[MS-ICE2]:
Interactive Connectivity Establishment (ICE) Extensions 2.0

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp) or the Community Promise (available here: http://www.microsoft.com/interop/cp/default.mspx). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
12/12/2008	1.0		Initial version
02/13/2009	1.01		Revised and edited the technical content.
03/13/2009	1.02		Revised and edited the technical content.
07/13/2009	1.03	Major	Revised and edited the technical content
08/28/2009	1.04	Editorial	Revised and edited the technical content
11/06/2009	1.05	Editorial	Revised and edited the technical content
02/19/2010	1.06	Editorial	Revised and edited the technical content
03/31/2010	1.07	Major	Updated and revised the technical content
04/30/2010	2.08	Editorial	Revised and edited the technical content
06/07/2010	2.09	Editorial	Revised and edited the technical content
06/29/2010	2.10	Editorial	Changed language and formatting in the technical content.
07/23/2010	2.10	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	3.0	Major	Significantly changed the technical content.
11/15/2010	3.0	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	3.0	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document specifies the Interactive Connectivity Establishment (ICE) Extensions protocol. It is a proprietary extension to the ICE protocol. ICE specifies a protocol for setting up Real-Time Transport Protocol (RTP) streams in a way that allows the streams to traverse network address translation (NAT) and firewalls.

Signaling protocols such as the Session Initiation Protocol (SIP) are used to set up and negotiate media sessions. As part of setting up and negotiating the session, signaling protocols carry the Internet Protocol (IP) addresses and ports of the call participants that receive RTP streams. Because NATs alter IP addresses and ports, 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.

1.1 Glossary

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

authentication
Internet Protocol version 4 (IPv4)
NAT binding
network address translation (NAT)
Transmission Control Protocol (TCP)
User Datagram Protocol (UDP)

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

agent
answer
callee
caller
candidate
candidate pair
Check List
component
connectivity check
controlled agent
controlling agent
default candidate
default candidate pair
endpoint
final offer
full
Host Candidate
ICE keep-alive message
initial offer
INVITE
local candidate
local transport address
offer
peer
peer-derived candidate
provisional answer
Real-Time Transport Control Protocol (RTCP)
Real-Time Transport Protocol (RTP)

Relayed Candidate
remote candidate
remote endpoint
RTCP packet
SDP offer
Server Reflexive Candidate
Session Description Protocol (SDP)
Session Initiation Protocol (SIP)
Simple Traversal of UDP through NAT (STUN)
transport address
Traversal Using Relay NAT (TURN)
TURN candidate
TURN server

The following terms are specific to this document:

Aggressive Nomination: The process of selecting a valid candidate pair for media flow by sending Simple Traversal of UDP through NAT (STUN) binding requests that include the flag for every STUN binding request such that the first candidate pair that is validated is used for media flow.

base: The base of a host candidate is the host candidate itself. The base of server reflexive candidates and peer reflexive candidates is the host candidate from which they are derived. The base of a relayed candidate is the relayed candidate itself.

foundation: A string that is a property associated with a candidate. The string is the same for candidates that are of the same type, protocol, and base IP addresses, and are obtained from the same STUN/TURN server for relayed and server reflexive candidates.

Lite: An implementation that supports a minimal subset of Interactive Connectivity Establishment (ICE) functionality, as described in [MS-ICE2], to work with a Full ICE implementation. A Lite implementation responds to but does not send connectivity checks.

nominated: A candidate pair for which the nominated flag is set.

Ordinary Check: A connectivity check that is generated periodically by an endpoint (5) based on the timers for connectivity checks.

Regular Nomination: The process of selecting a valid candidate pair for media flow by validating the candidate pairs with Simple Traversal of UDP through NAT (STUN) binding requests, and then selecting a valid candidate pair by sending STUN binding requests with a flag indicating that the candidate pair was nominated.

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

triggered check: A connectivity check that is generated in response to a connectivity check packet that is received from a peer.

Valid List: A list of candidate pairs that have been validated by connectivity checks.

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

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[IETFDRAFT-ICENAT-19] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-19, October 2007, http://tools.ietf.org/html/draft-ietf-mmusic-ice-19

[IETFDRAFT-ICETCP-07] Rosenberg, J., "TCP Candidates with Interactive Connectivity Establishment (ICE)", draft-ietf-mmusic-ice-tcp-07, July 2008, http://tools.ietf.org/html/draft-ietf-mmusic-ice-tcp-07

[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

[MS-TURN] Microsoft Corporation, "Traversal Using Relay NAT (TURN) Extensions", June 2008.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

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

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary", June 2008.

[MS-SDPEXT] Microsoft Corporation, "Session Description Protocol (SDP) Version 2.0 Protocol Extensions", June 2008.

1.3 Protocol Overview (Synopsis)

This protocol is used to establish media flow between a **callee endpoint (5)** and a **caller** endpoint (5). In typical deployments, a **network address translation (NAT)** or firewall might exist between the two endpoints (5) 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 (5) belong. This type of deployment blocks incoming traffic. If the endpoint (5) advertises its local interface address, the **remote endpoint** might not be able to reach it.

The address exposed by NAT or the firewall is not exactly what the endpoints (5) need to determine the external routable mapping address created by the NAT, or the NAT-mapped address, for its local interface address. Moreover, NATs and firewalls exhibit differing behavior in the way they create the NAT-mapped addresses. ICE provides a generic mechanism to assist media in traversing NATs and firewalls without requiring the endpoints (5) 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 (5) can potentially use to communicate, and then ICE determines which transport address is best for both endpoints (5) to use to establish a media session.

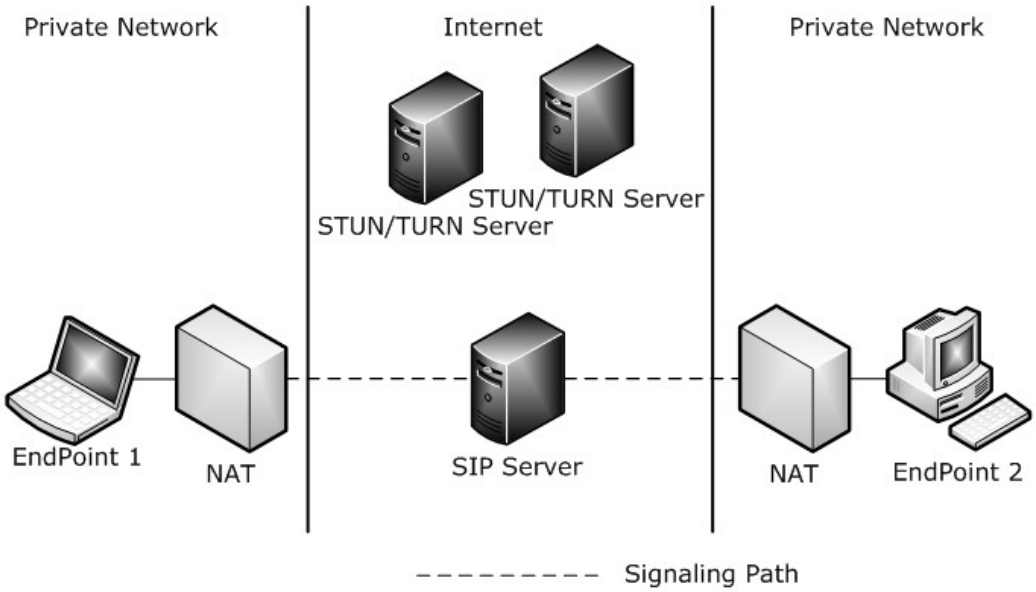


Figure 1: ICE deployment scenario

The preceding figure shows a typical deployment scenario with two endpoints (5) that establish a media session. To facilitate ICE, a communication channel through which the endpoints (5) can exchange messages, such as **Session Description Protocol (SDP)**, using a signaling protocol, such as the **Session Initiation Protocol (SIP)**, is necessary. ICE assumes that such a channel exists and is not intended to be used for NAT traversal for these signaling protocols. ICE is often deployed in conjunction with **Simple Traversal of UDP through NAT (STUN)** and **Traversal Using Relay NAT (TURN)** servers. The endpoints (5) can share the same STUN and **TURN servers** or use different servers.

The sequence diagram in the following figure outlines the various phases involved in establishing a session between two endpoints (5) using this protocol.

- **Candidates** gathering and exchange of gathered transport addresses between the caller and callee endpoints (5).
- **Connectivity checks**.
- Exchange of candidates selected by connectivity checks.

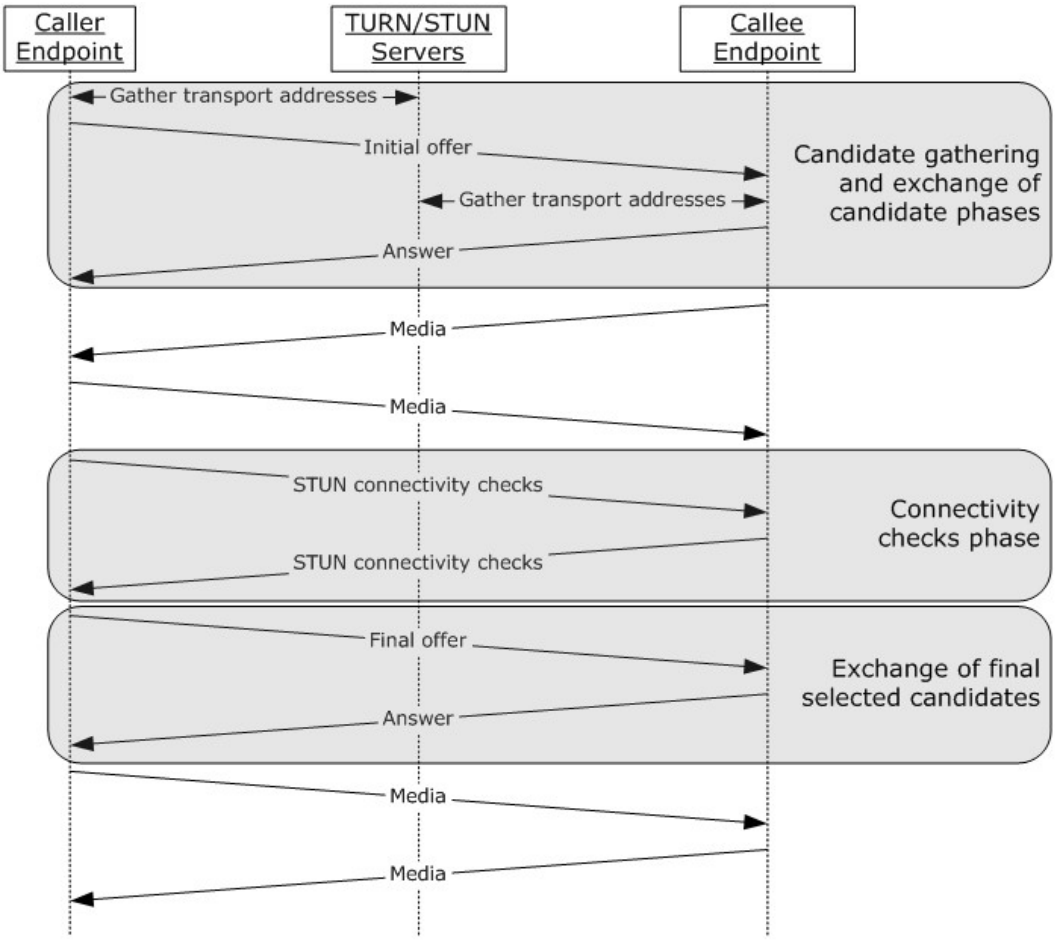


Figure 2: ICE sequence diagram

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 **Host Candidate**.
- Transport addresses that are mappings on the public side of a NAT, which is a **Server Reflexive Candidate**.
- Transport addresses allocated from a TURN server, which is a **Relayed Candidate**.

The gathered transport addresses are used to form candidates. A candidate is a set of transport addresses that can potentially be used for media flow. For example, in the case of real-time media flow using the **Real-Time Transport Protocol (RTP)**, each candidate consists of two **components**, one for RTP and another for **Real-Time Transport Control Protocol (RTCP)**.

Each gathered candidate is assigned a **foundation** and a priority value based on how they were obtained. This priority indicates the preference of an endpoint (5) to use one candidate over another if both candidates are reachable from the peer. The foundation is a string associated with each

candidate. Two candidates have the same foundation if they are of the same type. Types of candidates are Host Candidates, Server Reflexive Candidates, Relayed Candidates, or **peer-derived candidates**. In addition to matching types, to have the same foundation the two candidates have the same **base** and are derived from the same STUN or TURN server. Candidates obtained from local network interfaces are often given a higher priority than the candidates obtained from TURN servers. The endpoint (5) 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 can be encoded into an **SDP offer** and exchanged over a signaling protocol such as SIP. The caller endpoint (5) serves as the **controlling agent** and is responsible for selecting the final candidates for media flow.

The callee, after receiving the offer, follows the same procedure to gather its candidates. The gathered candidates are encoded and sent to the caller in the **answer**. With the exchange of candidates complete, both the endpoints (5) are now aware of their peer's candidates.

The start of the connectivity checks phase is triggered at an endpoint (5) when it is aware of its peer's candidates. Both endpoints (5) pair up the **local candidates** and **remote candidates** to form a **Check List** of **candidate pairs** that are ordered based on the priorities of the candidate pairs. Each candidate pair consists of constituent component pairs and has the same foundation as the candidate pair. In the case of RTP, each candidate pair has an RTP component pair and an RTCP component pair. The candidate pair priorities are computed using the priorities of the local candidate and the remote candidate so that both endpoints (5) have the same ordering of candidate pairs. Each candidate pair has an associated foundation that is formed as a concatenation of the foundations of the local candidate and the remote candidate that constitute the candidate pair. Candidate pairs with the same foundations have similar network properties, and this is leveraged to reduce the number of connectivity checks. If connectivity checks for a component pair fail, it is very likely that connectivity checks for other component pairs with the same foundation will also fail. Each endpoint (5) goes through the candidate pair Check List and sets the state of the higher component pair, or the RTCP component pair, to a frozen state. If more than one candidate pair has the same foundation, all candidate pairs except for the highest priority candidate pair with the same foundation are set to a frozen state. When the connectivity check for a component pair succeeds, all component pairs with the same foundations are unfrozen. The callee serves as the **controlled agent** and waits for the controlling agent to select the final candidate pair for media flow.

Both endpoints (5) systematically perform connectivity checks, starting from the top of the candidate pair Check List to determine the highest priority candidate pair that can be used by the endpoints (5) 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 component pair, the component pair is considered to be in "Succeeded" state.

The endpoints (5) can begin 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 candidate pairs are spaced at regular intervals to avoid flooding the network. Depending on the topology, many of the 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, as both endpoints (5) are behind NATs. These connectivity checks, sent periodically to validate the candidate pairs, are called **Ordinary Checks**. In addition, to optimize the connectivity checks, an endpoint (5), on receiving a STUN binding request for a candidate pair, immediately

schedules a connectivity check for that candidate pair. These connectivity checks are called **triggered checks**.

The endpoints (5) 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 transport address that does not match any of the remote candidates.
- The STUN binding response message has a mapped address that does not match the transport address of any of the local candidates.

These scenarios arise if new external mappings are created by the NATs residing between the endpoints (5). 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.

The controlling agent concludes the connectivity checks by nominating a valid candidate pair found by the connectivity checks for media flow. The controlling agent can follow either **Regular Nomination** or **Aggressive Nomination** to nominate the validated candidate pairs. If the controlling agent is following Regular Nomination, it allows connectivity checks to continue until at least one valid candidate pair has been found. At the end of the connectivity checks, the controlling agent picks the best valid candidate pair from the **Valid List** and sends another round of STUN binding requests for this candidate pair with a flag set to notify the peer that this candidate pair has been **nominated** for media flow. In the case of Aggressive Nomination, the controlling agent sets this flag on every STUN binding request. With Aggressive Nomination, the ICE processing completes when connectivity checks succeed for the first candidate pair, and the controlling agent does not have to send a second STUN binding request to nominate the candidate pair. Aggressive Nomination is faster than Regular Nomination, but does not always select the optimal path that has the lowest latency. At the end of the connectivity checks phase, the controlling agent 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 (5) begin using the selected candidate pair for media flow.

1.4 Relationship to Other Protocols

This protocol is an application layer protocol that depends on, and works with, **Transmission Control Protocol (TCP)** and **User Datagram Protocol (UDP)** transport protocols for **Internet Protocol version 4 (IPv4)** addresses only.

This protocol works with implementations of TURN protocols, as described in [MS-TURN], to create **TURN candidates** and **STUN candidates**.

This protocol can perform connectivity checks only with endpoints (5) that follow the message formats in the STUN specifications and follow the STUN attributes and usage specification in section 3.1.4.3.

This protocol depends on signaling protocols, such as SIP, to perform an offer and answer exchange of encoded messages, such as SDP messages described in [MS-SDPEXT].

This protocol is used to establish a communication channel which is eventually used for media flow for protocols such as RTP and RTCP.

1.5 Prerequisites/Preconditions

This protocol requires that the endpoints (5) are able to communicate through a signaling protocol, such as SIP, to exchange candidates.

1.6 Applicability Statement

This protocol is a **full** implementation, and requires the peer endpoint (5) to perform Regular Nomination. It does not support or work with peer endpoints (5) that perform Aggressive Nomination.

This protocol treats a **Lite** implementation peer as a peer that does not support ICE and does not follow the procedures for handling a Lite implementation peer.

This protocol treats each stream in a session independently for ICE processing, if the session has more than one stream. The procedures specified in this protocol are per media stream.

This protocol does not support ICE Restarts.

This protocol requires TURN servers to be deployed to facilitate communication across NATs and firewalls. In the absence of TURN servers, this protocol might not be able to establish connectivity between endpoints (5) in such topologies.

This protocol is appropriate for establishing a communication channel between two endpoints (5) for media exchange.

This protocol can operate in two modes, regular mode and TCP-only mode. This protocol cannot be used for establishing a communication channel through TCP in the absence of a TURN server in regular mode. Both the caller and callee endpoint (5) support and operate in the same mode for this protocol to establish connectivity.

This protocol is used to establish connectivity for streaming RTP media. As a result, this protocol supports 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 RTCP. As a result, endpoints (5) that need to communicate with an endpoint (5) that implements this protocol are required to support sending and receiving media to RTP and RTCP on non-consecutive ports, whether or not they support ICE itself.

This protocol multiplexes both the components to the same IP address and port when the connection is established through TCP. The application layer is required to de-multiplex 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 (5) multiplex components over TCP.

This protocol does not support multiplexing of RTP and RTCP components to the same IP address and port when the connection is established over UDP.

During the connectivity checks, **ICE keep-alive messages** are sent for both RTP and RTCP components for validated component pairs and for candidate pairs whose local candidates are Relayed Candidates. For the candidate that is being used for media flow, the ICE keep-alive messages are sent only for the RTP component's transport addresses. **RTCP packets** are sent to keep the NAT bindings and TURN allocations active for the RTCP component's transport addresses. ICE keep-alive messages are sent regardless of whether UDP or TCP is the underlying transport.

1.7 Versioning and Capability Negotiation

Currently, this protocol has no versioning and capability negotiation constraints, with the following exception.

This protocol is implemented on top of the TCP (IPv4) and UDP (IPv4) transport protocols, as described in section 2.1.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

This protocol uses the TCP (IPv4) and UDP (IPv4) transport protocols. Endpoints (5) 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.

2.2 Message Syntax

This section specifies the various messages used by the implementation of this protocol. 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 the [MS-TURN] protocol to discover Server Reflexive Candidates and Relayed Candidates. The message syntax used by the endpoint (5) implementing [MS-TURN] to communicate with the TURN server is specified in [MS-TURN] section 2.

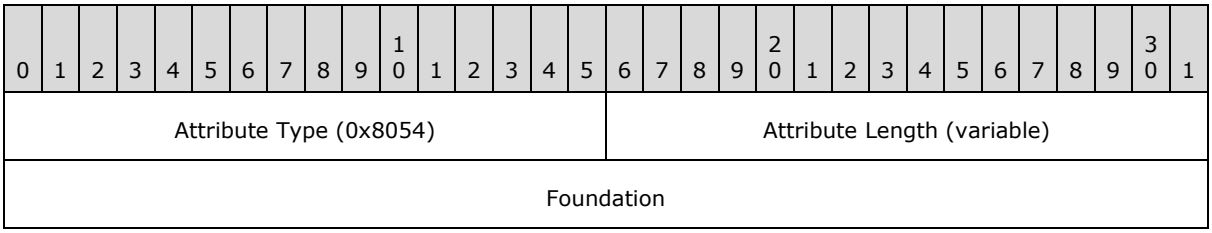
2.2.2 STUN Messages

This protocol uses STUN binding request and response messages for connectivity checks between the two endpoints (5). The STUN messages MUST follow the message formats specified in [IETFDRAFT-STUN-02] section 10. STUN messages sent over TCP MUST follow the framing method specified in [RFC4571] section 2. This method is required to de-multiplex the received application data and STUN packets. STUN messages MUST support the STUN extensions and attributes specified in [IETFDRAFT-ICENAT-19] section 19.

This protocol defines two additional attributes, **CANDIDATE-IDENTIFIER** and **IMPLEMENTATION-VERSION**, which MUST be supported per the procedures in [IETFDRAFT-STUN-02] section 10.2. The **CANDIDATE-IDENTIFIER** attribute MUST be sent only with STUN binding request messages. The **IMPLEMENTATION-VERSION** attribute MUST be added to all STUN binding request and response messages.

2.2.2.1 CANDIDATE-IDENTIFIER

The **CANDIDATE-IDENTIFIER** attribute MUST be added to STUN binding request messages that are sent for connectivity checks. The **CANDIDATE-IDENTIFIER** attribute is used to identify the remote candidate from which the connectivity check is received. The value of **CANDIDATE-IDENTIFIER** MUST be a valid foundation string. If the length of the **CANDIDATE-IDENTIFIER** value is not at a four byte boundary, the value MUST be padded with NULLs to be at a four byte boundary on the wire. The usage of this attribute MUST follow the specification in section 3.1.4.8.2.4. The format of this attribute is as follows.



Attribute Type (2 bytes): 0x8054 specifies the type of the attribute.

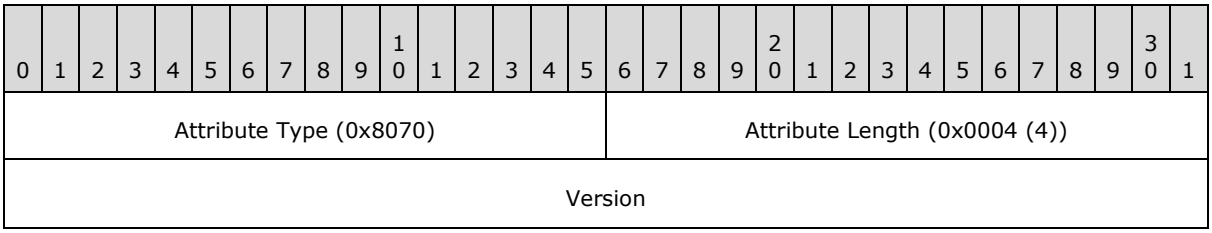
Attribute Length (2 bytes): (variable) specifies the length of the attribute.

Foundation (variable bytes): The value of this attribute MUST be set to the foundation of the local candidate for which the request is being sent, if the candidate is not peer-derived. If the local candidate is peer-derived, the value MUST be set to the foundation of the peer-derived local candidate's base.

2.2.2.2 IMPLEMENTATION-VERSION

This section follows the behavior described in the endnote <1>.

The **IMPLEMENTATION-VERSION** attribute is the ICE protocol implementation version. This attribute SHOULD be included in all connectivity check request and response messages. The format of this attribute is as follows.



Attribute Type (2 bytes): 0x8070 specifies the type of the attribute.

Attribute Length (2 bytes): 0x0004 (4) specifies the length of the attribute.

Version (4 bytes): ICE implementations MUST be set to the version as described in the product behavior<2>.

2.2.3 ICE keep-alive Message

The ICE keep-alive message MUST be a valid STUN binding request message, as specified in [IETFDRAFT-STUN-02], which MUST follow the additional specifications in this section. ICE keep-alive messages sent over TCP MUST follow the framing method specified in [RFC4571] section 2. The **transaction ID** can be any valid transaction identifier. The ICE keep-alive message MUST have the **MESSAGE-INTEGRITY** attribute set to a value of zero ("0") or a valid message integrity value. The ICE keep-alive message MUST NOT have any other attributes.

3 Protocol Details

3.1 Common Details

The procedures specified apply to both TCP and UDP transport protocols unless the procedures explicitly specify a transport protocol. This protocol MUST support operating in either the regular mode or the TCP-only mode based on the cue from the application layer that builds on top of this protocol. By default this protocol operates in the regular mode. The differences between the operating modes are only during the candidates gathering phase specified in section 3.1.4.8.1.

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-19] section 7.

3.1.2 Timers

Candidates Gathering Phase timer: This timer tracks the maximum duration for the candidate gathering phase. This timer has a default value, as specified in section 3.1.6.1.

Connectivity Phase timer: This timer tracks the maximum duration for which connectivity checks can be performed between the candidate pairs. The values for this timer are specified in section 3.1.6.2.

ICE keep-alive timer: This timer tracks the spacing of ICE keep-alive messages. These messages are sent to keep the NAT bindings and TURN allocations active. The default value is specified in section 3.1.6.3.

USE-CANDIDATE Checks timer: This timer tracks the maximum duration for which **USE-CANDIDATE** checks can be performed to nominate the candidate pairs selected by connectivity checks as part of Regular Nomination. The values for this timer are specified in section 3.1.6.4. This timer is applicable only for the controlling agent.

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 of candidate lists during and after the connectivity checks is allowed by [IETFDRAFT-ICENAT-19] section 9.3.1.4. This extension specifies that there MUST NOT be an additional offer or exchange of candidates other than those specified in this section. Processing for this protocol is specified for each media stream. If connectivity has to be established for more than one media stream, connectivity establishment MUST be carried out independently for each media stream. If the transport address for media or any of the candidates needs to change, the endpoints (5) 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 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 before sending the gathered candidates to the peer endpoint (5) through the pre-established signaling channel. For example, the candidates can be encoded into an SDP offer.

The caller MUST designate one of the local candidates as the default candidate in the initial offer. In the regular mode, the default candidate MUST be a UDP candidate. If no UDP candidate has been gathered, the call MUST fail. In the TCP-only mode, the default candidate MUST be a TCP candidate and no UDP candidates can be gathered or sent in the offer. If no TCP candidate has been allocated, the call MUST fail. Once the candidates have been gathered successfully, the caller SHOULD be ready to respond to connectivity checks from the callee.

3.1.4.2 Receipt of the Initial Offer and Generation of 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 before sending the gathered candidates to the peer through the pre-established signaling channel. For example, the candidates can be encoded into an SDP offer.

The callee MUST designate one of the local candidates as the default candidate in the answer to the initial offer. In the regular mode, the default candidate MUST be a UDP candidate. If no UDP candidates are gathered, the call MUST fail. In the TCP-only mode, the default candidate MUST be a TCP candidate and no UDP candidates can be gathered or sent in the answer. If no TCP candidate is gathered, the call MUST fail.

When the callee receives the initial offer with the caller's candidates, the callee MUST begin the connectivity phase after gathering its local candidates, as specified in section 3.1.4.8.2. Applications that want to reduce the perceived latency of call establishment for the user SHOULD have the callee encode the gathered candidates and send them in a **provisional answer** to the caller before sending out the answer to the initial offer. If an endpoint (5) sends a provisional answer, the subsequent answer for the initial offer MUST have the same set of candidates and default candidate as the provisional answer.

3.1.4.3 Processing of Provisional Answer to Initial Offer

The caller, after receiving the provisional answer with the callee's candidates, MUST begin the connectivity checks as specified in section 3.1.4.8.2. A single initial offer can result in multiple provisional answers being received as a result of forking. The ICE processing MUST be carried out independently for each provisional answer, as specified in [IETFDRAFT-ICENAT-19] section 6.

Implementations of this protocol SHOULD NOT support the processing of more than 20 provisional answers. Implementations of this protocol can support less than 20 provisional answers if the resources are not available to process 20 provisional answers. Provisional answers that arrive after the maximum number of supported provisional answers has been exceeded MUST be ignored.

3.1.4.4 Processing the Answer to the Initial Offer from Full ICE Peer

The caller, upon receiving the answer to its initial offer with the callee's candidates, MUST begin the connectivity checks phase, as specified in section 3.1.4.8.2, if the connectivity checks were not already started as a result of receiving a provisional answer. If a provisional answer was already received from the peer endpoint (5), connectivity checks that were started as a result of processing the provisional answer MUST be continued.

3.1.4.4.1 Processing of Answer to Initial Offer from peer not supporting ICE/supporting Lite implementation

If an answer is received from a peer that does not support ICE or supports Lite implementation, the procedure outlined in this section MUST be followed.

STUN binding request messages MUST be sent by the caller from the default candidate to the transport addresses, one for RTP and one for RTCP, advertised by the peer that does not support ICE.

These STUN binding request messages serve only to open up permissions on the TURN servers and NATs for the peer that does not support ICE. After the answer is received from a peer that does not support ICE or a peer that supports Lite implementation , no further connectivity checks processing or offer and answer exchanges are required. The default candidate advertised in the initial offer MUST be used for media flow to the remote candidate advertised in the answer.

3.1.4.5 Generating the Final Offer

At the end of the connectivity checks phase, the controlling agent MUST send the final offer. The final offer MUST be encoded and MUST contain only the local candidate and remote candidate selected by this protocol, to its peer. For example, the final offer can be encoded into an SDP offer.

The final offer MUST be generated, even if the selected local and remote candidates match the default local and remote candidates, respectively, of the initial offer and answer.

3.1.4.6 Receiving of the Final Offer and Generation of the Answer

The controlled agent, upon receiving the final offer, MUST validate the candidates received in the final offer by verifying that it has a candidate pair that consists of the local and remote candidates in the final offer. If the remote candidate in the final offer is not known, the call MUST fail. If the local candidate in the final offer is not known, the endpoint (5) checks the triggered check queue to see if there are triggered checks queued as a result of the STUN binding request with the **nomination** flag received from the controlling agent during nomination. If no corresponding triggered checks are found, the call MUST fail. If found, these triggered checks are processed until the local candidate that matches the local candidate in the final offer is discovered, or the application layer terminates the call.

If a matching candidate pair for the candidates in the final offer is found, the endpoint (5) MUST switch to using the local and remote candidates in the offer for media flow. It MUST acknowledge the receipt of the final offer similarly, with a response that MUST contain only the local candidate and the remote candidate to be used for media flow. If the selected local candidate is a TURN candidate, a **Set Active Destination** message SHOULD be sent for that candidate. The format for the **Set Active Destination** message and subsequent processing SHOULD follow the specifications in [MS-TURN] section 3.2.5. Local candidates other than the selected local candidate SHOULD be freed.

3.1.4.7 Processing the Answer to Final Offer

The controlling agent, after receiving the answer to its final offer, MUST validate the local candidate and remote candidate in the answer to ensure that they are the same candidates that the controlling agent selected and sent in the final offer. If the validation fails, the call MUST fail. If the answer is successfully validated, the controlling agent MUST switch to using the local and remote candidates in the answer for media flow. An endpoint (5), on 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] section 3.2.5, SHOULD be sent for that candidate.

3.1.4.8 Common Procedures

The following sections specify common procedures triggered by higher-layer events.

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 (5). This section specifies the operations involved in the candidates gathering phase. The candidates gathering phase MUST end when the candidates gathering phase timer fires or when the process of gathering candidates is complete.

Because this protocol is used for streaming RTP media, each candidate MUST have two components. One component is for RTP and the other for RTCP. [IETFDRAFT-ICENAT-19] section 4.1.1 specifies how to gather IPv4 addresses for TCP and UDP transports.

Implementers of this protocol MUST NOT support sending more than 20 candidates in the offer or answer. If an endpoint (5) 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 STUN amplification attack specified in section 5.1.4.

This protocol does not implement Candidate Keep Alives, as specified in [IETFDRAFT-ICENAT-19] section 4.1.1.4. At the end of the candidate gathering phase, redundant candidates MUST be eliminated, as specified in [IETFDRAFT-ICENAT-19] section 4.1.3. The default candidates MUST be selected, as specified in sections 3.1.4.1 and 3.1.4.2.

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:

- UDP Host Candidates
- UDP Server Reflexive Candidates
- UDP Relayed Candidates
- Active/Passive TCP Host Candidates
- Active TCP Server Reflexive Candidates
- Active/Passive TCP Relayed Candidates

The implementer of this protocol MUST NOT support the gathering of other candidate types or candidate behaviors. The RTP and 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 same 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 ports of the gathered transport addresses MUST NOT be in the port range 0-1023.

3.1.4.8.1.2 Gathering UDP Candidates

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 VPN. The candidates MUST be gathered following the specification in [IETFDRAFT-ICENAT-19] section 4.1.

UDP Relayed Candidates SHOULD be obtained following the procedures for allocating candidates on the TURN server, as specified in [MS-TURN] section 3.2.4.1.

UDP Server Reflexive Candidates SHOULD be discovered by following the procedure specified in [MS-TURN] section 3.2.5.1.

Implementations of this protocol SHOULD NOT pair all Host Candidates with the TURN server, as specified in [IETFDRAFT-ICENAT-19] section 4.1.1.2. This protocol selects the best host interface to communicate with a configured TURN server and gathers Server Reflexive Candidates and Relayed Candidates only for that interface. If multiple TURN servers are configured, implementations of this protocol SHOULD gather reflexive and relayed candidates from every configured TURN server.

3.1.4.8.1.3 Gathering TCP Candidates

The gathering of TCP candidates varies based on the operation mode. The following subsections specify the difference in candidate gathering between the two operations modes. This protocol does not support gathering simultaneous-open candidates and does not work with simultaneous-open candidates.

Implementations of this protocol MUST set the ports to any value in the valid port range, which is outside of 0 – 1023. The port value advertised is not important, as the outbound connection for the active candidates is done from ephemeral ports. Implementations of this protocol MUST multiplex both RTP and RTCP on the same port.

3.1.4.8.1.3.1 TCP-Only Mode

In the TCP-only mode of operation, one active and one passive Host Candidate MUST be gathered from every available network interface.

TCP candidates SHOULD be obtained following the procedures for allocating candidates on the TURN server, as specified in [MS-TURN] section 3.2.4.1. If multiple local interfaces are available, Relayed Candidates and Server Reflexive Candidates SHOULD be obtained by selecting the best local interface to communicate with the relay. If multiple TURN servers are configured, implementations of this protocol SHOULD gather reflexive and relayed candidates from every configured TURN server.

Only one Relayed Candidate MUST be obtained per server implemented because the Relayed Candidate gathered following [MS-TURN] section 3.2.4.1 serves as both active and passive candidate. The Relayed Candidate MUST be advertised separately as an active Relayed Candidate and a passive Relayed Candidate in the encoded offer when the candidates are exchanged. The Server Reflexive Candidate obtained from the allocate response SHOULD be advertised as an active Server Reflexive Candidate.

3.1.4.8.1.3.2 Regular Mode

In the regular mode, active and passive Host Candidates are not gathered. Only Relayed Candidates and Server Reflexive Candidates SHOULD be gathered if TURN servers have been configured. The procedures for gathering Server Reflexive Candidates and Relayed Candidates is the same as outlined for TCP-only mode.

When no TCP TURN servers have been configured in regular mode, implementations of this protocol SHOULD create an active Server Reflexive Candidate that has the same IP address as one of the UDP Server Reflexive Candidates, if one exists. If no UDP Server Reflexive Candidates exist, a Server Reflexive Candidate SHOULD be created with the same IP address as one of the host UDP candidates. This is done to facilitate a potential TCP connectivity path, even in the absence of TCP Relayed Candidates for one of the endpoints (5) in regular operation mode.

3.1.4.8.1.4 Generation of Candidate Foundations and Priorities

The candidate foundations MUST be generated as specified in [IETFDRAFT-ICENAT-19] section 4.1.1.3. Priorities for UDP and TCP candidates MUST be computed as specified in [IETFDRAFT-ICENAT-19] section 4.1.2 and [IETFDRAFT-ICETCP-07] section 3.2 respectively.

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 time-out, 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 time-out for the connectivity check MUST be reduced to the value specified in section 3.1.6.2.

During the connectivity checks phase, whenever a connectivity check request or response is sent, an additional connectivity check request or response SHOULD<3> be sent along with it. This additional request or response is identical to the original request or response, except that the **fingerprint** for these additional messages MUST be computed using the following CRC32 lookup table. This is done to interoperate with implementations that used the CRC32 lookup table.

0x00000000	**0x77073096**	**0xee0e612c**	**0x990951ba**	**0x076dc419**	**0x706af48f**	**0xe963a535**	**0x9e6495a3**
0x0edb8832	0x79dcb8a4	0xe0d5e91e	0x97d2d988	0x09b64c2b	0x7eb17cbd	0xe7b82d07	0x90bf1d91
0x1db71064	0x6ab020f2	0xf3b97148	0x84be41de	0x1adad47d	0x6ddde4eb	0xf4d4b551	0x83d385c7
0x136c9856	0x646ba8c0	0xfd62f97a	0x8a65c9ec	0x14015c4f	0x63066cd9	0xfa0f3d63	0x8d080df5
0x3b6e20c8	0x4c69105e	0xd56041e4	0xa2677172	0x3c03e4d1	0x4b04d447	0xd20d85fd	0xa50ab56b
0x35b5a8fa	0x42b2986c	0xdbbbc9d6	0xacbcf940	0x32d86ce3	0x45df5c75	0xdcd60dcf	0xabd13d59
0x26d930ac	0x51de003a	0xc8d75180	0xbfd06116	0x21b4f4b5	0x56b3c423	0xcfba9599	0xb8bda50f
0x2802b89e	0x5f058808	0xc60cd9b2	0xb10be924	0x2f6f7c87	0x58684c11	0xc1611dab	0xb6662d3d
0x76dc4190	0x01db7106	0x98d220bc	0xefd5102a	0x71b18589	0x06b6b51f	0x9fbfe4a5	0xe8b8d433
0x7807c9a2	0x0f00f934	0x9609a88e	0xe10e9818	0x7f6a0dbb	0x086d3d2d	0x91646c97	0xe6635c01

0x00000000	0x77073096	0xee0e612c	0x990951ba	0x076dc419	0x706af48f	0xe963a535	0x9e6495a3
0x6b6b51f4	0x1c6c6162	0x856530d8	0xf262004e	0x6c0695ed	0x1b01a57b	0x8208f4c1	0xf50fc457
0x65b0d9c6	0x12b7e950	0x8bbe8ea	0xfcb9887c	0x62dd1ddf	0x15da2d49	0x8cd37cf3	0xfbd44c65
0x4db26158	0x3ab551ce	0xa3bc0074	0xd4bb30e2	0x4adfa541	0x3dd895d7	0xa4d1c46d	0xd3d6f4fb
0x4369e96a	0x346ed9fc	0xad678846	0xda60b8d0	0x44042d73	0x33031de5	0xaa0a4c5f	0xdd0d7cc9
0x5005713c	0x270241aa	0xbe0b1010	0xc90c2086	0x5768b525	0x206f85b3	0xb966d409	0xce61e49f
0x5edef90e	0x29d9c998	0xb0d09822	0xc7d7a8b4	0x59b33d17	0x2eb40d81	0xb7bd5c3b	0xc0ba6cad
0xedb88320	0x9abfb3b6	0x03b6e20c	0x74b1d29a	0xead54739	0x9dd277af	0x04db2615	0x73dc1683
0xe3630b12	0x94643b84	0x0d6d6a3e	0x7a6a5aa8	0xe40ecf0b	0x9309ff9d	0x0a00ae27	0x7d079eb1
0xf00f9344	0x8708a3d2	0x1e01f268	0x6906c2fe	0xf762575d	0x806567cb	0x196c3671	0x6e6b06e7
0xfed41b76	0x89d32be0	0x10da7a5a	0x67dd4acc	0xf9b9df6f	0x8ebeeff9	0x17b7be43	0x60b08ed5
0xd6d6a3e8	0xa1d1937e	0x38d8c2c4	0x4fdff252	0xd1bb67f1	0xa6bc5767	0x3fb506dd	0x48b2364b
0xd80d2bda	0xaf0a1b4c	0x36034af6	0x41047a60	0xdf60efc3	0xa867df55	0x316e8eef	0x4669be79
0xcb61b38c	0xbc66831a	0x256fd2a0	0x5268e236	0xcc0c7795	0xbb0b4703	0x220216b9	0x5505262f
0xc5ba3bbe	0xb2bd0b28	0x2bb45a92	0x5cb36a04	0xc2d7ffa7	0xb5d0cf31	0x2cd99e8b	0x5bdeae1d
0x9b64c2b0	0xec63f226	0x756aa39c	0x026d930a	0x9c0906a9	0xeb0e363f	0x72076785	0x05005713
0x95bf4a82	0xe2b87a14	0x7bb12bae	0x0cb61b38	0x92d28e9b	0xe5d5be0d	0x7cdcefb7	0x0bdbdf21
0x86d3d2d4	0xf1d4e242	0x68ddb3f8	0x1fda836e	0x81be16cd	0xf6b9265b	0x6fb077e1	0x18b74777
0x88085ae6	0xff0f6a70	0x66063bca	0x11010b5c	0x8f659eff	0xf862ae69	0x616bffd3	0x166ccf45
0xa00ae278	0xd70dd2ee	0x4e048354	0x3903b3c2	0xa7672661	0xd06016f7	0x4969474d	0x3e6e77db
0xaed16a4	0xd9d65a	0x40df0b6	0x37d83bf	0xa9bcae5	0xdebb9e	0x47b2cf7	0x30b5ffe

0x00000000	0x77073096	0xee0e612c	0x990951ba	0x076dc419	0x706af48f	0xe963a535	0x9e6495a3
a	dc	6	0	3	c5	f	9
0xbdbdf21c	0xcabac28a	0x53b39330	0x24b4a3a6	0xbad03605	0xcdd70693	0x54de5729	0x23d967bf
0xb3667a2e	0xc4614ab8	0x5d681b02	0x2a6f2b94	0xb40bbe37	0xc30c8ea1	0x5a05df1b	0x2d02ef8d

Transmission of this additional connectivity check packet SHOULD be stopped upon receiving a connectivity check request or response from the peer endpoint (5) with the **IMPLEMENTATION-VERSION** attribute specified in section 2.2.2.2. The additional messages MUST NOT be sent for ICE keep-alive messages. When a connectivity request or response is received the fingerprint checks MUST follow MUST use the fingerprint mechanism specified in [IETFDRAFT-ICENAT-19] section 7.1.1. If the fingerprint checks fail and if the connectivity check request or response does not have the implementation version attribute, the fingerprint checks SHOULD<4> be tried using CRC32 table shown in this section. If the fingerprint checks succeed with the CRC32 table in this section then the packet SHOULD<5> be considered a valid packet and SHOULD be processed as such.

3.1.4.8.2.1 Formation of Candidate Pairs

Once the offer and answer exchange of the candidates is finished, both endpoints (5) have a set of local and remote candidates. The local candidates and remote candidates are paired together to form the 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 four transport addresses, one for the RTP component for local candidate, one for the RTP component for remote candidate, one for the RTCP component for local candidate, and one for the RTCP component for remote candidate. For a candidate pair, the components of the local candidate MUST be paired with the corresponding components of the remote candidate to form a component pair. For example, the local candidate's RTP component transport address is paired with the remote candidate's RTP component transport address. The formation of TCP candidate pairs MUST follow the specifications in [IETFDRAFT-ICETCP-07] section 4.2. Endpoints (5) implementing this protocol MUST NOT generate more than 40 candidate pairs.

3.1.4.8.2.2 Ordering of Candidate Pairs

The priorities for candidate pairs MUST be computed as specified in [IETFDRAFT-ICENAT-19] section 5.7.2. The candidate pairs MUST be ordered and pruned to form the Check List of candidate pairs, as specified in [IETFDRAFT-ICENAT-19] section 5.7.3.

3.1.4.8.2.3 Candidate Pair States

Each candidate pair state is updated as the connectivity checks progress. The candidate pair states and the transitions between the different states are specified in [IETFDRAFT-ICENAT-19] section 5.7.4.

3.1.4.8.2.4 Forming and Sending Binding Requests for Connectivity Checks

Connectivity checks are performed between the two endpoints (5) by sending peer-to-peer STUN binding request messages, as specified in [IETFDRAFT-ICENAT-19] section 5.8. The STUN binding request message MUST have the **USERNAME** and **MESSAGE-INTEGRITY** attributes and MUST use

the STUN short-term credential mechanism. The **USERNAME** and **MESSAGE-INTEGRITY** attributes MUST be formed as specified in [IETFDRAFT-ICENAT-19] section 7.1.1. Mandating the use of the **MESSAGE-INTEGRITY** attribute in STUN binding request messages serves to mitigate attacks on connectivity described in section 5.1.3. These two attributes MUST support the additional attributes specified in section 2.2 and MUST follow the usage specified in [IETFDRAFT-ICENAT-19] section 7.1.1. The connectivity checks MUST use the fingerprint mechanism specified in [IETFDRAFT-ICENAT-19] section 7.1.1.

The STUN binding request message MUST have the **CANDIDATE-IDENTIFIER** attribute. The value of this attribute MUST be set to the foundation of the local candidate for which the request is being sent if the candidate is not peer-derived. If the local candidate is peer-derived, the value of the **CANDIDATE-IDENTIFIER** MUST be set to the foundation of the peer-derived local candidate's base.

The connectivity checks are sent between component pairs based on the ordering of candidate pairs in the Check List, following the procedures specified in [IETFDRAFT-ICENAT-19] section 5.8. The processing of connectivity checks and the responses are specified in section 3.1.5.

3.1.4.8.2.5 Spacing of Connectivity Checks

To avoid flooding the network, the connectivity checks and their retries SHOULD be spaced as specified in [IETFDRAFT-ICENAT-19] section 5.8.

3.1.4.8.2.6 Termination of Connectivity Checks

The connectivity checks phase MUST be terminated either when the connectivity checks timer is triggered or when the connectivity checks for all candidate pairs are complete. Connectivity checks for a candidate pair MUST be considered complete if the candidate pair is either in "Succeeded" or "Failed" state. At the end of the connectivity checks phase, if there are no candidate pairs in the Valid List on the controlling agent, the call MUST fail. On the controlling agent, the endpoint (5) MUST begin performing Regular Nomination, as specified in [IETFDRAFT-ICENAT-19] section 8.1.1.1, for the candidate pair with the highest priority in the Valid List. If the nomination connectivity checks are successful, the nominated candidate pair MUST be selected for final media flow. If the Regular Nomination connectivity checks fail, the call MUST fail. The controlling agent MUST respond to connectivity checks until it gets the answer to its final offer. The controlled agent MUST continue to respond to connectivity checks until it gets the final offer from the controlling agent.

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 in regular mode can begin sending media after the initial exchange of candidates is finished. Endpoints (5) that follow this protocol SHOULD be prepared to accept media on any of the base transport addresses of the published candidates. 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 has both of its constituent component pairs in "Succeeded" state. After the final exchange of the candidates selected by the connectivity checks phase, media flow MUST be switched to use the best local and remote candidates exchanged. Applications in TCP-only mode MUST wait for connectivity checks to complete if they require data to be delivered reliably.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Processing TURN Messages

The processing of TURN messages, response generation, and error handling is performed as specified in [MS-TURN] section 3.2.5 when communicating with a TURN server.

3.1.5.2 Processing STUN Connectivity Check Messages

This protocol sends peer-to-peer STUN messages between endpoints (5) during the connectivity checks phase to select the candidate pairs for streaming media.

This section specifies the processing of STUN binding request messages by the two endpoints (5).

3.1.5.2.1 Processing the STUN Binding Request

The STUN binding request messages might be received before the remote candidates are received from the peer endpoint (5) in the offer or answer. The endpoint (5) MUST validate the request. If the request is invalid, the endpoint (5) SHOULD send a binding error response for the STUN binding request message, as specified in section 3.1.5.2.2. If the request is valid, the endpoint (5) MUST follow the procedures specified in [IETFDRAFT-ICENAT-19] section 7.2 for processing the STUN binding request.

3.1.5.2.2 Validation of STUN Binding Request

Validation procedures for STUN binding request messages specified in [IETFDRAFT-STUN-02] section 8 differ from the procedures described in this section. Endpoints (5) that follow this protocol 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. ICE keep-alive messages are discarded if they do not have the **USERNAME** attribute. If the **USERNAME** attribute is not valid, the message MUST be discarded. A **USERNAME** attribute is considered valid if it consists of two values separated by a colon and the first value equals the **username** fragment generated by the endpoint (5) in the offer. If the received STUN binding request message does not have the **fingerprint** attribute, the message MUST be discarded. If the STUN binding request message does not have the **MESSAGE INTEGRITY** attribute, the endpoint (5) MUST send a binding error response with error code 401 Unauthorized, as specified in [IETFDRAFT-STUN-02] section 8. If the **MESSAGE INTEGRITY** attribute exists, the endpoint (5) MUST use the STUN short-term credential mechanism using the password that was sent to the peer to compute the message integrity, and verify against the message integrity value in the request. If the message integrity check fails, the endpoint (5) MUST send a binding error response with the error code 431 Integrity Check Failure, as specified in [IETFDRAFT-STUN-02] section 8. Generated binding error responses MUST have a **USERNAME** attribute set to the value of the **USERNAME** attribute received in the STUN binding request message.

3.1.5.2.3 Sending the STUN Binding Response

If the request is valid, the endpoint (5) MUST send a STUN binding response message, as specified in [IETFDRAFT-STUN-02] section 8, with a subset of its attributes. The STUN binding response message MUST implement only the following attributes:

- **XOR-MAPPED-ADDRESS**
- **USERNAME**

- **MESSAGE-INTEGRITY**
- **IMPLEMENTATION-VERSION**

The **XOR-MAPPED-ADDRESS** format MUST be as specified in [IETFDRAFT-STUN-02] section 8.1. The **X-PORT** and **X-ADDRESS** attributes MUST be computed as specified in [IETFDRAFT-STUN-02] section 8.1 for the IP address and port from which the STUN binding request message was received. The **USERNAME** and **MESSAGE-INTEGRITY** attributes MUST be formed as specified in [IETFDRAFT-ICENAT-19] section 7.1.1.

3.1.5.3 STUN Binding Response

This section specifies the way an endpoint (5) processes 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 component pairs and discovery of peer-derived candidates. The processing of STUN binding responses MUST follow the procedures specified in [IETFDRAFT-ICENAT-19] section 7.1.2.

3.1.5.3.1 Validation of the STUN Binding Response

If a 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. If the component pair is in failed state, the STUN binding response message MUST be discarded. If the received STUN binding response message does not have the **fingerprint** attribute, the message MUST be discarded.

The password received from the peer endpoint (5) is 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.

3.1.5.3.2 Processing STUN Binding Response

The STUN binding response MUST be processed following the procedures specified in [IETFDRAFT-ICENAT-19] section 7.1.2.

3.1.5.3.3 STUN Binding Error Response

The error response message MUST be validated in the same way as STUN binding response messages. The validation procedure is specified in section 3.1.5.3.2.

If the component for which the error response is received is already in "Succeeded" state, 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 component pair MUST be set to a "Failed" state.

3.1.6 Timer Events

3.1.6.1 Candidates Gathering-Phase Timer

The candidates gathering phase timer tracks the maximum duration for the candidates gathering phase during which the endpoint (5) gathers the different transport addresses. Default values SHOULD be set to 10 seconds. The firing of the candidates gathering-phase timer signals the end of the candidates gathering phase. The endpoint (5) MUST exchange the gathered local candidates with its peer.

3.1.6.2 Connectivity Phase Timer

The connectivity phase timer tracks the maximum duration for which connectivity checks can be performed for all the candidate pairs. Maximum time-out for this phase MUST be set to 10 seconds. Once a STUN binding request message and response are received from the peer, the timer MUST be reset to 5 seconds. The firing of this timer signals the end of the connectivity checks phase. When this timer fires, the controlling agent MUST pick the best candidate pair selected by the connectivity checks and send it to the controlled agent. If no candidate pair is validated by the connectivity checks when the timer fires on the controlling agent, the call MUST fail. Further connectivity check attempts MUST NOT be made after this timer fires. When this timer fires on the controlled agent, it MUST stop its connectivity checks.

3.1.6.3 ICE keep-alive Timer

The ICE keep-alive messages are sent from the local transport address to the remote transport address in the component pair. ICE keep-alive messages MUST be sent even if the peer endpoint (5) does not implement ICE for the RTP component pair that is associated with the candidate pair that is used for media flow. ICE keep-alive messages MUST be STUN binding request messages, as specified in section 2.2.3. The ICE keep-alive timer MUST have a default value of 19 seconds or less.

During the connectivity checks phase, the ICE keep-alive timer SHOULD fire for validated component pairs and for component pairs whose local candidates are Relayed Candidates, if no connectivity check packets or ICE keep-alive messages have been sent for the component pair for the timer value specified in this section. When the ICE keep-alive timer fires, an ICE keep-alive message SHOULD be sent for the component pair.

In addition to the keep-alive messages during the connectivity checks, for the candidate that is being used for media flow, the ICE keep-alive timer MUST fire when there has been no flow of media or ICE keep-alive messages for the duration specified in this section. When the ICE keep-alive timer fires, an ICE keep-alive message MUST be for the RTP component pair that is associated with the media flow candidate pair. ICE keep-alive messages SHOULD NOT be sent for an RTCP component because the flow of RTCP packets is sufficient to keep the NAT bindings and TURN allocations active.

3.1.6.4 USE-CANDIDATE Checks Timer

The USE-CANDIDATE checks timer tracks the maximum duration for which the nomination checks with the **USE-CANDIDATE** attribute can be performed on the controlling agent. The maximum timeout for this phase SHOULD be 10 Seconds.

3.1.7 Other Local Events

None.

4 Protocol Examples

This protocol example is based on the following sample topology.

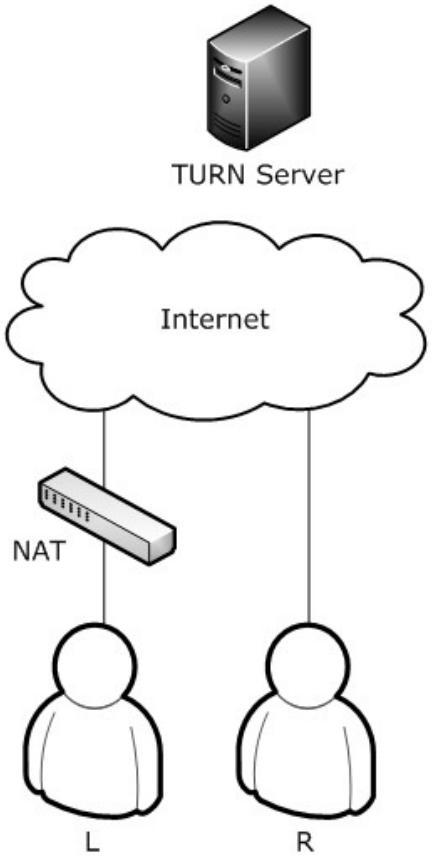


Figure 3: ICE implementations

The figure shows Endpoint L and Endpoint R using ICE. Both agents are full ICE implementations and use Regular Nominations for selecting the candidates to be used for media flow. Endpoint L is behind a NAT in a private address space 192.168.2.1 with the public edge of the NAT at 10.107.0.71 and **agent** R is on the public internet at 10.104.0.68. Both endpoints (5) are configured with the same UDP TURN server that is listening on IP address 10.101.0.57and port 3478.

The transport address follows a similar naming convention to the sample in [IETFDRAFT-ICENAT-19] section 17.

Transport addresses are referred to using the mnemonic names with format entity-type-seqno, where entity refers to the entity whose IP address the transport address is on, and is one of "L", "R", "NAT" or "TURN". The type is either "PUB" for transport addresses that are publicly reachable on the internet or "PRIV" for transport addresses that are not reachable from the internet. The seq-no is a number that is different for transport addresses of the same type on an entity. The TURN server has the transport address TURN-PUB-1 (10.101.0.57and port 3478).

For the call flow:

- "S=" refers to the source transport address.
- "D=" refers to the destination transport address.
- "MA=" refers to the mapped address in the STUN binding response.
- "RA=" refers to the reflexive address.
- "TA=" refers to the relay transport address.

For clarity, the example does not show the TURN **authentication (2)** mechanisms and RTCP component.

The example focuses on the RTP component for establishing a media session between Endpoint L and Endpoint R. Endpoint L initiates the media session and becomes the controlling agent because Endpoint L is a full ICE implementation. Endpoint L gathers its UDP Host Candidate by binding to its local interface and then gathers UDP Relayed Candidates and UDP Server Reflexive Candidates from the configured TURN server. Because no TCP TURN servers are configured, Endpoint L creates a TCP-ACT Server Reflexive Candidate based on the UDP Server Reflexive Candidate. After gathering the candidates, Endpoint L sends the **INVITE** to Endpoint R. A sample INVITE SDP for Endpoint L's topology is as follows:

v=0

o=- 0 0 IN IP4 10.101.0.57

s=session

c=IN IP4 10.101.0.57

b=CT:99980

t=0 0

m=audio 52732 RTP/AVP 114 111 112 115 116 4 8 0 97 13 118 101

a=ice-ufrag:qkEP

a=ice-pwd:ed6f9GuHjLcoCN6sC/Eh7fVl

a=candidate:1 1 UDP 2130706431 192.168.2.1 50005 typ host

a=candidate:2 1 UDP 16648703 10.101.0.57 52732 typ relay raddr 10.107.0.71 rport 50033
a=candidate:3 1 UDP 1694234623 10.107.0.71 50033 typ srflx raddr 192.168.2.1 rport 50033
a=candidate:4 1 TCP-ACT 1684797951 10.107.0.71 50033 typ srflx raddr 192.168.2.1 rport 50033
a=rtpmap:114 x-msrta/16000

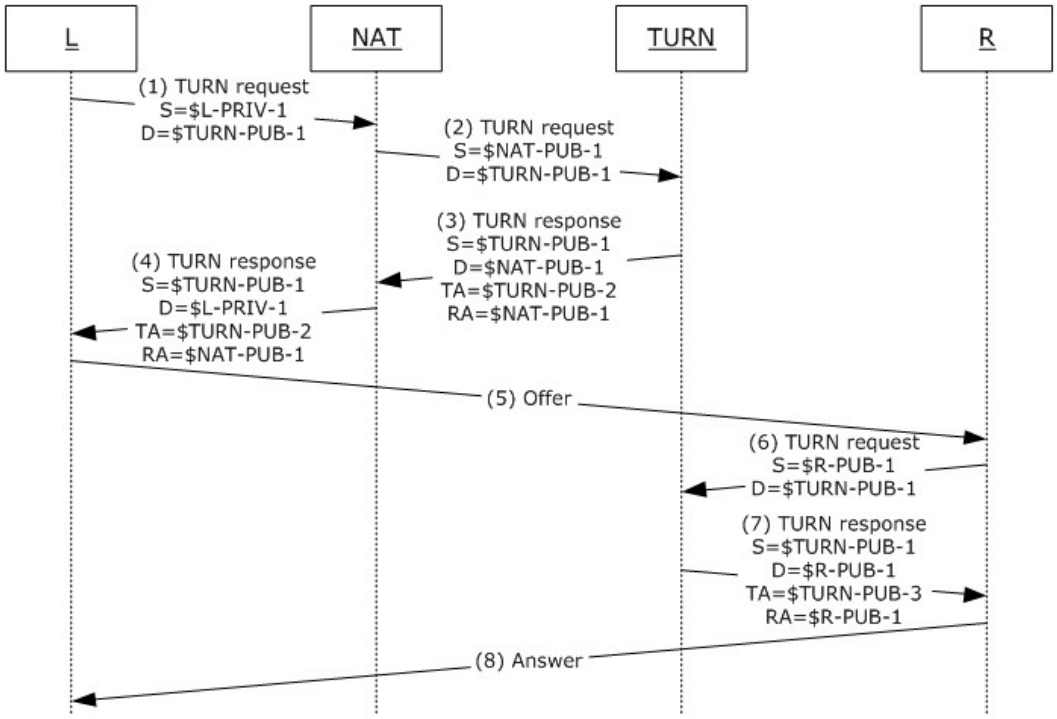


Figure 4: ICE request and response sequence

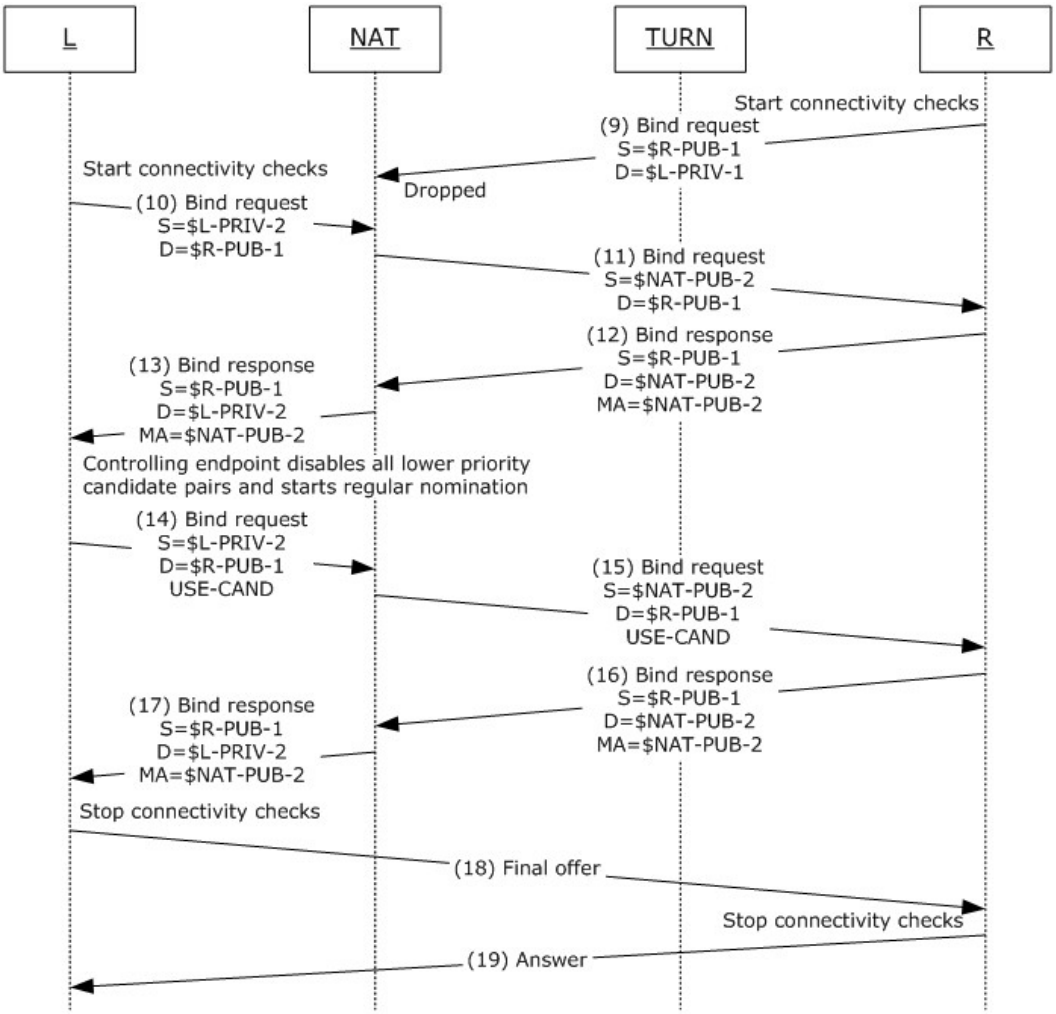


Figure 5: ICE request and response sequence (continued)

Endpoint R, upon receiving the offer, gathers its candidates. It gathers its UDP Host Candidate by binding to its local interface and then gathers UDP Relayed Candidates from the configured TURN server. Endpoint R is not behind a NAT no UDP Server Reflexive Candidates are created. Because no TCP TURN servers are configured, Endpoint R creates a TCP-ACT Server Reflexive Candidate based on the UDP Host Candidate. Endpoint R sends its candidates to Endpoint L in the answer. Endpoint R pairs its local candidates with Endpoint L's remote candidates and starts connectivity checks. A sample answer SDP for Endpoint R's topology is as follows:

v=0

o=- 0 0 IN IP4 10.101.0.57

s=session

c=IN IP4 10.101.0.57

b=CT:99980

t=0 0

m=audio 52714 RTP/AVP 114 111 112 115 116 4 8 0 97 13 118 101

a=ice-ufrag:qkEP

a=ice-pwd:ed6f9GuHjLcoCN6sC/Eh7fVl

a=candidate:1 1 UDP 2130706431 10.104.0.68 50025 typ host

a=candidate:2 1 UDP 16648703 10.101.0.57 52714 typ relay raddr 10.104.0.68 rport 50036
a=candidate:3 1 TCP-ACT 1684797951 10.104.0.68 50025 typ srflx raddr 10.104.0.68 rport 50025
a=rtpmap:114 x-msrta/16000

Endpoint L, upon receiving the answer from Endpoint R, pairs its local candidates with the candidates received in the answer and starts connectivity checks. Both endpoints (5) perform connectivity checks with the highest priority candidate pairs.

The preceding sequence diagram shows that Endpoint R sends a STUN binding request from R-PUB-1 to L-PRIV-2, which does not reach L-PRIV-2 because it is not directly reachable from R-PUB-1. At this point, endpoint (5) L sends a STUN binding request from L-PRIV-2 to R-PUB-1. This request goes through the NAT and Endpoint R eventually receives the packet at R-PUB-1 with the source as NAT-PUB-2. Agent R sends a STUN binding response with the mapped address set to NAT-PUB-2. Endpoint L eventually gets the packet from the NAT and discovers a new peer-derived candidate, because the mapped address is different from the address the STUN binding request sent. The endpoint (5) validates this candidate pair and disables all lower priority candidate pairs. Because this is the highest priority candidate pair, Endpoint L nominates this candidate pair and sends a STUN binding request to R-PUB-1 with the **USE-CANDIDATE** flag set. Endpoint R, upon getting the request with the **USE-CANDIDATE** flag, responds with a STUN binding response. Upon receiving the response, Endpoint L stops its connectivity checks because it has found the candidate pair that has to be used for media flow.

Endpoint L sends the final offer to Endpoint R, with the final local and remote candidate to be used for media flow. A sample final offer is as follows:

v=0

o=- 0 0 IN IP4 10.107.0.71

s=session

c=IN IP4 10.107.0.71

b=CT:99980

t=0 0

m=audio 50005 RTP/SAVP 114 111 112 115 116 4 8 0 97 13 118 101

a=ice-ufrag:32sD

a=ice-pwd:YF9/OwRcN/pXUglBv1c+5QMu

a=candidate:7 1 UDP 1862270719 10.107.0.71 50005 typ prflx raddr 192.168.2.4 rport 50005

a=remote-candidates:1 10.104.0.68 50025

a=rtpmap:114 x-msrta/16000

Endpoint R, upon receiving the final offer, stops its connectivity checks and sends its answer to the final offer.

v=0

o=- 0 0 IN IP4 10.104.0.68s=sessionc=IN IP4 10.104.0.68

b=CT:99980

t=0 0

m=audio 50025 RTP/SAVP 114 111 112 115 116 4 8 0 97 13 118 101

a=ice-ufrag:32sD

a=ice-pwd:YF9/OwRcN/pXUglBv1c+5QMu

a=candidate:7 1 UDP 1862270719 10.104.0.68 50025 typ host

a=remote-candidates:1 10.107.0.71 50005

a=rtpmap:114 x-msrta/16000

With the receipt of the final answer, the connectivity checks phase ends and both ends stream media using the final candidates selected by the connectivity checks.

5 Security

5.1 Security Considerations for Implementers

This protocol has similar security concerns as those described in [IETFDRAFT-ICENAT-19] section 18. Additional considerations and mitigations pertaining to this protocol are listed in this section.

5.1.1 Attacks on Address Gathering

The security considerations for using the protocol described in [MS-TURN] for gathering STUN candidates and TURN candidates are addressed in [MS-TURN] section 5.

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-based connectivity checks, either to maliciously gain control of the call and related media to a different endpoint (5) or to cause failure of the connectivity checks. The malicious user can potentially inject connectivity check packets to fool an endpoint (5) into considering a valid candidate pair invalid or vice versa. Alternatively, the malicious user can cause the endpoints (5) 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 (DOS) attack as the default candidate in its offer and send the offer to multiple endpoints (5). This action can potentially result in each endpoint (5) that received the offer attempting to send media to the target of the DOS 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

This malicious activity is similar to the voice amplification attack. Instead of media flow, the STUN connectivity checks are directed to the target of the DOS attack. The malicious user proceeds by generating an offer with a large number of candidates for the DOS target. The peer endpoint (5), after receiving the offers, performs connectivity checks with all the candidates specified in 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 and response to 20 candidates and 40 candidate pairs. This protocol mitigates the similar attack of generating multiple provisional answers to an offer by limiting the number of provisional answers supported. 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 released service packs:

- Microsoft® Office Communications Server 2007 R2
- Microsoft® Office Communicator 2007 R2
- Microsoft® Lync™ Server 2010
- Microsoft® Lync™ 2010

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 2.2.2.2: Office Communicator 2007 R2, Office Communications Server 2007 R2: The **Implementation Version** attribute is not supported.

<2> Section 2.2.2.2: Office Communicator 2007 R2, Office Communications Server 2007 R2: MUST set the **IMPLEMENTATION-VERSION** attribute as 0x00000001. Lync 2010, Lync Server 2010: MUST set the **IMPLEMENTATION-VERSION** attribute as 0x00000002.

<3> Section 3.1.4.8.2: Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<4> Section 3.1.4.8.2: Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<5> Section 3.1.4.8.2: Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

7 Change Tracking

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

8 Index

T

V