allocate public **transport address**es on behalf of the protocol client. The type of transport addresses allocated by the TURN server depend on the values of the **MS-Version** attribute, specified in section 2.2.2.17, and the **Requested Address Family** attribute, specified in section 2.2.2.15, in the request.
- If the request did not include the **MS-Version** attribute or if it did include the **MS-Version** attribute with a value equal to or less than "0x03" (3) the TURN server MUST allocate an **IPv4** public transport address.
- If the request did include the **MS-Version** attribute with a value equal to or greater than "0x04" (4):
  - If the request included the **Requested Address Family** attribute with the **Family** value set to "0x01" (1), the TURN server MUST allocate an IPv4 public transport address.
  - If the request included the **Requested Address Family** attribute with the **Family** value set to "0x02" (2), the TURN server MUST allocate an **IPv6** public transport address.
  - If the associated **Allocate** request message did not include the **Requested Address Family** attribute:
    - If the TURN server was configured to support allocation of IPv4 addresses the TURN server MUST allocate an IPv4 public transport address.
    - If the TURN server was configured to support allocation of IPv6 addresses the TURN server MUST allocate an IPv6 public transport address.
  - If the **MS-Version** attribute value was equal to or greater than "0x05" (5) and the **TURN client** is connected to the TURN server over **UDP**, the response MUST allocate a unique **Multiplexed TURN Session ID** attribute as specified in section 2.2.2.22. In this case the **allocated transport address** SHOULD be a single port used by the TURN server to multiplex traffic for all allocated TURN clients.
  - If the **MS-Version** attribute value was equal to or greater than "0x06" (6) and the TURN client is connected to the TURN server over **TCP**, the response MUST allocate a unique **Multiplexed TURN Session ID** attribute as specified in section 2.2.2.22. In this case the allocated transport address SHOULD be a single port used by the TURN server to multiplex traffic for all allocated TURN clients.
- If allocation of a transport address fails for any reason, the TURN server MUST respond with an **Allocate** error response, as specified in step 2, with an error response code of either **300 Try Alternate** or **500 Server Error**. The TURN server SHOULD use an error response code of **Alternate Server** if it is configured in a way that it knows about other servers (2) in the

deployment that implement this protocol. Otherwise, the TURN server MUST use an error response code of **Server Error**.

- If the allocation of the public transport address is successful, the TURN server MUST respond with an **Allocate** response.
- The response MUST be formed as specified in section 3.1.8.
- The response SHOULD include the MS-Version attribute, as specified in section 2.2.2.17.
- If the allocation request was for either an IPv4 or an IPv6 address:
  - The response MUST include the **Mapped Address** attribute, as specified in section 2.2.2.1.The value of the attribute MUST be that of either the IPv4 or IPv6 transport address allocated by the TURN server.
- If the allocation request was for both an IPv4 and an IPv6 address:
  - The response MUST include the **Mapped Address** attribute, as specified in section 2.2.2.1.The value of the attribute MUST be that of the IPv4 transport address allocated by the TURN server.
  - The response MUST include the **MS-Alternate Mapped Address** attribute, as specified in section 2.2.2.21.The value of the attribute MUST be that of the IPv6 transport address allocated by the TURN server.
- The response MUST include the **XOR Mapped Address** attribute, as specified in section 2.2.2.16.
- The response SHOULD include the **MS-Sequence Number** attribute, as specified in section 2.2.2.19.
- The response MUST be authenticated as specified in section 3.1.11.

3.3.5.2 Receiving Send Request Messages

This section follows the product behavior as described in product behavior note<36>

Processing of a **Send** **request message** is done as specified in [IETFDRAFT-TURN-08] section 7.3 with the following exceptions:

- The request MUST be verified as specified in section 3.1.10. If the request fails verification, it MUST be silently dropped by the **TURN server**.
- The request MUST be authenticated as specified in section 3.1.11. If the request fails **authentication**, it MUST be silently dropped by the TURN server.
- The TURN server MUST NOT respond to a protocol client with either a **Send** response or a **Send** error response.

3.3.5.3 Receiving Set Active Destination Request Messages

This section follows the product behavior as described in product behavior note<37>.

Processing of a **Set Active Destination** **request message** is done as specified in [IETFDRAFT-TURN-08] section 7.5, with the following exceptions:

- The request MUST be verified as specified in section 3.1.10.
- The request MUST be authenticated as specified in section 3.1.11.

Any **response message** sent to the protocol client after processing the request is formed as specified in [IETFDRAFT-TURN-08] section 7.5, with the following exceptions:

- The response MUST be formed as specified in section 3.1.8.
- The response MUST be authenticated as specified in section 3.1.11.

The **TURN server** SHOULD NOT implement the state computer from [IETFDRAFT-TURN-08] section 7.5 controlling the transition from one active peer to another. This mechanism has been removed from more recent versions of the draft. If the TURN server successfully processed the request, it SHOULD set the active destination before it sends the **Set Active Destination** response message. If an error occurred while the TURN server was processing the request, it SHOULD NOT change the current active destination. If this is the first **Set Active Destination** request, the TURN server SHOULD NOT set an active destination. If the active destination has been set through an earlier **Set Active Destination** request, the TURN server SHOULD NOT change the active destination.

3.3.5.4 Receiving Data and Connections on the Allocated Transport Address

Processing of incoming data or connection requests on the **allocated transport address** is done as specified in [IETFDRAFT-TURN-08] section 7.4, with the following exceptions:

- If the received data results in a **Data Indication** message sent to the protocol client, the **Data Indication** message MUST be formed as specified in section 3.1.8.
- If the received data is from a peer that has been identified as the active peer through a **Set Active Destination** request, it MUST be formed as specified in section 3.1.9.

3.3.5.5 Receiving Non-TURN Data from the Client

Once the protocol client has set a peer as the active destination, it can send non-TURN framed data to the **TURN server**. This data is relayed through the TURN server to the active peer. When the TURN server receives this data, it MUST follow the procedure specified in [IETFDRAFT-TURN-08] section 7.6, with the exception that the data MUST be formed as specified in section 3.1.9.

3.3.5.6 Receiving Multiplexed TURN Encapsulated Data from the Client

If a **Multiplexed TURN Session ID** (section 2.2.2.22) has been allocated for the given session, once the **TURN client** has set a peer as the active destination, it can send **Multiplexed TURN** encapsulated data to the **TURN server**. This data is relayed through the TURN server to the active peer. When the TURN server receives this data, it MUST follow the procedure specified in [IETFDRAFT-TURN-08] section 7.6, with the exception that the data MUST be formed as specified in section 3.1.9 and in section 2.2.3.<38>

3.3.6 Timer Events

Lifetime Expiration: When the lifetime timer fires, the **TURN server** processes it as specified in [IETFDRAFT-TURN-08] section 7.7.

3.3.7 Other Local Events

None.

4 Protocol Examples

In the following figure, a **TURN client** is behind a **NAT** and is communicating with a peer using **Session Initiation Protocol (SIP)**, as described in [RFC3261]. The protocol client and peer attempt to establish a media flow between them. Because the protocol client is behind a NAT, it allocates a public **transport address** which it includes in the **Session Description Protocol (SDP)** of the SIP **INVITE** sent to the peer, as described in [RFC4566]. The details of the **SIP message** exchange are not included in the example; only the basic message flow used to communicate the **public address** of the protocol client and peer to each other is included.

The TURN client has a private transport address of 10.0.0.1 that it uses for network connectivity. The NAT on the protocol client's private network has a public transport address of 192.0.2.10. The **TURN server** has a public transport address of 192.0.2.20. The peer is connected directly to the Internet and has a transport address of 192.0.2.30. The following figure shows the flow of **TURN** messages used to allocate a public transport address.

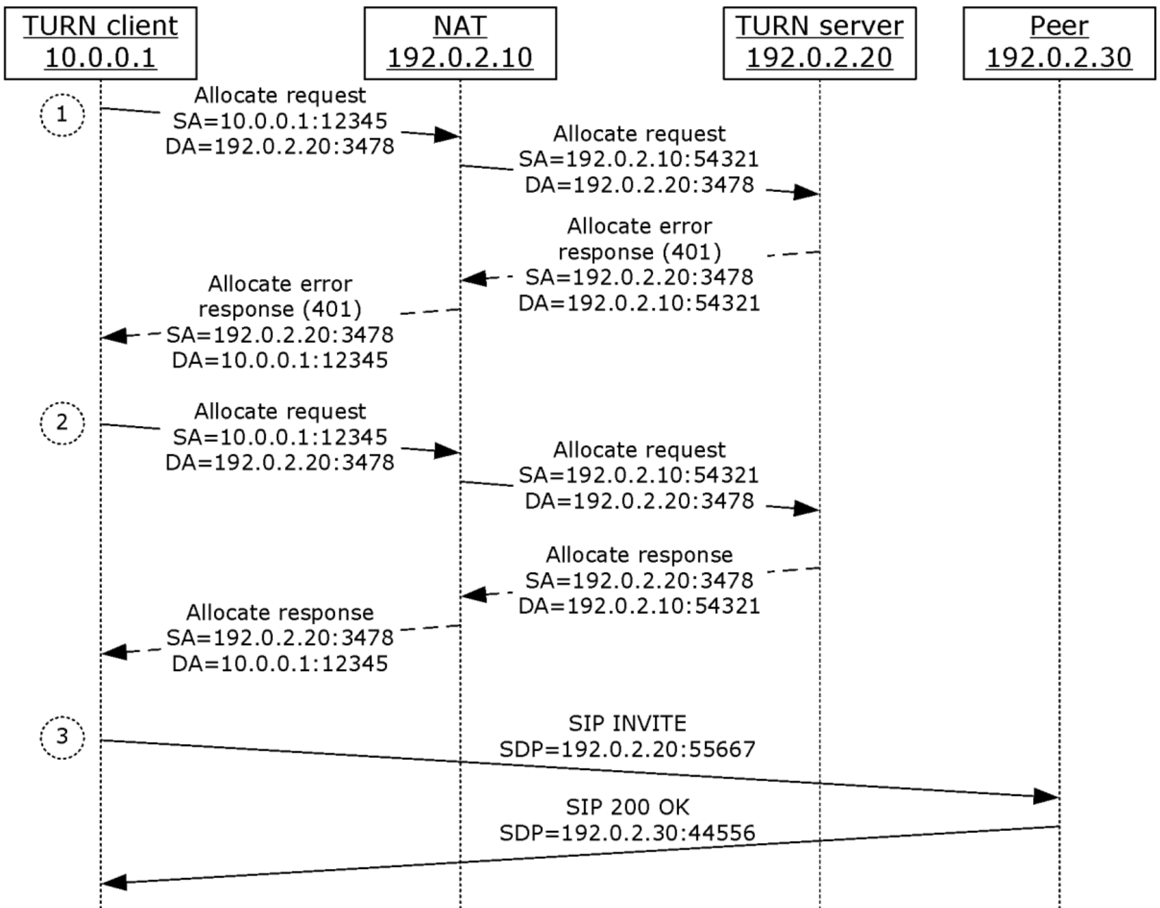

Figure 3: Example of TURN message flow

1. The protocol client sends an initial **Allocate request message** to the TURN server. This request message does not include a **Message Integrity** attribute and begins the **digest authentication** exchange specified in section 3.1.12. The source address for the request is 10.0.0.1:12345 and the destination address is 192.0.2.20:3478. The request passes through the NAT, which allocates a new port, 54321, and creates a binding between the internal address 10.0.0.1:12345 and 192.0.2.10:54321. The NAT translates the source address to 192.0.2.10:54321 and sends the request on to the TURN server. The TURN server checks the request for a **Message Integrity**

attribute. Because the **Message Integrity** attribute is missing, the TURN server challenges the protocol client for credentials by responding with an **Allocate** error response or with an error response code of **401 Unauthorized**. The TURN server sends the **response message** to the protocol client, through the NAT binding, with the NAT translating the destination address.

2. When the protocol client receives the **Allocate error response message**, it retries the **Allocate** request using the **Username**, **Nonce**, and **Realm** attributes specified in section 3.1.12. The request is sent through the NAT binding to the TURN server, with the NAT translating the source address discussed in Step 1. The TURN server validates and authenticates the new **Allocate** request and allocates transport address 192.0.2.20: 55667. It forms an **Allocate** response message and includes the **Mapped Address** attribute with a value of 192.0.2.20:55667 and the **XOR Mapped Address** attribute with a value of 192.0.2.10:54321 **XOR**'d with the **Transaction ID**, as specified in section 2.2.2.16. The response is sent to the protocol client through the NAT binding, with the NAT again doing the required address translation.

3. The protocol client receives the **Allocate** response and uses the **Mapped Address**, 192.0.2.20:55667, in the SDP of the SIP INVITE to signal to the peer the address to send data to. The peer responds to the SIP INVITE with a SIP **200 OK** and includes its address of 192.0.2.30:44556 in the SDP.

   At this point, both the protocol client and the peer have a transport address that they can use to receive data. However, until the protocol client has set permission on the allocated port, the TURN server does not allow any data to be received on the allocated port. The following figure shows the messages used to set permissions on an allocated port and the subsequent data flow.

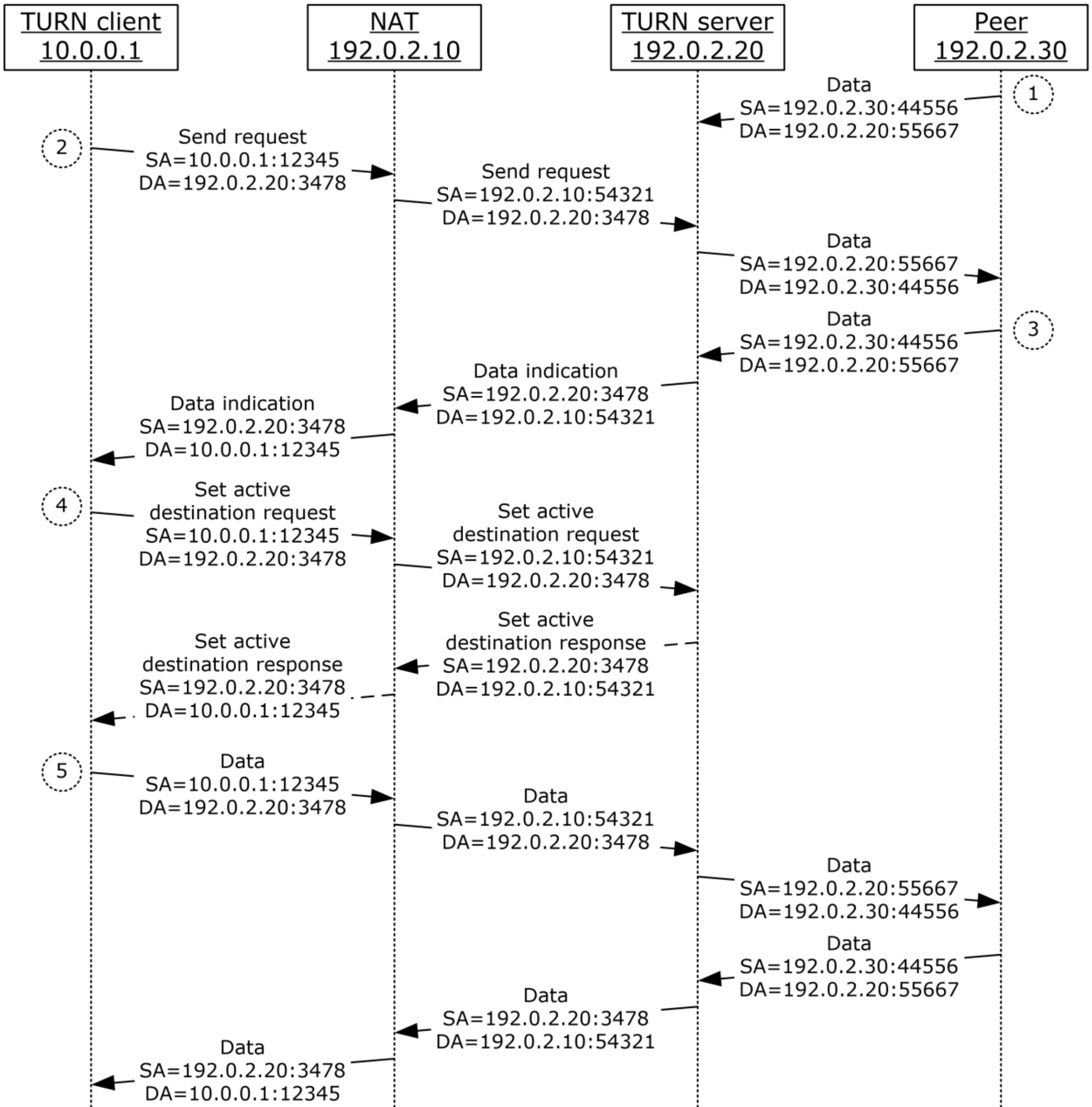

Figure 4: Using TURN messages to set permissions

1. Once the peer has the public transport address of the protocol client, it can start to send data. When the data arrives at the allocated port on the TURN server, the TURN server checks to see if the protocol client has permissions to receive data from the peer, 192.0.2.30:44556. Permissions are set when the protocol client does a **Send** request to the TURN server with the peer's transport address in the **Destination Address** attribute. Because the protocol client has not sent a **Send** request, the TURN server drops the data.

2. Once the protocol client has the public transport address of the peer, it can start to send data. It does this by sending a **Send** request message to the TURN server with the data to be sent in the **Data** attribute and the address of the peer, 192.0.2.30:44556, in the **Destination Address** attribute. The **Send** request is sent to the TURN server through the NAT binding. When the TURN server receives the **Send** request, it adds the peer's **IPv4** address to the permissions list for the

allocated address. It then forwards the data contained in the **Data** attribute on to the peer. The data is sent using the allocated address, 192.0.2.20:55667, as the source address and the address in the **Destination Address** attribute, 192.0.2.30:44556, as the destination address.

3. The peer again attempts to send data to the allocated address. The TURN server checks the permissions list and finds that the peer now has permissions to send data to the protocol client. The TURN server forwards the data to the protocol client using a **Data Indication** message, encapsulating the data in the **Data** attribute and identifying the peer as the source of the data by including a **Remote Address** attribute with the peer's address. The **Data Indication** message is sent to the protocol client through the NAT binding.

4. The protocol client is now ready to make the peer the active destination for all non-TURN encapsulated data. It sends a **Set Active Destination** request message to the TURN server with the peer's address in the **Destination Address** attribute. The request is sent to the TURN server through the NAT binding. When the TURN server receives the request, it identifies the peer as the active destination and sends a **Set Active Destination** response back to the protocol client.

5. Now that the protocol client has established the peer as the active destination, all non-TURN data sent by either the protocol client or the peer is relayed between the two with non-TURN message encapsulation. Only transport specific framing is required. This is a more efficient mechanism for relaying the data.

5 Security

5.1 Security Considerations for Implementers

The security considerations for this protocol are the same as described in [IETFDRAFT-TURN-08] section 10.

The **long-term credentials**, which are used for protocol client **authentication** with the **TURN server**, are valid for an extended period of time. Because the credentials are valid for this extended period, replay prevention is provided through the use of a **digest** challenge as described in section 3.1.12.

The long-term credential mechanism is also susceptible to offline dictionary attacks, so it is recommended that deployments use strong passwords.

5.2 Index of Security Parameters

Security parameter	Section
Use of **long-term credentials** in a **digest** challenge and response exchange.	3.1.12

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 updates to those products.

- 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
- Windows 10 v1511 operating system
- Windows Server 2016 operating system
- Windows Server operating system

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates 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, Lync Server 2010, Lync 2010: **IPv6** addresses are not supported.

<2> Section 2.2.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: This behavior is not supported.

<3> Section 2.2.2: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<4> Section 2.2.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: This behavior is not supported.

<5> Section 2.2.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010, Lync Server 2013, Lync Client 2013/Skype for Business, Skype for Business Server 2015, Skype for Business 2016: This behavior is not supported.

<6> Section 2.2.2.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: IPv6 addresses are not supported.

<7> Section 2.2.2.3: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: **HMAC SHA-256** algorithm is not supported.

<8> Section 2.2.2.7: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: IPv6 addresses are not supported.

<9> Section 2.2.2.10: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: IPv6 addresses are not supported.

<10> Section 2.2.2.11: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: IPv6 addresses are not supported.

<11> Section 2.2.2.15: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: This behavior is not supported.

<12> Section 2.2.2.16: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: IPv6 addresses are not supported.

<13> Section 2.2.2.17: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: This behavior is not supported.

<14> Section 2.2.2.17: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<15> Section 2.2.2.17: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: This behavior is not supported.

<16> Section 2.2.2.17: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: This behavior is not supported.

<17> Section 2.2.2.20: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<18> Section 2.2.2.21: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: This behavior not supported.

<19> Section 2.2.2.22: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010, Lync Server 2013, Lync Client 2013/Skype for Business, Skype for Business Server 2015, Skype for Business 2016: This behavior is not supported.

<20> Section 2.2.3: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010, Lync Server 2013, Lync Client 2013/Skype for Business, Skype for Business Server 2015, Skype for Business 2016: This behavior is not supported.

<21> Section 3.2.4.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010, Lync Server 2013, Lync Client 2013/Skype for Business, Skype for Business Server 2015, Skype for Business 2016: This behavior is not supported.

<22> Section 3.2.4.1: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<23> Section 3.2.4.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: IPv6 addresses are not supported in the **Destination Address** attribute.

<24> Section 3.2.4.3: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: IPv6 addresses are not supported in the **Destination Address** attribute.

<25> Section 3.2.4.4: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<26> Section 3.2.4.6: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010, Lync Server 2013, Lync Client 2013/Skype for Business, Skype for Business Server 2015, Skype for Business 2016: This behavior is not supported.

<27> Section 3.2.5.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: IPv6 addresses, the **Requested Address Family** attribute, and the **MS-Alternate Mapped Address** attribute are not supported.

<28> Section 3.2.5.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: This behavior is not supported.

<29> Section 3.2.5.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010, Lync Server 2013, Lync Client 2013/Skype for Business, Skype for Business Server 2015, Skype for Business 2016: This behavior is not supported.

<30> Section 3.2.5.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: IPv6 addresses and the **Requested Address Family** attribute are not supported.

<31> Section 3.2.5.2: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported. For all other products, this attribute can be included.

<32> Section 3.2.5.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: This behavior is not supported.

<33> Section 3.2.5.5: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: IPv6 addresses are not supported in the **Remote Address** attribute.

<34> Section 3.3.5.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: IPv6 addresses, the **Requested Address Family** attribute and the **MS-Alternate Mapped Address** attribute are not supported.

<35> Section 3.3.5.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: This behavior is not supported.

<36> Section 3.3.5.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: IPv6 addresses are not supported in the **Destination Address** attribute.

<37> Section 3.3.5.3: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: IPv6 addresses are not supported in the **Destination Address** attribute.

<38> Section 3.3.5.6: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010, Lync Server 2013, Lync Client 2013/Skype for Business, Skype for Business Server 2015, Skype for Business 2016: This behavior is not supported.

7 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

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

- A document revision that incorporates changes to interoperability requirements.
- A document revision that captures changes to protocol functionality.

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

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

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

Section	Description	Revision class
6 Appendix A: Product Behavior	Added product to the applicable products list.	Major

8 Index

A

Abstract data model
  client (section 3.1.1 36, section 3.2.1 37, section 3.3.1 42)
  server (section 3.1.1 36, section 3.2.1 37, section 3.3.1 42)
Applicability 14

C

Capability negotiation 14
Change tracking 56
Client
  abstract data model (section 3.1.1 36, section 3.2.1 37, section 3.3.1 42)
  higher-layer triggered events 36
    allocate a public address 38
    send multiplexed TURN encapsulated data to the peer 39
    send non-TURN data to peer 39
    send TURN data to peer 38
    set peer as destination 38
    tear down an allocation 39
  initialization (section 3.1.3 36, section 3.2.3 38)
  message processing 36
    Allocate error response 40
    Allocate response 39
    data indication 42
    digest challenge extension 37
    forming raw data 36
    inbound TURN message 36
    message authentication 37
    outbound TURN message 36
    receive non-TURN data 42
    set active destination error response 41
    set active destination response 41
  other local events (section 3.1.7 36, section 3.2.7 42)
  sequencing rules 36
    Allocate error response 40
    Allocate response 39
    data indication 42
    receive non-TURN data 42
    set active destination error response 41
    set active destination response 41
  timer events (section 3.1.6 36, section 3.2.6 42)
  timers (section 3.1.2 36, section 3.2.2 37)

D

Data model - abstract
  client (section 3.1.1 36, section 3.2.1 37, section 3.3.1 42)
  server (section 3.1.1 36, section 3.2.1 37, section 3.3.1 42)

E

Examples 47

F

Fields - vendor-extensible 14

G

Glossary 6

H

Higher-layer triggered events
  client 36
    allocate a public address 38
    send multiplexed TURN encapsulated data to the peer 39
    send non-TURN data to peer 39
    send TURN data to peer 38
    set peer as destination 38
    tear down an allocation 39
  server (section 3.1.4 36, section 3.3.4 43)

I

Implementer - security considerations 51
Index of security parameters 51
Informative references 9
Initialization
  client (section 3.1.3 36, section 3.2.3 38)
  server (section 3.1.3 36, section 3.3.3 43)
Introduction 6

M

Message Attribute message 19
  Alternate Server 25
  Bandwidth 26
  Data 28
  Destination Address 26
  Error Code 23
  Lifetime 24
  Magic Cookie 26
  Mapped Address (section 2.2.2.1 21, section 2.2.2.21 33)
  Message Integrity 22
  MS-Sequence Number 32
  MS-Service Quality (section 1.3 10, section 2.2.2.20 33)
  MS-Version 30
  Nonce 28
  Realm 28
  Remote Address 27
  Unknown Attributes 24
  Username 21
  XOR Mapped Address 29
Message Header message 18
Message processing
  client 36
    Allocate error response 40
    Allocate response 39
    data indication 42
    digest challenge extension 37

forming raw data 36
inbound TURN message 36
message authentication 37
outbound TURN message 36
receive non-TURN data 42
set active destination error response 41
set active destination response 41
server 36
Allocate request 43
data and connections 46
digest challenge extension 37
forming raw data 36
inbound TURN message 36
message authentication 37
multiplexed TURN encapsulated data 46
non-TURN data 46
outbound TURN message 36
send request 45
set active destination request 45
Messages
Message Attribute 19
Alternate Server 25
Bandwidth 26
Data 28
Destination Address 26
Error Code 23
Lifetime 24
Magic Cookie 26
Mapped Address (section 2.2.2.1 21, section 2.2.2.21 33)
Message Integrity 22
MS-Sequence Number 32
MS-Service Quality (section 1.3 10, section 2.2.2.20 33)
MS-Version 30
Nonce 28
Realm 28
Remote Address 27
Unknown Attributes 24
Username 21
XOR Mapped Address 29
Message Header 18
Multiplexed TURN 34
transport 15
Pseudo-TLS over TCP (section 2.1.1 15, section 2.1.2 17, section 2.1.3 18)
TCP 18
UDP 18
Multiplexed TURN message 34

N

Normative references 8

O

Other local events
client (section 3.1.7 36, section 3.2.7 42)
server (section 3.1.7 36, section 3.3.7 46)
Overview (synopsis) 10

P

Parameters - security index 51
Preconditions 13
Prerequisites 13
Product behavior 52

R

References 8
informative 9
normative 8
Relationship to other protocols 13

S

Security
implementer considerations 51
parameter index 51
Sequencing rules
client 36
Allocate error response 40
Allocate response 39
data indication 42
receive non-TURN data 42
set active destination error response 41
set active destination response 41
server 36
Allocate request 43
data and connections 46
multiplexed TURN encapsulated data 46
non-TURN data 46
send request 45
set active destination request 45
Server
abstract data model (section 3.1.1 36, section 3.2.1 37, section 3.3.1 42)
higher-layer triggered events (section 3.1.4 36, section 3.3.4 43)
initialization (section 3.1.3 36, section 3.3.3 43)
message processing 36
Allocate request 43
data and connections 46
digest challenge extension 37
forming raw data 36
inbound TURN message 36
message authentication 37
multiplexed TURN encapsulated data 46
non-TURN data 46
outbound TURN message 36
send request 45
set active destination request 45
other local events (section 3.1.7 36, section 3.3.7 46)
sequencing rules 36
Allocate request 43
data and connections 46
multiplexed TURN encapsulated data 46
non-TURN data 46
send request 45
set active destination request 45
timer events (section 3.1.6 36, section 3.3.6 46)
timers (section 3.1.2 36, section 3.3.2 42)
Standards assignments 14

T

Timer events
client (section 3.1.6 36, section 3.2.6 42)

server (section 3.1.6 36, section 3.3.6 46)
Timers
client (section 3.1.2 36, section 3.2.2 37)
server (section 3.1.2 36, section 3.3.2 42)
Tracking changes 56
Transport 15
Pseudo-TLS over TCP (section 2.1.1 15, section 2.1.2 17, section 2.1.3 18)
TCP 18
UDP 18
Triggered events - higher-layer
client 36
allocate a public address 38
send multiplexed TURN encapsulated data to the peer 39
send non-TURN data to peer 39
send TURN data to peer 38
set peer as destination 38
tear down an allocation 39
server (section 3.1.4 36, section 3.3.4 43)

V

Vendor-extensible fields 14
Versioning 14