[MS-ICE]:

Interactive Connectivity Establishment (ICE) Extensions

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.
- **Trademarks**. The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

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.

Preliminary Documentation. This particular Open Specifications document provides documentation for past and current releases and/or for the pre-release version of this technology. This document provides final documentation for past and current releases and preliminary documentation, as applicable and specifically noted in this document, for the pre-release version. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. Because this documentation might change between the pre-release version and the final

version of this technology, there are risks in relying on this preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

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	4.0.1	Editorial	Changed language and formatting in the technical content.
2/11/2013	4.0.1	None	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
7/30/2013	4.0.1	None	No changes to the meaning, language, or formatting of the technical content.
11/18/2013	4.0.1	None	No changes to the meaning, language, or formatting of the technical content.
2/10/2014	4.0.1	None	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	4.1	Minor	Clarified the meaning of the technical content.
7/31/2014	4.1	None	No changes to the meaning, language, or formatting of the technical content.
10/30/2014	4.1	None	No changes to the meaning, language, or formatting of the technical content.
3/30/2015	5.0	Major	Significantly changed the technical content.
9/4/2015	5.0	None	No changes to the meaning, language, or formatting of the technical content.
7/15/2016	5.0	None	No changes to the meaning, language, or formatting of the technical content.
8/24/2016	5.0	None	No changes to the meaning, language, or formatting of the technical content.
9/14/2016	5.0	None	No changes to the meaning, language, or formatting of the technical content.
4/27/2018	6.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

This document specifies the Interactive Connectivity Establishment (ICE) Extensions. This protocol consists of a set of proprietary extensions to the ICE protocol. ICE specifies a protocol for setting up the audio/video **Real-Time Transport Protocol (RTP)** streams in a way that allows the streams to traverse Network Address Translators (NAT).

Signaling protocols, such as **Session Initiation Protocol (SIP)**, are used to set up and negotiate audio/video sessions. As part of setting up and negotiating the session, signaling protocols carry the IP addresses and ports of the call participants that receive RTP streams. For this reason, the exchange of local IP addresses and ports might not be sufficient to establish connectivity. ICE uses protocols such as **Simple Traversal of UDP through NAT (STUN)** and **Traversal Using Relay NAT (TURN)** to establish and verify connectivity between two **endpoints**.

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:

answer: A message that is sent in response to an **offer** that is received from an offerer.

callee: An **endpoint** to which a call is initiated by a **caller**.

caller: An **endpoint** that initiates a call to establish a media session.

candidate: A set of **transport addresses** that form an atomic unit for use with a media session. For example, in the case of Real-Time Transport Protocol (RTP) there are two transport addresses for each candidate, one for RTP and another for the Real-Time Transport Control Protocol (RTCP). A candidate has properties such as type, priority, foundation, and base.

candidate identifier: A random string that uniquely identifies a **candidate**.

candidate pair: A set of candidates that is formed from a **local candidate** and a **remote candidate**.

Check List: An ordered list of **candidate pairs** that determines the order in which connectivity checks are performed for those candidate pairs.

component: A representation of a constituent **transport address** if a **candidate** consists of a set of transport addresses. For example, media streams that are based on the Real-Time Transfer Protocol (RTP) have two components, one for RTP and another for the Real-Time Transfer Control Protocol (RTCP).

component identifier: A simple integer that identifies each component in a **candidate** and increments by one for each component.

connectivity check: A **Simple Traversal of UDP through NAT (STUN)** binding request that is sent to validate connectivity between the local and remote candidates in a **candidate pair**.

default candidate: A **candidate** that is designated for streaming media before connectivity checks can be finished. The candidate that is most likely to stream media to the remote **endpoint** successfully is designated as the default candidate.

default candidate pair: A **candidate pair** that consists of the **caller's default candidate** and the **callee**'s default candidate.

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

final offer: An offer that is sent by a **caller** at the end of connectivity checks and carries the **local candidate** and the **remote candidate** that were selected for media flow.

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

ICE keep-alive message: A message that is sent periodically to keep active the **NAT bindings** at intermediate NATs and allocations on the **TURN server**.

initial offer: An offer that is sent by a **caller** and with the caller's **local candidates** when the caller initiates a media session with a **callee**.

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.

local candidate: A **candidate** whose transport addresses are local transport addresses.

local transport address: A transport address that is obtained by binding to a specific port from an IP address on the host computer. The IP address can be from physical interfaces or from logical interfaces such as Virtual Private Networks (VPNs).

matching transport address pair: A transport address pair that is associated with a binding request or a response that is received at a local transport address.

NAT binding: The string representation of the protocol sequence, NetworkAddress, and optionally the endpoint. Also referred to as "string binding." For more information, see [C706] section "String Bindings."

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

offer: A message that is sent by an offerer.

peer: An additional **endpoint** that is associated with an endpoint in a session. An example of a peer is the **callee** endpoint for a **caller** endpoint.

peer-derived candidate: A **candidate** whose **transport addresses** are new mapping addresses, typically allocated by **NATs**, that are discovered during **connectivity checks**.

peer-derived transport address: A derived transport address that is obtained from a connectivity check that is sent to a peer **endpoint**.

provisional answer: An optional message that carries **local candidates** for a **callee** and can be sent by the callee in response to a **caller's** initial offer.

Real-Time Transport Control Protocol (RTCP): A network transport protocol that enables monitoring of Real-Time Transport Protocol (RTP) data delivery and provides minimal control and identification functionality, as described in [RFC3550].

Real-Time Transport Protocol (RTP): A network transport protocol that provides end-to-end transport functions that are suitable for applications that transmit real-time data, such as audio and video, as described in [RFC3550].

remote candidate: A **candidate** that belongs to a remote **endpoint** in a session.

remote endpoint: See **peer**.

RTCP packet: A control packet consisting of a fixed header part similar to that of RTP packets, followed by structured elements that vary depending upon the RTCP packet type. Typically, multiple RTCP packets are sent together as a compound RTCP packet in a single packet of the underlying protocol; this is enabled by the length field in the fixed header of each RTCP packet. See [RFC3550] section 3.

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].

Simple Traversal of UDP through NAT (STUN): A protocol that enables applications to discover the presence of and types of network address translations (NATs) and firewalls that exist between those applications and the Internet.

STUN candidate: A **candidate** whose transport addresses are STUN-derived transport addresses. See also **Simple Traversal of UDP through NAT (STUN)**.

STUN-derived transport address: A derived transport address that is obtained by an **endpoint** from a configured STUN server. See also **Simple Traversal of UDP through NAT (STUN)**.

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 address pair: The **transport address** of a component of the **local candidate** and the transport address of the same component of the **remote candidate** in a **candidate pair**.

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 candidate: A **candidate** whose transport addresses are TURN-derived transport addresses. See also **Traversal Using Relay NAT (TURN)**.

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.

TURN-derived transport address: A derived transport address that is obtained from a Traversal Using Relay NAT (TURN) server.

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

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-ICENAT-06] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-06, October 2005, http://tools.ietf.org/html/draft-ietf-mmusic-ice-06

[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-TCPCICE-00] Rosenberg, J., "TCP Candidates with Interactive Connectivity Establishment", draft-ietf-mmusic-ice-tcp-00, February 2006, http://tools.ietf.org/html/draft-ietf-mmusic-ice-tcp-00

[MS-TURN] Microsoft Corporation, "Traversal Using Relay NAT (TURN) Extensions".

[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

[RFC4571] Lazzaro, J., "Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport", RFC 4571, July 2006, http://www.ietf.org/rfc/rfc4571.txt

1.2.2 Informative References

[MS-SDPEXT] Microsoft Corporation, "Session Description Protocol (SDP) Version 2.0 Extensions".

[RFC3264] Rosenberg, J., and Schulzrinne, H., "An Offer/Answer Model with the Session Description Protocol (SDP)", RFC 3264, June 2002, http://www.rfc-editor.org/rfc/rfc3264.txt

1.3 Overview

This protocol is used to establish media flow between a **caller endpoint** and a **callee** endpoint. In typical deployments, network address translators (NATs) or firewalls exist between the two endpoints that are intended to communicate. **NATs** and firewalls are deployed to provide private address space and to secure the private networks to which the endpoints belong. This type of deployment blocks incoming traffic. If the endpoint advertises its local interface address, the **remote endpoint** might not be able to reach it.

Advertising the address exposed by the NAT or firewall is not as straightforward because the endpoints need to determine the external routable mapping address created by the NAT, which is called a NAT-mapped address, for its local interface address. Moreover, NATs and firewalls are different in the way they create the NAT-mapped addresses. For more information about NAT types, see [IETFDRAFT-STUN-02] section 5. ICE provides a generic mechanism to assist media in traversing NATs and firewalls without requiring the endpoints to be aware of their network topologies. ICE assists media in traversing NATs and firewalls by gathering one or more **transport addresses**, which the two endpoints can potentially use to communicate, and then determining which transport address is best for both endpoints to use to establish a media session.

The following figure shows a typical deployment scenario with two endpoints that establish a media session.

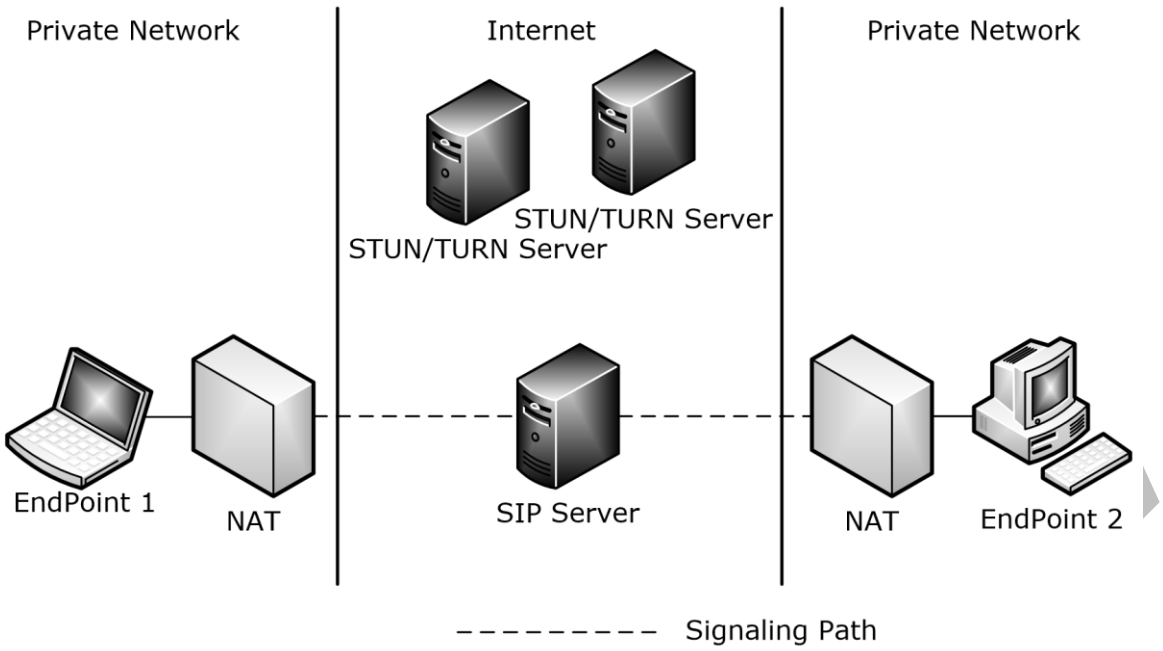

Figure 1: ICE deployment scenario

To facilitate ICE, a communication channel using a signaling protocol, such as **Session Initiation Protocol (SIP)**, through which the endpoints exchange messages is necessary. One example is **Session Description Protocol (SDP)**, as described in [RFC3264]. ICE assumes that such a channel exists and is not intended to be used for NAT traversal for these signaling protocols. ICE is typically deployed in conjunction with **Simple Traversal of UDP through NAT (STUN)** and **Traversal Using Relay NAT (TURN)** servers. The endpoints can share the same STUN and **TURN servers** or use different servers. For more information, see [IETFDRAFT-STUN-02] and [MS-TURN].

The sequence diagram in the following figure outlines the various phases involved in establishing a session between two endpoints using this protocol. These phases are:

1. The **candidates** gathering phase.
2. The exchange of gathered transport addresses between the caller and callee endpoints.
3. The **connectivity checks** phase.
4. The exchange of candidates selected by the connectivity checks phase.

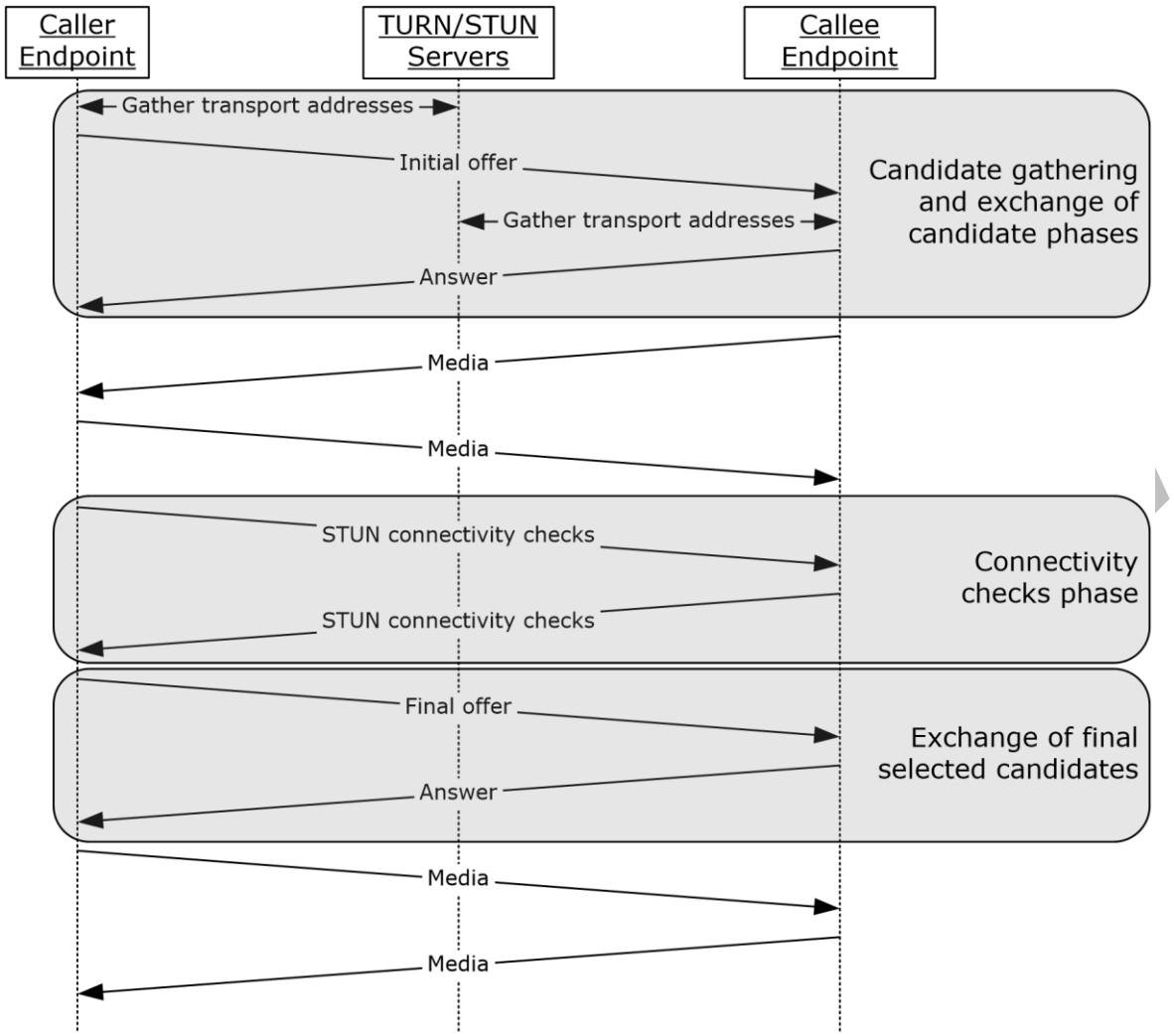

Figure 2: ICE sequence diagram

During the candidates gathering phase, the caller attempts to establish a media session and gathers transport addresses that can potentially be used to communicate with its **peer**. These potential transport addresses include:

- Transport addresses obtained by binding to attached network interfaces. These include both physical interfaces and virtual interfaces such as virtual private network (VPN), which is a "local" transport address.
- Transport addresses that are mappings on the public side of a NAT, which is also called a **STUN-derived transport address**.
- Transport addresses allocated from a TURN server, which are also called **TURN-derived transport addresses**.

The gathered transport addresses are used to form candidates. A candidate is a set of transport addresses that can be potentially used for media flow. For example, in the case of real-time media flow using **Real-Time Transport Protocol (RTP)**, each candidate consists of two transport addresses, one for RTP and another for **Real-Time Transport Control Protocol (RTCP)**. Each gathered candidate is assigned a unique identifier, called the **candidate identifier**, and a priority value based on how it was obtained. This priority indicates the preference of an endpoint to use one candidate over another, if both candidates are reachable from the peer. Typically, candidates obtained

from local network interfaces are given a higher priority than the candidates obtained from TURN servers. The endpoint also designates one of the gathered candidates as the **default candidate**, based on local policy. The gathered candidates are then sent to the peer in the **offer**. The offer is typically encoded into an SDP message and exchanged over a signaling protocol such as SIP.

The callee, after receiving the offer, follows the same procedure and gathers its candidates. The gathered candidates are encoded and sent to the caller in the **answer**. With the exchange of transport addresses complete, both the endpoints are now aware of their peer's transport addresses. The start of the connectivity checks phase is triggered at an endpoint when it is aware of its peer's candidates. Both endpoints pair up the local and **remote candidates** to form a list of **candidate pairs** that are ordered based on the priorities of the candidates. The candidate pair that consists of the default **local candidate** and default remote candidate is designated as the **default candidate pair**. The default candidate pair is moved to the top of the candidate pair **Check List**.

Both endpoints systematically perform connectivity checks starting from the top of the candidate pair list to determine the highest priority candidate pair that can be used by the endpoints for establishing a media session. Connectivity checks involve sending peer-to-peer STUN binding request messages and responses from the **local transport addresses** to the remote transport addresses of each candidate pair in the list. Once a STUN binding request message is received and it generates a successful STUN binding response message for a candidate pair, it is considered valid for sending. Once a successful STUN binding response message is received for a STUN binding request message sent for the candidate pair, it is considered valid for receiving. A connectivity check for a candidate pair is considered to be valid if a candidate pair is considered both valid for sending and valid for receiving. The endpoints can start streaming media from the local default candidate to the remote default candidate after the exchange of candidates is finished, even before the default candidate pair is validated by connectivity checks, but there is no guarantee that the media will reach the peer during this time.

The connectivity checks for the **transport address pairs** are spaced at regular intervals to avoid flooding the network. Depending on the topology, many of the possible candidate pairs might fail connectivity checks. For example, in the topology illustrated in the preceding figure titled "ICE deployment scenario," the transport addresses obtained from the local network interfaces cannot be used directly to establish a connection because both endpoints are behind NATs.

The endpoints can also discover new candidates during the connectivity check phase. This can happen in either of two scenarios:

- The STUN binding request message is received from a new transport address.
- The STUN binding response message was from a request received from a new mapped transport address.

These scenarios arise if new external mappings are created by the NATs residing between the endpoints. Connectivity checks are sent out on candidate pairs formed using these newly created candidates. These candidates can potentially be used for media flow as well. At the end of the connectivity checks phase, the caller sends a **final offer** with only the best local and remote candidate selected during the connectivity checks phase. The peer acknowledges the final offer with an answer and both endpoints start using the selected transport addresses for sending media.

1.4 Relationship to Other Protocols

This protocol is an application layer protocol that depends on, and works with, the **Transmission Control Protocol (TCP)** and **User Datagram Protocol (UDP)** transport protocols for **IPv4** addresses only.

This protocol works with implementations of **Traversal Using Relay NAT (TURN)** protocols, as described in [MS-TURN], to create **TURN candidates** and **STUN candidates**.

This protocol can perform **connectivity checks** only with **endpoints** that follow the message formats in **Simple Traversal of UDP through NAT (STUN)** specifications, as described in [IETFDRAFT-STUN-02], and that follow the STUN attributes and usage specification in section 3.1.4.3.

This protocol depends on signaling protocols, such as **Session Initiation Protocol (SIP)**, to perform an **offer** and **answer** exchange of **Session Description Protocol (SDP)** messages, as described in [MS-SDPEXT].

This protocol is used to establish a communication channel that is used for media flow for protocols such as **Real-Time Transport Protocol (RTP)** and **Real-Time Transport Control Protocol (RTCP)**.

1.5 Prerequisites/Preconditions

This protocol requires the **endpoints** to be able to communicate through a signaling protocol, such as **Session Initiation Protocol (SIP)**, to exchange **candidates**.

1.6 Applicability Statement

This protocol requires **TURN servers** to be deployed to facilitate communication across network address translators (NATs) and firewalls. In the absence of TURN servers, this protocol might not be able establish connectivity between **endpoints**.

This protocol is appropriate for establishing a communication channel between two endpoints for media exchange.

This protocol cannot be used for establishing a communication channel through **Transmission Control Protocol (TCP)** in the absence of a TURN server.

This protocol is used for establishing connectivity for streaming **Real-Time Transport Protocol (RTP)** media. As a result, this protocol supports having exactly two **components** for each **candidate**. It does not support scenarios that require less than two or greater than two components for each candidate.

This protocol does not guarantee consecutive ports for RTP and **Real-Time Transport Control Protocol (RTCP)**. As a result, endpoints that need to communicate with an endpoint that implements this protocol must support sending and receiving media to RTP and RTCP on nonconsecutive ports, whether or not they support ICE itself.

This protocol multiplexes both components to the same IP address and port when the connection is established through TCP. The application layer must be able to demultiplex the data sent for the two components if TCP candidates are used. For example, if the two components are RTP and RTCP, both RTP and RTCP are delivered to the same IP address and port. Both endpoints must multiplex components over TCP.

ICE keep-alive messages are sent only for the RTP component's **transport addresses**. **RTCP packets** are sent to keep the **NAT bindings** and **Traversal Using Relay NAT (TURN)** allocations active for RTCP component's transport addresses. ICE keep-alive messages are sent regardless of whether **User Datagram Protocol (UDP)** or TCP is the underlying transport used.

1.7 Versioning and Capability Negotiation

This protocol is implemented on top of the **Transmission Control Protocol (TCP)** and **User Datagram Protocol (UDP)** transport protocols for **IPv4**, as described in section 2.1.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

Endpoints implementing this protocol MUST NOT send messages that are greater than 1,500 bytes in length. They MUST be able to receive messages 1,500 bytes or less in length. This protocol uses the **Transmission Control Protocol (TCP)** and **User Datagram Protocol (UDP)** transport protocols for **IPv4**. This protocol does not support **IPv6**.

2.2 Message Syntax

This section specifies the various messages used by this protocol implementation. This includes both outgoing and incoming messages. This protocol does not define its own custom message formats. The messages used by this protocol and the protocols they belong to are listed later in this section.

2.2.1 TURN Messages

This protocol SHOULD use a **TURN server** that implements a protocol, as specified in [MS-TURN], to discover **STUN-derived transport addresses** and **TURN-derived transport addresses**. The **endpoint** implementing that protocol to communicate with the TURN server MUST use the message syntax that is specified in [MS-TURN].

2.2.2 STUN Messages

This protocol uses **Simple Traversal of UDP through NAT (STUN)** request and response messages for **connectivity checks** between the two **endpoints**. The STUN messages MUST follow the message formats specified in [IETFDRAFT-STUN-02] section 7 and [IETFDRAFT-STUN-02] section 10. STUN messages sent over **Transmission Control Protocol (TCP)** MUST follow the framing method specified in [RFC4571] section 2. This method is needed to demultiplex the received application data and STUN packets. The **Type** field of **XOR-MAPPED-ADDRESS** attribute MUST have a value of 0x0020.

2.2.3 ICE keep-alive

The **ICE keep-alive message** MUST be a valid **Simple Traversal of UDP through NAT (STUN)** binding request message, as specified in [IETFDRAFT-STUN-02] section 7 and [IETFDRAFT-STUN-02] section 10, and MUST follow the additional specifications in this section. ICE keep-alive messages sent over **Transmission Control Protocol (TCP)** MUST follow the framing method specified in [RFC4571] section 2. The **transaction ID** can be any valid **transaction ID**. The ICE keep-alive message MUST have the **MESSAGE-INTEGRITY** attribute set to a value of 0. It MUST NOT have any other attributes.

3 Protocol Details

3.1 Common Details

The procedures specified apply to both the **Transmission Control Protocol (TCP)** and **User Datagram Protocol (UDP)** transport protocols unless a procedure explicitly specifies a transport protocol.

3.1.1 Abstract Data Model

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

This protocol uses the abstract model specified in [IETFDRAFT-ICENAT-06] section 7 and [IETFDRAFT-TCPCICE-00] section 7.

3.1.2 Timers

The **Candidates Gathering Phase** timer tracks the maximum duration for the **candidates** gathering phase. This timer SHOULD have a default value of 10 seconds.

The **Connectivity Checks Phase** timer tracks the maximum duration for which **connectivity checks** can be performed between the **candidate pairs**. The maximum timeout for this timer MUST be set to 10 seconds.

The **ICE keep-alive** timer tracks the spacing of **ICE keep-alive messages**. These messages are sent to keep the **NAT bindings** and **Traversal Using Relay NAT (TURN)** allocations active. This timer MUST have a default value of 19 seconds or less.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

This section outlines the higher-layer events that trigger the start of the various phases of this protocol for connection establishment. Updating **candidate** lists during and after the **connectivity checks** is allowed, as specified in [IETFDRAFT-ICENAT-06]. This protocol specifies that there MUST NOT be additional **offer** or exchange of candidates other than those specified in this section. Processing is specified for each media stream. If connectivity has to be established for more than one media stream, connectivity establishment MUST be carried out separately for each media stream. If the **transport address** for media or any of the candidates needs to change, the **endpoints** MUST stop the specific media stream and restart it so that the procedure outlined in this section is triggered again. In case the **peer** does not support Interactive Connectivity Establishment (ICE), the default transport addresses used for media MUST NOT be changed after the **initial offer** and **answer**.

3.1.4.1 Sending the Initial Offer

The **caller** attempting to establish a media session with a **peer** MUST gather its **local candidates**, as specified in section 3.1.4.8.1. After the **candidates** are gathered, they MUST be encoded using protocols such as **Session Description Protocol (SDP)** for sending the gathered candidates to the peer **endpoint** through the pre-established signaling channel. It MUST designate one of the local

candidates as the **default candidate** in the **initial offer**. The default candidate MUST be a **User Datagram Protocol (UDP)** candidate. If no UDP candidate is gathered, the call MUST fail.

3.1.4.2 Receiving the Initial Offer and Generating the Answer

The **callee**, on receiving the **initial offer**, MUST gather its **local candidates**, as specified in section 3.1.4.8.1. After the **candidates** are gathered, they MUST be encoded into protocols, such as **Session Description Protocol (SDP)**, for sending the gathered candidates to the **peer** through the pre-established signaling channel. The callee MUST designate one of the local candidates as the **default candidate** in the **answer** to the initial offer. The default candidate MUST be a **User Datagram Protocol (UDP)** candidate. If no UDP candidates are gathered, the call MUST fail.

When the callee completes gathering its local candidates, it MUST start the **connectivity checks** phase as specified in section 3.1.4.8.2. The callee MAY encode the gathered candidates and send them in a **provisional answer** to the **caller** before sending the answer to the initial offer. This is done to reduce the latency of the connectivity establishment as perceived by the user. If an **endpoint** sends a provisional answer, the subsequent answer for the initial offer MUST have the same set of candidates and default candidate that was in the provisional answer.

3.1.4.3 Processing the Provisional Answer to the Initial Offer

The **caller**, after receiving the **provisional answer** with the **callee's candidates**, MUST start the **connectivity checks**, as specified in section 3.1.4.8.2, with the following differences:

- The **Simple Traversal of UDP through NAT (STUN)** binding request messages MUST be sent by the caller for **candidate pairs** whose **local candidates** are **Traversal Using Relay NAT (TURN)** derived.
- The STUN binding request messages sent by the caller for the connectivity checks MUST NOT have the **USERNAME** attribute. These STUN binding request messages are discarded by the **peer endpoint**. They serve only to open permissions on the **TURN servers** for the peer's connectivity checks. Retries to these STUN binding request messages MUST NOT be triggered until the **answer** to the **initial offer** is received.
- STUN binding request messages received from the peer MUST be responded to as specified in section 3.1.5.2.1. In particular, the received STUN binding request messages MUST be cached and they MUST be processed after the initial answer is received from the callee.

3.1.4.4 Processing the Answer to the Initial Offer

The **caller**, on receiving the **answer** to its **initial offer** with the **callee's candidates**, MUST start the **connectivity checks** phase, as specified in section 3.1.4.8.2.

3.1.4.5 Generating the Final Offer

At the end of the **connectivity checks** phase, the **endpoint** that initiated the media session MUST send the **final offer**. The final offer MUST contain only the **local candidate** and **remote candidate** selected by this protocol, encoded into **Session Description Protocol (SDP)** or something similar, to its **peer**. The final offer MUST be generated even if the selected local candidate and remote candidate match the default local candidate and remote candidate of the **initial offer** and **answer**. A media session can have more than one media stream. For example, assume that Endpoint A initiates a media session with an audio stream only with peer endpoint, Endpoint B. Later, Endpoint B adds a video stream to the media session. Endpoint A, the endpoint that initiated the media session, sends the final offer for the video stream also, even though Endpoint B initiated the video stream.

3.1.4.6 Receiving the Final Offer and Generating the Answer

An **endpoint**, on receiving the **final offer**, MUST switch to using the local and **remote candidates** in the **offer** for media flow. It MUST acknowledge the receipt of the final offer with a response that MUST contain only the **local candidate** and remote candidate to be used for media flow. If the selected local candidate is a **TURN candidate**, a **Set Active Destination** message, as specified in [MS-TURN], SHOULD be sent for that **candidate**, and the subsequent processing SHOULD be as specified in [MS-TURN]. Local candidates other than the selected local candidate SHOULD be freed.

3.1.4.7 Processing the Answer to the Final Offer

An **endpoint**, after receiving the **answer** to its **final offer**, MUST switch to using the local and **remote candidates** in the answer for media flow. An endpoint, upon receiving the answer to its final offer, SHOULD free all **local candidates** other than the selected local candidate. If the selected local candidate is a **TURN candidate**, a **Set Active Destination** message, as specified in [MS-TURN], SHOULD be sent for that **candidate**.

3.1.4.8 Common Procedures

3.1.4.8.1 Candidates Gathering Phase

The **candidates** gathering phase is common to both the **caller** and **callee**. Sections 3.1.4.1 and 3.1.4.2 specify when the candidates gathering phase is triggered on caller and callee **endpoints**. This section specifies the operations involved in the candidates gathering phase. The candidates gathering phase MUST end when the **C**andidates **Gathering Phase** timer fires or when the process of gathering candidates process is complete.

Because this protocol is used for streaming **Real-Time Transport Protocol (RTP)** media, each candidate MUST have two **components**. One component is for RTP and the other is for **Real-Time Transport Control Protocol (RTCP)**. This protocol gathers **IPv4** addresses for **Transmission Control Protocol (TCP)** and **User Datagram Protocol (UDP)** transports. Each candidate MUST be associated with a **candidate identifier** and password. Each candidate MUST be assigned a priority value from 0 through 1, with 1 being the highest priority, as specified in [IETFDRAFT-ICENAT-06].

Implementers of this protocol MUST NOT support sending more than 20 candidates in the **offer** or **answer**. If an endpoint gathers more than 20 candidates, it MUST send no more than 20 candidates for the offer exchange and discard the additional candidates. This is done to mitigate the **Simple Traversal of UDP through NAT (STUN)** amplification attack specified in section 5.1.4.

3.1.4.8.1.1 Gathering Candidates

This section specifies the **candidate** types and behavior supported by this protocol. An implementer of this protocol MUST support gathering candidates of the following types:

- **User Datagram Protocol (UDP) local candidates**
- UDP **STUN candidates**
- UDP **TURN candidates**
- **Transmission Control Protocol (TCP)** STUN candidates
- TCP TURN candidates

The implementer of this protocol MUST NOT support the gathering of other candidate types or candidate behaviors. The **Real-Time Transport Protocol (RTP)** and **Real-Time Transport Control Protocol (RTCP) components** of UDP candidates MUST have the same IP address and different

ports. For TCP candidates, both components MUST have the same IP address and port. As a result, for TCP candidates both of the components MUST be multiplexed onto the same IP address and port.

The gathered **transport addresses** MUST NOT be null (0.0.0.0), multicast, or broadcast IP addresses. The addresses MUST NOT be a **fully qualified domain name (FQDN)** as specified in [IETFDRAFT-ICENAT-06] section 7.3. The ports of the gathered transport addresses MUST NOT be in the port range 0–1023.

3.1.4.8.1.2 Gathering UDP Candidates

User Datagram Protocol (UDP) local candidates are obtained by binding to ephemeral ports on all available network interfaces. This includes both physical interfaces and virtual interfaces, such as virtual private network (VPN).

UDP **TURN candidates** SHOULD be obtained following the procedures for allocating **candidates** on the **TURN server**, as specified in [MS-TURN].

UDP **STUN candidates** SHOULD be discovered by following the procedure specified in [MS-TURN].

3.1.4.8.1.3 Gathering TCP Candidates

All gathered **Transmission Control Protocol (TCP) candidates** MUST have the same behavior as candidates that can both actively initiate and passively listen for new connections, otherwise known as *actpass candidates*, as specified in [IETFDRAFT-TCPCICE-00] section 7 for **connectivity checks**, with the following exceptions:

- TCP **TURN candidates** SHOULD be obtained following the procedures for allocating candidates on the **TURN server** specified in [MS-TURN].
- TCP **STUN candidates** SHOULD be discovered by following the procedure specified in [MS-TURN]. TCP STUN candidates MUST NOT listen on the associated **local transport address**. During the connectivity checks phase, outgoing connections for the TCP STUN candidates MUST be initiated from a port on the associated local transport address that is different from the port used to communicate with the **Traversal Using Relay NAT (TURN)** server when gathering the candidate.

3.1.4.8.1.4 Generating the Candidate Identifier, Password, and Component Identifier

The **candidate identifier** MUST be a randomly generated string of 32 characters. The password MUST be a randomly generated string of 16 characters. The **Real-Time Transport Protocol (RTP) component** MUST be assigned a **component identifier** of 1, and the **Real-Time Transport Control Protocol (RTCP)** component MUST be assigned a component identifier of 2. The candidate identifier, component identifiers, and password MUST be exchanged by the **endpoints** during the **offer** and **answer** exchange.

3.1.4.8.2 Connectivity Checks Phase

An application triggers the start of the **connectivity checks** phase after the completion of the **offer** and **answer** exchange of **candidates**, as specified in sections 3.1.4.2, 3.1.4.3, and 3.1.4.4. The connectivity checks phase MUST have an overall worst case timeout, as specified in section 3.1.6.2. When a connectivity check request and a connectivity check response packet have been received from the **peer**, the timeout for the connectivity check MUST be reduced to the value specified in section 3.1.6.2.

3.1.4.8.2.1 Forming the Candidate Pairs

After the **offer** and **answer** exchange of the **candidates** is finished, both **endpoints** have a set of **local candidates** and **remote candidates**. The local candidates and remote candidates are paired together to form **candidate pairs**. Local candidates and remote candidates with the same transport protocol MUST be paired together to form candidate pairs. Local candidates and remote candidates with different transport protocols MUST NOT be paired together to form candidate pairs.

Each candidate pair MUST consist of two **transport address pairs**, one for the **Real-Time Transport Protocol (RTP) component** and another for the **Real-Time Transport Control Protocol (RTCP)** component. For a candidate pair, the component of the local candidate MUST be paired with the corresponding component of the remote candidate to form a transport address pair. For example, the local candidate's RTP component **transport address** is paired with the remote candidate's RTP component transport address. Endpoints implementing this protocol MUST NOT generate more than 40 candidate pairs.

3.1.4.8.2.2 Ordering the Candidate Pairs

The **candidate pairs** MUST be ordered as specified in [IETFDRAFT-ICENAT-06] section 7.5.

3.1.4.8.2.3 Updating the Candidate Pair States

Each **candidate pair** state is updated as the **connectivity checks** progress. The state machine and **User Datagram Protocol (UDP)** candidate pair states are specified in [IETFDRAFT-ICENAT-06] section 7.6. The state machine and **Transmission Control Protocol (TCP)** candidate pair states are specified in [IETFDRAFT-TCPCICE-00] section 7.

3.1.4.8.2.4 Forming and Sending Binding Requests for Connectivity Checks

Connectivity checks are performed between the two **endpoints** by sending peer-to-peer **Simple Traversal of UDP through NAT (STUN)** binding request messages, as specified in [IETFDRAFT-ICENAT-06]. The STUN binding request message MUST have the **USERNAME** and **MESSAGE-INTEGRITY** attributes. Mandating the use of the **MESSAGE-INTEGRITY** attribute in STUN binding request messages serves to mitigate attacks on connectivity, as described in section 5.1.3.

The **USERNAME** of the STUN binding request message MUST be the **transport address pair** identifier of the corresponding transport address pair as seen by its **peer**. That is, the **USERNAME** is the transport address pair identifier that is computed by the peer for the specific transport address pair. The password of the **remote candidate** MUST be used as the password for computing the **MESSAGE-INTEGRITY**. The format of the STUN binding request message and the procedure for calculating the message integrity is specified in [IETFDRAFT-STUN-02] section 8.1.

The connectivity checks are sent between transport address pairs based on the check ordering of **candidate pairs**, as specified in [IETFDRAFT-ICENAT-06] section 7.6. The processing of connectivity checks and their responses are specified in section 3.1.5.

3.1.4.8.2.5 Spacing the Connectivity Checks

To avoid flooding the network, the **connectivity checks** SHOULD be spaced as specified in [IETFDRAFT-ICENAT-06] section 7.6.

The retry of connectivity checks for a **transport address pair** SHOULD be spaced by a constant duration. This spacing MUST be followed for connectivity check packets irrespective of whether the connectivity checks are sent over **User Datagram Protocol (UDP)** or **Transmission Control Protocol (TCP)**.

3.1.4.8.2.6 Terminating the Connectivity Checks

The **connectivity checks** phase MUST be terminated either when the **C**onnectivity **C**hecks timer is triggered or when the connectivity checks for all **candidate pairs** is complete. Connectivity checks for

a **User Datagram Protocol (UDP)** candidate pair MUST be considered complete if the candidate pair is in either the "valid" or the "invalid" state. At the end of the connectivity checks phase, if no valid candidate pairs are found, the call MUST fail. If the connectivity checks are successful, the candidate pair with the highest priority MUST be selected for the final media flow. Any connectivity check packet received after the completion of the connectivity checks phase SHOULD be discarded. If not, the packet MUST be processed in the same way as if the packet was received during the connectivity checks phase.

3.1.4.8.3 Media Flow

This section specifies the **candidate pair** that is used for media flow during processing, as designated by this protocol. Applications can begin sending media after the initial exchange of **candidates** is finished. Any media sent at this stage MUST be sent using the **default candidate pair**. However, there is no guarantee that the media will reach the **peer** at this stage. During the **connectivity checks** phase, media SHOULD be switched to use the first candidate pair that becomes "Recv-Valid" for **User Datagram Protocol (UDP)** or "Valid" for **Transmission Control Protocol (TCP)**. This happens even if those candidates have not been exchanged through the signaling channel. After the final exchange of the candidates selected by the connectivity checks phase, media flow MUST be switched to use the best candidate pair exchanged. **Endpoints** that follow this protocol SHOULD be prepared to accept media on any of the published candidates' **local transport addresses**.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Processing TURN Messages

The processing of **Traversal Using Relay NAT (TURN)** messages, response generation, and error handling is performed as specified in [MS-TURN] when communicating with a **TURN server** as specified in [MS-TURN].

3.1.5.2 Processing STUN Messages

This protocol sends peer-to-peer **Simple Traversal of UDP through NAT (STUN)** messages between **endpoints** during the **connectivity checks** phase to select the **candidate pairs** for streaming media.

3.1.5.2.1 STUN Binding Request

This section specifies the processing of **Simple Traversal of UDP through NAT (STUN)** binding request messages by the two **endpoints**. The processing consists of two tasks. The first task is the validation of the STUN binding request message and the generation of the response. The second task consists of updating **transport address pair** state values and discovering **peer-derived candidates**.

3.1.5.2.1.1 Processing the STUN Binding Request

If a **Simple Traversal of UDP through NAT (STUN)** binding request message is received before the **remote candidates** are received from the **peer endpoint** in the **offer** and **answer**, the endpoint MUST validate the request. If the request is invalid, the endpoint SHOULD send a binding error response for the STUN binding request message, as specified in section 3.1.5.2.1.2. If the request is valid, the endpoint MUST send a STUN binding response message, as specified in section 3.1.5.2.1.3. In addition, the STUN binding request message MUST be cached. When the peer endpoint's **candidates** are received and **candidate pairs** are formed, the cached requests MUST be processed and the candidate pair states MUST be updated accordingly. Additional responses or error responses MUST NOT be sent for the cached requests because they have already been acknowledged.

If a STUN binding request message is received after the remote candidates have been received from the peer in an offer and answer, or if a cached request is being processed, the **USERNAME** attribute in the STUN binding request message is used to identify the **transport address pair** for which the STUN binding request message was sent, by comparing the complete **USERNAME** in the STUN binding request message with each transport pair identifier. This transport address pair is called the **matching transport address pair** for that STUN binding request message. If no matching transport address pair is found, the STUN binding request message MUST be discarded. The corresponding candidate pair, to which the transport address pair belongs, is called the matching candidate pair. If the matching transport address pair is already in the "Invalid" state, the STUN binding request message MUST be discarded.

3.1.5.2.1.2 Validating the STUN Binding Request

The validation procedures for **Simple Traversal of UDP through NAT (STUN)** binding request messages as specified in [IETFDRAFT-STUN-02] differ from the procedures described in this section. **Endpoints** that follow this protocol MUST follow the procedures in this section to validate the STUN binding request messages that are received for **connectivity checks**.

If a STUN binding request message is received without a **USERNAME** attribute, the STUN binding request message MUST be discarded. The **USERNAME** is considered valid if the leftmost portion, up to but excluding the second colon, matches the **transport address** identifier of one of the local transport addresses. If the **USERNAME** is not valid, the message MUST be discarded. If the STUN binding request message does not have the **MESSAGE-INTEGRITY** attribute, the endpoint MUST send a binding error response with error code 401 (Unauthorized), as specified in [IETFDRAFT-STUN-02]. If **MESSAGE-INTEGRITY** exists, the password of the corresponding **local candidate** MUST be used to compute the message integrity and to verify against the message integrity value in the request. If the message integrity check fails, the endpoint MUST send a binding error response with the error code 431 (Integrity Check Failure), as specified in [IETFDRAFT-STUN-02]. Generated binding error responses MUST have a **USERNAME** set to the **USERNAME** received in the STUN binding request message.

3.1.5.2.1.3 Sending the STUN Binding Response

If the request is valid, the **endpoint** MUST send a **Simple Traversal of UDP through NAT (STUN)** binding response message, as specified in [IETFDRAFT-STUN-02] section 7 and [IETFDRAFT-STUN-02] section 10, with a subset of attributes as specified in [IETFDRAFT-STUN-02] section 10.2. The STUN binding response message MUST implement only the following attributes:

- **XOR-MAPPED-ADDRESS**
- **USERNAME**
- **MESSAGE-INTEGRITY**

The format of the **XOR-MAPPED-ADDRESS** attribute MUST be as specified in [IETFDRAFT-STUN-02] section 10.2.12. The **Type** field of **XOR-MAPPED-ADDRESS** attribute MUST have a value of 0x0020. **X-PORT** and **X-ADDRESS** MUST be computed as specified in [IETFDRAFT-STUN-02] section 10.2.12 from the IP address and port from which the STUN binding request message was received. The **USERNAME** attribute MUST have the same value as the **USERNAME** attribute in the corresponding STUN binding request message. The **MESSAGE-INTEGRITY** attribute MUST have the message integrity value that is computed by using the password of the **local candidate** in the matching **candidate pair**.

3.1.5.2.1.4 Learning Peer-Derived Candidates

For a **Simple Traversal of UDP through NAT (STUN)** binding request message that resulted in the generation of a success response, the source IP address and port are compared to the remote **transport address** in the **matching transport address pair** for the STUN binding request message. If they do not match, a new **peer-derived transport address** has been discovered. The procedures

for learning and processing new **peer-derived candidates** from the STUN binding request message for **User Datagram Protocol (UDP)** are performed as specified in [IETFDRAFT-ICENAT-06] section 7.10.1. The procedures for learning and processing new peer-derived candidates from the STUN binding request message for **Transmission Control Protocol (TCP)** are performed as specified in [IETFDRAFT-TCPCICE-00] section 9.

3.1.5.2.1.5 Updating the Transport Addresses Pair State for UDP

For a **Simple Traversal of UDP through NAT (STUN)** binding request message that resulted in the generation of a success response, the **transport addresses** pair state MUST be updated for **User Datagram Protocol (UDP) candidate pairs** as specified in [IETFDRAFT-ICENAT-06] section 7.6. If the **matching transport address pair** is already in a "Valid" state, further state updates MUST NOT be done. If a candidate pair becomes "Valid" as a result of this state update—that is, if all the **transport address pairs** in that candidate pair are "Send-Valid" and "Recv-Valid"—no additional STUN binding request messages SHOULD be sent for those candidate pairs that are lower in priority than the matching candidate pair.

3.1.5.2.1.6 Updating the Transport Addresses Pair State for TCP

For a **Simple Traversal of UDP through NAT (STUN)** binding request message that results in the generation of a success response, the **transport addresses** pair state MUST be updated for **Transmission Control Protocol (TCP) candidate pairs**, as specified in [IETFDRAFT-TCPCICE-00] section 7. If the **matching transport address pair** is already in a "Valid" state, further state updates MUST NOT be done. If all **transport address pairs** in a candidate pair become "Valid" as a result of this state update, additional STUN binding **connectivity check** requests SHOULD NOT be sent for those candidate pairs that are lower in priority than the matching candidate pair.

3.1.5.2.2 STUN Binding Response

This section specifies the way an **endpoint** processes **Simple Traversal of UDP through NAT (STUN)** binding response messages. The processing consists of two tasks. The first task is the validation of the STUN binding response message. The second task is the **connectivity check** processing, which includes updating the state of the **transport address pairs** and discovering **peer-derived candidates**.

3.1.5.2.2.1 Validating the STUN Binding Response

If a **Simple Traversal of UDP through NAT (STUN)** binding response message is received before the **peer's candidates** are received through the **offer** exchange, it MUST be discarded. If a STUN binding response message is received without a **USERNAME** attribute, it MUST be discarded. The **USERNAME** attribute MUST be used to find the **matching transport address pair** for which the STUN binding response message is received. If a matching transport address pair is not found, the STUN binding response message MUST be discarded. If the **transport address pair** is in an invalid state, the STUN binding response message MUST be discarded.

The transaction identifier MUST be checked to see whether the transaction identifier on the response matches the transaction that was used for the corresponding request. If the transaction identifier does not match, the STUN binding response message MUST be discarded. If the STUN binding response message does not have a **MESSAGE-INTEGRITY** attribute, it MUST be discarded.

The password of the corresponding **remote candidate** MUST be used to compute the message integrity. The computed message integrity value MUST be verified against the **MESSAGE-INTEGRITY** attribute value in the message. If the message integrity check fails, the STUN binding response message MUST be discarded. If the message does not have the **XOR-MAPPED-ADDRESS** attribute, the STUN binding response message MUST be discarded. If the IP address in **XOR-MAPPED-ADDRESS** is null ("0.0.0.0"), "Broadcast", or "Multicast", the STUN binding response message MUST be discarded.

3.1.5.2.2.2 Learning Peer-Derived Candidates

For a **Simple Traversal of UDP through NAT (STUN)** response that successfully passes the message validation checks, the source IP address and port are extracted from the **XOR-MAPPED-ADDRESS** attribute of the message by performing the same **XOR** operations specified during the creation of the **XOR-MAPPED-ADDRESS** attribute in section 3.1.5.2.1.3. The IP address and port are compared to the **local transport address** in the **matching transport address pair** for the STUN binding response message. If they do not match, a new **peer-derived transport address** has been discovered. The procedures for learning and processing new **peer-derived candidates** from the STUN binding request message for **User Datagram Protocol (UDP)** are performed as specified in [IETFDRAFT-ICENAT-06] section 7.10.2. The procedures for learning and processing new peer-derived candidates from the STUN binding response message for **Transmission Control Protocol (TCP)** are performed as specified in [IETFDRAFT-TCPCICE-00] section 9.

3.1.5.2.2.3 Updating the Transport Addresses Pair State for UDP

For a valid **Simple Traversal of UDP through NAT (STUN)** binding response message, the **candidate pair** state MUST be updated for **User Datagram Protocol (UDP)** candidate pairs as specified in this [IETFDRAFT-ICENAT-06] section 7.6. If the **matching transport address pair** is already in the "Valid" state, further state updates MUST NOT be done. If a candidate pair becomes "Valid" as a result of this state update,—that is, if all the **transport address pairs** in that candidate pair are "Send-Valid" and "Recv-Valid"—additional STUN binding **connectivity check** requests SHOULD NOT be sent for those candidate pairs that are lower in priority than the matching candidate pair.

3.1.5.2.2.4 Updating the Transport Addresses Pair State for TCP

For a **Simple Traversal of UDP through NAT (STUN)** binding response that was successfully validated, the **transport addresses** pair state MUST be updated for **Transmission Control Protocol (TCP) candidate pairs** as specified in [IETFDRAFT-TCPCICE-00] section 7. If the **matching transport address pair** is already in the "Valid" state, further state updates MUST NOT be done. If all **transport address pairs** in the TCP candidate pair become "Valid" as a result of this state update, additional STUN binding **connectivity check** requests SHOULD NOT be sent for those candidate pairs that are lower in priority than the matching candidate pair.

3.1.5.2.2.5 STUN Binding Error Response

The error response message MUST be validated in the same way as **Simple Traversal of UDP through NAT (STUN)** binding response messages. The validation procedure is specified in section 3.1.5.2.2.1.

If the **transport address** for which the error response is received is already in the "Recv-Valid" or "Valid" state for **User Datagram Protocol (UDP)** or in the "Valid" state for **Transmission Control Protocol (TCP)**, the error response message MUST be discarded. If the error code in the error response message is 401, 430, 431, 432, or 500, **connectivity checks** for the transport address SHOULD be retried. If any other error code is received in the binding error response message, the **transport address pair** MUST be set to the "Invalid" state.

3.1.6 Timer Events

3.1.6.1 Candidates Gathering Phase Timer

The firing of the **Candidates Gathering Phase** timer signals the end of the **candidates** gathering phase. The **endpoint** MUST exchange the gathered **local candidates** with its **peer**.

3.1.6.2 Connectivity Checks Phase Timer

After a **Simple Traversal of UDP through NAT (STUN)** binding request message and response are received from the **peer**, the **Connectivity Checks Phase** timer MUST be reset to 3 seconds. The firing of this timer signals the end of the **connectivity checks** phase. When this timer fires, the **caller** MUST pick the best **candidate pair** selected by the connectivity checks and send them to the **callee**. If no candidate pair is validated by the connectivity checks when the timer fires, the call MUST fail. Further connectivity check attempts MUST NOT be made after this timer fires.

3.1.6.3 ICE keep-alive Timer

The **ICE keep-alive** timer MUST fire when there has been no flow of media or **ICE keep-alive messages** for the duration of the timer. When the **ICE keep-alive timer** fires, an ICE keep-alive message MUST be sent only for the **Real-Time Transport Protocol (RTP) component's transport address pair** that is associated with the **candidate pair** that is currently being using for media flow. The ICE keep-alive messages are sent from the **local transport address** to the remote **transport address** in the transport address pair. ICE keep-alive messages SHOULD NOT be sent for an **Real-Time Transport Control Protocol (RTCP)** component because the flow of **RTCP packets** is sufficient to keep the **NAT bindings** and **Traversal Using Relay NAT (TURN)** allocations active. ICE keep-alive messages MUST be sent even if the **peer endpoint** does not implement Interactive Connectivity Establishment (ICE) for the RTP component's transport address pair that is associated with the candidate pair that is used for media flow. ICE keep-alive messages MUST be **Simple Traversal of UDP through NAT (STUN)** binding request messages, as specified in section 2.2.3.

3.1.7 Other Local Events

None.

4 Protocol Examples

This protocol follows a protocol example similar to the one described in [IETFDRAFT-ICENAT-06] section 11, with the exception of the **Simple Traversal of UDP through NAT (STUN)** server interaction in the **candidates** gathering phase. This protocol suggests using messages described in [MS-TURN] to communicate with a **TURN server** to gather both its **STUN candidates** and its **TURN candidates**.

5 Security

5.1 Security Considerations for Implementers

This protocol has similar security concerns as those described in [IETFDRAFT-ICENAT-06]. Additional considerations and mitigations pertaining to this protocol are listed in this section.

5.1.1 Attacks on Address Gathering

The security considerations for gathering **STUN candidates** and **TURN candidates** are described in [MS-TURN] section 5.1.

5.1.2 Attacks on Connectivity Checks

An attacker might attempt to sniff the signaled **candidates** and passwords to maliciously obtain control of the call and related media. This protocol relies on the existence of a secure channel to exchange candidates. A malicious user might attempt to attack the **STUN connectivity checks** either to maliciously gain control of the call and related media to a different **endpoint** or to cause a failure of the connectivity checks. The malicious user can potentially inject connectivity check packets to fool an endpoint into considering a valid **transport address pair** invalid or vice versa. Alternatively, the malicious user can cause the endpoints to discover incorrect **peer-derived candidates**. These attacks are mitigated by this protocol by mandating the **MESSAGE-INTEGRITY** attribute in the STUN connectivity checks and responses.

5.1.3 Voice Amplification Attack

A malicious user can include the target address of the denial of service attack as the **default candidate** in its **offer** and send the offer to multiple **endpoints**. This action can potentially result in each endpoint that received the offer attempting to send media to the target of the denial of service attack. This attack can be mitigated by using this protocol in conjunction with a secure signaling layer for offer exchange that is associated with targeted **candidates** and associated credentials.

5.1.4 STUN Amplification Attack

The **Simple Traversal of UDP through NAT (STUN)** amplification attack is similar to the voice amplification attack. Instead of media flow, the STUN **connectivity checks** are directed to the target of the denial of service attack. The malicious user proceeds by generating an **offer** with a large number of **candidates** for the denial of service target. The **peer endpoint**, after receiving the offers, performs connectivity checks with all the candidates specified on the offer. This malicious activity can generate a significant volume of data flow with STUN connectivity checks. This malicious activity cannot be completely prevented by this protocol, but the protocol can mitigate this type of malicious activity to a certain extent by limiting the total number of candidates that are sent in an offer or response to 20 candidates and 40 **candidate pairs**. In addition, this protocol relies on a secure signaling layer for offer exchanges of candidates and associated user names and passwords.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include 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 2010
- Microsoft Lync Server 2010
- Microsoft Lync Client 2013/Skype for Business
- Microsoft Lync Server 2013
- Microsoft Skype for Business 2016
- Microsoft Skype for Business Server 2015
- Microsoft Skype for Business 2019 Preview

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.

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	Updated list of supported products.	major

8 Index

V