

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 14, 2014

J. Goldberg  
Cisco  
M. Westerlund  
Ericsson  
T. Zeng  
Nextwave Wireless, Inc.  
February 10, 2014

**A Network Address Translator (NAT) Traversal Mechanism for Media  
Controlled by Real-Time Streaming Protocol (RTSP)  
draft-ietf-mmusic-rtsp-nat-20**

**Abstract**

This document defines a solution for Network Address Translation (NAT) traversal for datagram based media streams set up and controlled with Real-time Streaming Protocol version 2 (RTSP 2.0). It uses Interactive Connectivity Establishment (ICE) adapted to use RTSP as a signalling channel, defining the necessary RTSP extensions and procedures.

**Status of This Memo**

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 14, 2014.

**Copyright Notice**

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Definitions . . . . .	<a href="#">4</a>
<a href="#">3.</a>	Solution Overview . . . . .	<a href="#">4</a>
<a href="#">4.</a>	RTSP Extensions . . . . .	<a href="#">6</a>
<a href="#">4.1.</a>	ICE Transport Lower Layer . . . . .	<a href="#">6</a>
<a href="#">4.2.</a>	ICE Candidate Transport Header Parameter . . . . .	<a href="#">7</a>
<a href="#">4.3.</a>	ICE Password and Username Transport Header Parameters . . . . .	<a href="#">10</a>
<a href="#">4.4.</a>	ICE Feature Tag . . . . .	<a href="#">11</a>
<a href="#">4.5.</a>	Status Codes . . . . .	<a href="#">11</a>
<a href="#">4.5.1.</a>	150 ICE Connectivity Checks in Progress . . . . .	<a href="#">12</a>
<a href="#">4.5.2.</a>	480 ICE Processing Failed . . . . .	<a href="#">12</a>
<a href="#">4.6.</a>	New Reason for PLAY_NOTIFY . . . . .	<a href="#">12</a>
<a href="#">4.7.</a>	Server Side SDP Attribute for ICE Support . . . . .	<a href="#">12</a>
<a href="#">5.</a>	ICE-RTSP . . . . .	<a href="#">12</a>
<a href="#">5.1.</a>	ICE Features Not Required . . . . .	<a href="#">13</a>
<a href="#">5.1.1.</a>	ICE-Lite . . . . .	<a href="#">13</a>
<a href="#">5.1.2.</a>	ICE-Mismatch . . . . .	<a href="#">13</a>
<a href="#">5.1.3.</a>	ICE Remote Candidate Transport Header Parameter . . . . .	<a href="#">13</a>
<a href="#">5.2.</a>	High-Reachability Configuration . . . . .	<a href="#">13</a>
<a href="#">6.</a>	Detailed Solution . . . . .	<a href="#">14</a>
<a href="#">6.1.</a>	Session description and RTSP DESCRIBE (optional) . . . . .	<a href="#">14</a>
<a href="#">6.2.</a>	Setting up the Media Streams . . . . .	<a href="#">15</a>
<a href="#">6.3.</a>	RTSP SETUP Request . . . . .	<a href="#">15</a>
<a href="#">6.4.</a>	Gathering Candidates . . . . .	<a href="#">16</a>
<a href="#">6.5.</a>	RTSP Server Response . . . . .	<a href="#">17</a>
<a href="#">6.6.</a>	Server to Client ICE Connectivity Checks . . . . .	<a href="#">17</a>
<a href="#">6.7.</a>	Client to Server ICE Connectivity Check . . . . .	<a href="#">18</a>
<a href="#">6.8.</a>	Client Connectivity Checks Complete . . . . .	<a href="#">19</a>
<a href="#">6.9.</a>	Server Connectivity Checks Complete . . . . .	<a href="#">19</a>
<a href="#">6.10.</a>	Freeing Candidates . . . . .	<a href="#">20</a>
<a href="#">6.11.</a>	Steady State . . . . .	<a href="#">20</a>
<a href="#">6.12.</a>	Re-SETUP . . . . .	<a href="#">20</a>
<a href="#">6.13.</a>	Server Side Changes After Steady State . . . . .	<a href="#">21</a>
<a href="#">7.</a>	ICE and Proxies . . . . .	<a href="#">23</a>
<a href="#">7.1.</a>	Media Handling Proxies . . . . .	<a href="#">23</a>
<a href="#">7.2.</a>	Signalling Only Proxies . . . . .	<a href="#">24</a>
<a href="#">7.3.</a>	Non-supporting Proxies . . . . .	<a href="#">24</a>
<a href="#">8.</a>	RTP and RTCP Multiplexing . . . . .	<a href="#">25</a>
<a href="#">9.</a>	Fallback and Using Partial ICE functionality to improve NAT/Firewall traversal . . . . .	<a href="#">26</a>



<a href="#">10.</a>	IANA Considerations . . . . .	<a href="#">27</a>
<a href="#">10.1.</a>	RTSP Feature Tags . . . . .	<a href="#">27</a>
<a href="#">10.2.</a>	Transport Protocol Specifications . . . . .	<a href="#">28</a>
<a href="#">10.3.</a>	RTSP Transport Parameters . . . . .	<a href="#">28</a>
<a href="#">10.4.</a>	RTSP Status Codes . . . . .	<a href="#">28</a>
<a href="#">10.5.</a>	Notify-Reason value . . . . .	<a href="#">29</a>
<a href="#">10.6.</a>	SDP Attribute . . . . .	<a href="#">29</a>
<a href="#">11.</a>	Security Considerations . . . . .	<a href="#">29</a>
<a href="#">11.1.</a>	ICE and RTSP . . . . .	<a href="#">29</a>
<a href="#">12.</a>	Acknowledgements . . . . .	<a href="#">30</a>
<a href="#">13.</a>	References . . . . .	<a href="#">30</a>
<a href="#">13.1.</a>	Normative References . . . . .	<a href="#">30</a>
<a href="#">13.2.</a>	Informative References . . . . .	<a href="#">31</a>
	Authors' Addresses . . . . .	<a href="#">32</a>

## [1.](#) Introduction

Real-time Streaming Protocol (RTSP) [[RFC2326](#)] and RTSP 2.0 [[I-D.ietf-mmusic-rfc2326bis](#)] are protocols used to setup and control one or more media streams delivering media to receivers. It is RTSP's functionality of setting up media streams that causes serious issues with Network Address Translators (NAT) [[RFC3022](#)] unless extra provisions are taken by the protocol. There is thus a need for a NAT traversal mechanism for the media setup using RTSP.

RTSP 1.0 [[RFC2326](#)] has suffered from the lack of a standardized NAT traversal mechanism for a long time, however due to quality of the RTSP 1.0 specification, the work was difficult to specify in an interoperable fashion. This document is therefore built on the specification of RTSP 2.0 [[I-D.ietf-mmusic-rfc2326bis](#)]. RTSP 2.0 is similar to RTSP 1.0 in many respects but significantly for this work, it contains a well defined extension mechanism that allows a NAT traversal extension to be defined that is backwards compatible with RTSP 2.0 peers not supporting the extension. This extension mechanism was not possible in RTSP 1.0 as it would break RTSP 1.0 syntax and cause compatibility issues.

There have been a number of suggested ways of resolving the NAT-traversal of media for RTSP of which a large number are already used in implementations. The evaluation of these NAT traversal solutions in [[I-D.ietf-mmusic-rtsp-nat-evaluation](#)] has shown that there are many issues to consider, so after extensive evaluation, a mechanism based on Interactive Connectivity Establishment (ICE) [[RFC5245](#)] was selected. There were mainly two reasons: Firstly, the mechanism supports RTSP servers behind NATs and secondly, the mechanism mitigates the security threat of using RTSP servers as Distributed Denial of Service (DDoS) attack tools.



This document specifies an ICE based solution that is optimized for media delivery from server to client. If future extensions are specified for other delivery modes than "PLAY", then the optimizations in regards to when PLAY request are sent needs to be reconsidered.

The NAT problem for RTSP signalling traffic is a less prevalent problem than the NAT problem for RTSP media streams. Consequently, the former is left for future study.

The ICE usage defined in this specification is called ICE-RTSP and does not match the full ICE for SIP/SDP or ICE-Lite as defined in the ICE specification [[RFC5245](#)]. ICE-RTSP is tailored to the needs of RTSP and is slightly simpler than ICE-Full for both clients and servers.

## **2. Definitions**

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].

## **3. Solution Overview**

This overview assumes that the reader has some familiarity with how ICE [[RFC5245](#)] in the context of "SIP: Session Initiation Protocol" [[RFC3261](#)] and "An Offer/Answer Model with the Session Description Protocol (SDP)" [[RFC3264](#)] works, as it primarily points out how the different ICE steps are accomplished in RTSP.

1. The RTSP server should indicate it has support for ICE via a new SDP [[RFC4566](#)] attribute ("a=rtsp-ice-d-m") in, for example, the SDP returned in the RTSP DESCRIBE message. This allows RTSP clients to only perform the new ICE exchanges with servers that support ICE. If RTSP DESCRIBE is used, the normal capability determination mechanism should also be used, i.e., "Supported" header with a new ICE feature tag. Note: Both mechanisms should be supported, as there are various use cases where only one of them is used.
2. The RTSP client reviews the session description returned, for example by an RTSP DESCRIBE message, to determine what media streams need to be setup. For each of these media streams where the transport protocol supports Session Traversal Utilities for (NAT) (STUN) [[RFC5389](#)] based connectivity checks, the client gathers candidate addresses. See [section 4.1.1](#) in ICE [[RFC5245](#)]. The client then runs a STUN server on each of the local candidates transport addresses it has gathered.



3. The RTSP client sends SETUP requests containing a transport specification with a lower layer indicating ICE and a new RTSP Transport header parameter "candidates" listing the ICE candidates for each media stream.
4. After receiving the list of candidates from a client, the RTSP server gathers its own candidates. If the server is not behind a NAT, then a single candidate per address family (e.g. IPv4 and IPv6), media stream and media component tuple can be included to reduce the number of combinations and speed up the completion.
5. The server sets up the media and if successful responds to the SETUP request with a 200 OK response. In that response the server selects the transport specification using ICE and includes its candidates in the candidates parameter.
6. The server starts the connectivity checks following the procedures described in [Section 5.7](#) and 5.8 of ICE [[RFC5245](#)]. If the server is not behind a NAT and uses a public IP address with a single candidate per (media stream, component, address family) tuple, then the server may be configured to not initiate connectivity checks.
7. The client receives the SETUP response and learns the candidate addresses to use for the connectivity checks, and then initiates its connectivity check, following the procedures in [Section 6](#) of ICE [[RFC5245](#)].
8. When a connectivity check from the client reaches the server it will result in a triggered check from the server. This is why servers not behind a NAT can wait until this triggered check to send out any checks for itself, so saving resources and mitigating the DDoS potential from server initiated connectivity checks.
9. When the client has concluded its connectivity checks, including nominating candidates, and has correspondingly received the server connectivity checks on the nominated candidates for all mandatory components of all media streams, it can issue a PLAY request. If the connectivity checks have not concluded successfully, then the client may send a new SETUP request if it has any new information or believes the server may be able to do more that can result in successful checks.
10. When the RTSP server receives a PLAY request, it checks to see that the connectivity checks have concluded successfully, and only then can it play the stream. If there is a problem with the checks then the server sends either a 150 (ICE connectivity





checks in progress) response to show that it is still working on the connectivity checks, or a 480 (ICE Processing Failed) response to indicate a failure of the checks. If the checks are successful, then the server sends a 200 OK response and starts delivering media.

The client and server may release unused candidates when the ICE processing has concluded and a single candidate per component has been nominated and a PLAY response has been received (Client) or sent (Server).

The client needs to continue to use STUN as a keep-alive mechanism for the used candidate pairs to keep their NAT bindings current. RTSP Servers behind NATs will also need to send keep-alive messages when not sending media. This is important since RTSP media sessions often contain only media traffic from the server to the client so the bindings in the NAT need to be refreshed by client to server traffic provided by the STUN keep-alive.

#### **4. RTSP Extensions**

This section defines the necessary RTSP extensions for performing ICE with RTSP. Note that these extensions are based on the SDP attributes in the ICE specification unless expressly indicated otherwise.

##### **4.1. ICE Transport Lower Layer**

A new lower layer "D-ICE" for transport specifications is defined. This lower layer is datagram clean except that the protocol used must be possible to demultiplex from STUN messages (see STUN [[RFC5389](#)]). With datagram clean we mean that it must be capable of describing the length of the datagram, transport that datagram (as a binary chunk of data) and provide it at the receiving side as one single item. This lower layer can be any transport type defined for ICE which does provide datagram transport capabilities. UDP based transport candidates are defined in ICE [[RFC5245](#)] and MUST be supported. It is OPTIONAL to also support TCP based candidates as defined by "TCP Candidates with Interactive Connectivity Establishment (ICE)" [[RFC6544](#)]. The TCP based candidate fulfills the requirements on providing datagram transport and can thus be used in combination with RTP. Additional transport types for candidates may be defined in the future.

This lower layer uses ICE to determine which of the different candidates shall be used and then, when the ICE processing has concluded, uses the selected candidate to transport the datagrams over this transport.



This lower layer transport can be combined with all upper layer media transport protocols that are possible to demultiplex with STUN and which use datagrams. This specification defines the following combinations:

- o RTP/AVP/D-ICE
- o RTP/AVPF/D-ICE
- o RTP/SAVP/D-ICE
- o RTP/SAVPF/D-ICE

This list can be extended with more transport specifications after having performed the evaluation that they are compatible with D-ICE as lower layer. The registration is required to follow the registry rules for the Transport Protocol Identifier (See Section 22.13.1 of [[I-D.ietf-mmusic-rfc2326bis](#)]).

The lower-layer "D-ICE" has the following rules for the inclusion of the RTSP transport header ([Section 18.54](#) of RTSP 2.0 [[I-D.ietf-mmusic-rfc2326bis](#)]) parameters:

unicast: ICE only supports unicast operations, thus it is REQUIRED that one include the unicast indicator parameter, (see [section 18.54](#) in RTSP 2.0 [[I-D.ietf-mmusic-rfc2326bis](#)]).

candidates: The "candidates" parameter SHALL be included as this specifies at least one candidate to try to establish a working transport path with.

dest\_addr: This parameter MUST NOT be included since "candidates" is used instead to provide the necessary address information.

ICE-Password: This parameter SHALL be included (See [Section 4.2](#)).

ICE-ufrag: This parameter SHALL be included (See [Section 4.2](#)).

#### **[4.2](#). ICE Candidate Transport Header Parameter**

This section defines a new RTSP transport parameter for carrying ICE candidates related to the transport specification they appear within, which may then be validated with an end-to-end connectivity check using STUN [[RFC5389](#)]. Transport parameters may only occur once in each transport specification. For transport specifications using "D-ICE" as lower layer, this parameter MUST be present. The parameter can contain one or more ICE candidates. In the SETUP response there is only a single transport specification, and if that



uses the "D-ICE" lower layer this parameter MUST be present and include the server side candidates.

The ABNF [[RFC5234](#)] for these transport header parameters are:

```
trns-parameter = <Defined in Section 20.2.3 of
                  \[I-D.ietf-mmusic-rfc2326bis\]>
trns-parameter =/ SEMI ice-trn-par
ice-trn-par     = "candidates" EQUAL DQ SWS ice-candidate
                  *(SEMI ice-candidate) SWS DQ
ice-candidate   = foundation SP
                  component-id SP
                  transport SP
                  priority SP
                  connection-address SP
                  port SP
                  cand-type
                  [SP rel-addr]
                  [SP rel-port]
                  [SP tcp-type-ext] ; Mandatory if transport = TCP
                  *(SP extension-att-name SP extension-att-value)

foundation      = <See section 15.1 of \[RFC5245\]>
component-id    = <See section 15.1 of \[RFC5245\]>
transport       = <See section 15.1 of \[RFC5245\]>
priority        = <See section 15.1 of \[RFC5245\]>
cand-type       = <See section 15.1 of \[RFC5245\]>
rel-addr        = <See section 15.1 of \[RFC5245\]>
rel-port        = <See section 15.1 of \[RFC5245\]>
tcp-type-ext    = <See section 4.5 of \[RFC6544\]>
extension-att-name = <See section 15.1 of \[RFC5245\]>
extension-att-value = <See section 15.1 of \[RFC5245\]>
connection-address = <See \[RFC4566\]>
port            = <See \[RFC4566\]>
EQUAL           = <Defined in \[I-D.ietf-mmusic-rfc2326bis\]>
DQ              = <Defined in \[I-D.ietf-mmusic-rfc2326bis\]>
SWS             = <Defined in \[I-D.ietf-mmusic-rfc2326bis\]>
SEMI            = <Defined in \[I-D.ietf-mmusic-rfc2326bis\]>
SP              = <Defined in \[I-D.ietf-mmusic-rfc2326bis\]>
```

<connection-address>: is the unicast IP address of the candidate, allowing for IPv4 addresses, IPv6 addresses and Fully qualified domain names (FQDN), taken from SDP [[RFC4566](#)]. Note, the syntax allows multicast addresses, but they SHALL NOT be used in this context. The connection address SHOULD use the same format (explicit IP or FQDN) as in the dest\_addr parameter used to express fallbacks. An IP address SHOULD be used, but an FQDN MAY be used in place of an IP address. In that case, when receiving a



SETUP request or response containing an FQDN in a candidate parameter, the FQDN is looked up in the DNS first using an AAAA record (assuming the agent supports IPv6), and if no result is found or the agent only supports IPv4, using an A record. If the DNS query returns more than one IP address, one is chosen, and then used for the remainder of ICE processing which in RTSP is subsequent RTSP SETUPS for the same RTSP session.

<port>: is the port of the candidate; the syntax is defined by SDP [[RFC4566](#)].

<transport>: indicates the transport protocol for the candidate. The ICE specification defines UDP. "TCP Candidates with Interactive Connectivity Establishment (ICE)" [[RFC6544](#)] defines how TCP is used as candidates. Additional extensibility is provided to allow for future transport protocols to be used with ICE, such as the Datagram Congestion Control Protocol (DCCP) [[RFC4340](#)].

<foundation>: is an identifier that is equivalent for two candidates that are of the same type, share the same base, and come from the same STUN server, and is composed of one to thirty two <ice-char>. The foundation is used to optimize ICE performance in the Frozen algorithm (as described in [[RFC5245](#)]).

<component-id>: identifies the specific component of the media stream for which this is a candidate and is a positive integer belonging to the range 1-256. It MUST start at 1 and MUST increment by 1 for each component of a particular media stream. For media streams based on RTP, candidates for the actual RTP media MUST have a component ID of 1, and candidates for RTCP MUST have a component ID of 2 unless RTP and RTCP Multiplexing ([Section 8](#)) is used when the second component is omitted and RTP and RTCP are both transported over the first component. Other types of media streams which require multiple components MUST develop specifications which define the mapping of components to component IDs. See [Section 14 in \[RFC5245\]](#) for additional discussion on extending ICE to new media streams.

<priority>: is a positive integer in the range 1 to ( $2^{31} - 1$ ).

<cand-type>: encodes the type of candidate. The ICE specification defines the values "host", "srflx", "prflx" and "relay" for host, server reflexive, peer reflexive and relayed candidates, respectively. The set of candidate types is extensible for the future.





<rel-addr> and <rel-port>: convey transport addresses related to the candidate, useful for diagnostics and other purposes. <rel-addr> and <rel-port> MUST be present for server reflexive, peer reflexive and relayed candidates. If a candidate is server or peer reflexive, <rel-addr> and <rel-port> are equal to the base for that server or peer reflexive candidate. If the candidate is relayed, <rel-addr> and <rel-port> are equal to the mapped address in the Allocate Response that provided the client with that relayed candidate (see [Appendix B.3](#) of ICE [RFC5245] for a discussion of its purpose). If the candidate is a host candidate <rel-addr> and <rel-port> MUST be omitted.

<tcp-type-ext>: conveys the candidates connection type (active, passive, or S-O) for TCP based candidates. This MUST be included for candidates that have <transport> set to TCP and MUST NOT be included for other transport types, including UDP, unless explicitly specified for that transport protocol.

<extension-att-name> and <extension-att-value>: These are prototypes for future extensions of the candidate line. The ABNF for these allow any 8-bit value except NUL, CR, or LF. However, the extensions will occur within a structured line that uses the DQ, SEMI, SWS and SP ABNF constructs as delimiters, thus those delimiter characters MUST be escaped if they would occur within an extension-att-name or extension-att-value. The escape mechanism that MUST be used is the Percent-Encoding defined in [Section 2.1 of \[RFC3986\]](#). This mechanism is selected as it anyway needs to be supported in an RTSP implementation to deal with URIs. The byte values (in hex) that MUST be escaped are the following ones: 0x09, 0x20, 0x22, 0x25, 0x3B.

#### **4.3. ICE Password and Username Transport Header Parameters**

The ICE password and username for each agent needs to be transported using RTSP. For that purpose new Transport header parameters are defined (see section 18.54 of [[I-D.ietf-mmusic-rfc2326bis](#)]).

There MUST be an "ICE-Password" and "ICE-ufrag" parameter for each media stream. If two SETUP requests in the same RTSP session have identical ICE-ufrag values, they MUST have identical ICE-Password values.

The ICE-ufrag and ICE-Password parameter values MUST be chosen randomly at the beginning of a session. The ICE-ufrag value MUST contain at least 24 bits of randomness, and the ICE-Password value MUST contain at least 128 bits of randomness. This means that the ICE-ufrag value will be at least 4 characters long, and the ICE-Password value at least 22 characters long, since the grammar for



these attributes allows for 6 bits of randomness per character. The values MAY be longer than 4 and 22 characters respectively, of course, up to 256 characters. The upper limit allows for buffer sizing in implementations. Its large upper limit allows for increased amounts of randomness to be added over time.

The ABNF [[RFC5234](#)] for these parameters are:

```
trns-parameter    =/ SEMI ice-password-par
trns-parameter    =/ SEMI ice-ufrag-par
ice-password-par  = "ICE-Password" EQUAL DQ password DQ
ice-ufrag-par     = "ICE-ufrag" EQUAL DQ ufrag DQ
password          = <Defined in [RFC5245], Section 15.4>
ufrag             = <Defined in [RFC5245], Section 15.4>
EQUAL             = <Defined in [I-D.ietf-mmusic-rfc2326bis]>
SEMI              = <Defined in [I-D.ietf-mmusic-rfc2326bis]>
DQ                = <Defined in [I-D.ietf-mmusic-rfc2326bis]>
```

#### 4.4. ICE Feature Tag

A feature tag is defined for use in the RTSP capabilities mechanism for ICE support of media transport using datagrams: "setup.ice-d-m". This feature tag indicates that one supports all the mandatory functions of this specification. It is applicable to all types of RTSP agents; clients, servers and proxies.

The RTSP client SHOULD send the feature tag "setup.ice-d-m" in the "Supported" header in all SETUP requests that contain the "D-ICE" lower layer transport.

#### 4.5. Status Codes

ICE needs two new RTSP response codes to indicate progress and errors.

Code	Reason	Method
150	Server still working on ICE connectivity checks	PLAY
480	ICE Connectivity check failure	PLAY, SETUP

Table 1: New Status codes and their usage with RTSP methods



#### **4.5.1. 150 ICE Connectivity Checks in Progress**

The 150 response code indicates that ICE connectivity checks are still in progress and haven't concluded. This response SHALL be sent within 200 milliseconds of receiving a PLAY request that currently can't be fulfilled because ICE connectivity checks are still running. Subsequently, every 3 seconds after the previous one was sent, a 150 reply SHALL be sent until the ICE connectivity checks conclude either successfully or in failure, and a final response for the request can be provided.

#### **4.5.2. 480 ICE Processing Failed**

The 480 client error response code is used in cases when the request can't be fulfilled due to a failure in the ICE processing, such as all the connectivity checks have timed out. This error message can appear either in response to a SETUP request to indicate that no candidate pair can be constructed, or in response to a PLAY request to indicate that the server's connectivity checks resulted in failure.

#### **4.6. New Reason for PLAY\_NOTIFY**

A new value used in the PLAY\_NOTIFY methods Notify-Reason header is defined: "ice-restart". This reason indicates that an ICE restart needs to happen on the identified resource and session.

Notify-Reas-val =/ "ice-restart"

#### **4.7. Server Side SDP Attribute for ICE Support**

If the server supports the media NAT traversal for RTSP controlled sessions as described in this RFC, then the server SHOULD include the "a=rtsp-ice-d-m" SDP attribute in any SDP (if used) describing content served by the server. This is a session-level only attribute, see [[RFC4566](#)].

The ABNF [[RFC5234](#)] for the "rtsp-ice-d-m" attribute is:

rtsp-ice-d-m-attr = "a=" "rtsp-ice-d-m"

### **5. ICE-RTSP**

This section discusses differences between the regular ICE usage defined in [[RFC5245](#)] and ICE-RTSP. The reasons for the differences relate to the clearer client/server roles that RTSP provides and how the RTSP Session establishment signalling occurs within RTSP compared to SIP/SDP Offer/Answer.



## **5.1. ICE Features Not Required**

A number of ICE signalling features are not needed with RTSP and are discussed below.

### **5.1.1. ICE-Lite**

The ICE-Lite attribute SHALL NOT be used in the context of RTSP. The ICE specification describes two implementations of ICE: Full and Lite, where hosts that are not behind a NAT are allowed to implement only Lite. For RTSP, the Lite implementation is insufficient because it does not cause the media server to send a connectivity check, which is used to protect against making the RTSP server a denial of service tool.

### **5.1.2. ICE-Mismatch**

The ice-mismatch parameter indicates that the offer arrived with a default destination for a media component that didn't have a corresponding candidate attribute. This is not needed for RTSP as the ICE based lower layer transport specification either is supported or another alternative transport is used. This is always explicitly indicated in the SETUP request and response.

### **5.1.3. ICE Remote Candidate Transport Header Parameter**

The Remote candidate attribute is not needed for RTSP for the following reasons. Each SETUP results in an independent ICE processing chain which either fails or results in nominating a single candidate pair to usage. If a new SETUP request for the same media is sent, this needs to use a new username fragment and password to avoid any race conditions or uncertainty about which round of processing the STUN requests relate to.

## **5.2. High-Reachability Configuration**

ICE-RTSP contains a high-reachability configuration when the RTSP servers are not behind NATs. Please note that "not behind NATs" may apply in some special cases also for RTSP servers behind NATs given that they are in an address space that has reachability for all the RTSP clients intended to be able to reach the server. The high-reachability configuration is similar to ICE-Lite as it allows for some reduction in the server's burden. However, due to the need to still verify that the client is actually present and wants to receive the media stream, the server must also initiate binding requests and await binding responses. The reduction for the high-reachability configuration of ICE-RTSP is that they don't need to initiate their own checks, and instead rely on triggered checks for verification.





This also removes a denial of service threat where a RTSP SETUP request will trigger large amount of STUN connectivity checks towards provided candidate addresses.

## **6. Detailed Solution**

This section describes in detail how the interaction and flow of ICE works with RTSP messages.

### **6.1. Session description and RTSP DESCRIBE (optional)**

The RTSP server is RECOMMENDED to indicate it has support for ICE by sending the "a=rtsp-ice-d-m" SDP attribute in the response to the RTSP DESCRIBE message if SDP is used. This allows RTSP clients to only send the new ICE exchanges with servers that support ICE thereby limiting the overhead on current non-ICE supporting RTSP servers. When not using RTSP DESCRIBE it is still RECOMMENDED to use the SDP attribute for the session description.

A client can also use the DESCRIBE request to determine explicitly if both server and any proxies support ICE. The client includes the "Supported" header with its supported feature tags, including "setup.ice-d-m". Any proxy upon seeing the "Supported" header will include the "Proxy-Supported" header with the feature tags it supports. The server will echo back the "Proxy-Supported" header and its own version of the Supported header so enabling a client to determine if all involved parties support ICE or not. Note that even if a proxy is present in the chain that doesn't indicate support for ICE, it may still work (see [Section 7](#)).



For example:

```
C->S: DESCRIBE rtsp://server.example.com/fizzle/foo RTSP/2.0
      CSeq: 312
      User-Agent: PhonyClient 1.2
      Accept: application/sdp, application/example
      Supported: setup.ice-d-m, setup.rtp.rtcp.mux
```

```
S->C: RTSP/2.0 200 OK
      CSeq: 312
      Date: 23 Jan 1997 15:35:06 GMT
      Server: PhonyServer 1.1
      Content-Type: application/sdp
      Content-Length: 367
      Supported: setup.ice-d-m, setup.rtp.rtcp.mux
```

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 192.0.2.46
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/lectures/sdp.ps
e=seminar@example.com (Seminar Management)
t=2873397496 2873404696
a=recvonly
a=rtsp-ice-d-m
a=control: *
m=audio 3456 RTP/AVP 0
a=control: /audio
m=video 2232 RTP/AVP 31
a=control: /video
```

## 6.2. Setting up the Media Streams

The RTSP client reviews the session description returned, for example by an RTSP DESCRIBE message, to determine what media resources need to be setup. For each of these media streams where the transport protocol supports ICE connectivity checks, the client SHALL gather candidate addresses for UDP transport as described in [section 4.1.1](#) in ICE [[RFC5245](#)] according to standard ICE rather than the ICE-Lite implementation and according to [section 5](#) of ICE TCP [[RFC6544](#)] for TCP based candidates.

## 6.3. RTSP SETUP Request

The RTSP client will then send at least one SETUP request per media stream to establish the media streams required for the desired session. For each media stream where it desires to use ICE it MUST include a transport specification with "D-ICE" as the lower layer, and each media stream SHALL have its own unique combination of ICE



candidates and ICE-ufrag. This transport specification SHOULD be placed first in the list to give it highest priority. It is RECOMMENDED that additional transport specifications are provided as a fallback in case of non-ICE supporting proxies. The RTSP client will be initiating and thus the controlling party in the ICE processing. For example (Note that some lines are broken in contradiction with the defined syntax due to space restrictions in the documenting format):

```
C->S: SETUP rtsp://server.example.com/fizzle/foo/audio RTSP/2.0
      CSeq: 313
      Transport: RTP/AVP/D-ICE; unicast; ICE-ufrag=8hhY;
                ICE-Password=asd88fgpdd777uzjYhagZg; candidates="
                1 1 UDP 2130706431 10.0.1.17 8998 typ host;
                2 1 UDP 1694498815 192.0.2.3 45664 typ srflx
                raddr 10.0.1.17 rport 8998"; RTCP-mux,
                RTP/AVP/UDP; unicast; dest_addr=":6970"/":6971",
                RTP/AVP/TCP;unicast;interleaved=0-1
      Accept-Ranges: NPT, UTC
      User-Agent: PhonyClient/1.2
      Supported: setup.ice-d-m, setup.rtp.rtcp.mux
```

#### 6.4. Gathering Candidates

Upon receiving a SETUP request the server can determine what media resource should be delivered and which transport alternatives the client supports. If one based on D-ICE is on the list of supported transports and preferred among the supported, the below applies.

The transport specification will indicate which media protocol is to be used and based on this and the client's candidates, the server determines the protocol and if it supports ICE with that protocol. The server SHALL then gather its UDP candidates according to [section 4.1.1](#) in ICE [[RFC5245](#)] and any TCP based ones according to [section 5](#) of ICE TCP [[RFC6544](#)].

Servers that have an address that is generally reachable by any client within the address scope the server intends to serve MAY be specially configured (high-reachability configuration). This special configuration has the goal of reducing the server side candidate to preferably a single one per (address family, media stream, media component) tuple. Instead of gathering all possible addresses including relayed and server reflexive addresses, the server uses a single address per address family that the server knows should be reachable by a client behind one or more NATs. The reason for this special configuration is twofold: Firstly, it reduces the load on the server in address gathering and in ICE processing during the connectivity checks. Secondly, it will reduce the number of



permutations for candidate pairs significantly thus potentially speeding up the conclusion of the ICE processing. Note however that using this option on a server that doesn't fulfill the requirement of being reachable is counter-productive and it is important that this is correctly configured.

The above general consideration for servers applies also for TCP based candidates. A general implementation should support several candidate collection techniques and connection types. For TCP based candidates a high-reachability configured server is recommended to only offer Host candidates. In addition to passive connection types the server can select to provide active or simultaneous-open (S-O) connection types to match the client's candidates.

### **6.5. RTSP Server Response**

The server determines if the SETUP request is successful from the other perspectives and if so returns a 200 OK response; otherwise it returns an error code. At that point the server, having selected a transport specification using the "D-ICE" lower layer, will need to include that transport specification in the response message. The transport specification SHALL include the candidates gathered in [Section 6.4](#) in the "candidates" transport header parameter as well as the server's ICE username fragment and password. In the case that there are no valid candidate pairs with the combination of the client and server candidates, a 480 (ICE Processing Failed) error response SHALL be returned which MUST include the server's candidates. The return of a 480 error allows both the server and client to release their candidates.

Example of a successful response to the request in [Section 6.3](#).

```
S->C: RTSP/2.0 200 OK
      CSeq: 313
      Session: 12345678
      Transport: RTP/AVP/D-ICE; unicast; RTCP-mux; ICE-ufrag=MkQ3;
                ICE-Password=pos12Dgp9FcAjpq82ppaF; candidates="
                1 1 UDP 2130706431 192.0.2.56 50234 typ host"
      Accept-Ranges: NPT
      Date: 23 Jan 1997 15:35:06 GMT
      Server: PhonyServer 1.1
      Supported: setup.ice-d-m, setup.rtp.rtcp.mux
```

### **6.6. Server to Client ICE Connectivity Checks**

The server SHALL start the connectivity checks following the procedures described in [Section 5.7](#) and 5.8 of ICE [[RFC5245](#)] unless it is configured to use the high-reachability option. If it is then





it MAY suppress its own checks until the server's checks are triggered by the client's connectivity checks.

Please note that ICE [\[RFC5245\] section 5.8](#) does specify that the initiation of the checks are paced and new ones are only started every  $T_a$  milliseconds. The motivation for this is documented in [Appendix B.1](#) of ICE [\[RFC5245\]](#) as for SIP/SDP all media streams within an offer/answer dialog are running using the same queue. To ensure the same behavior with RTSP, the server SHALL use a single pacer queue for all media streams within each RTSP session.

The values for the pacing of STUN and TURN transactions  $T_a$  and  $RTO$  can be configured but have the same minimum values defined in the ICE specification.

When a connectivity check from the client reaches the server it will result in a triggered check from the server as specified in [Section 7.2.1.4](#) of ICE [\[RFC5245\]](#). This is why servers with a high-reachability address can wait until this triggered check to send out any checks for itself so saving resources and mitigating the DDoS potential.

#### **6.7. Client to Server ICE Connectivity Check**

The client receives the SETUP response and learns the candidate addresses to use for the connectivity checks. The client SHALL initiate its connectivity check(s), following the procedures in [Section 6](#) of ICE [\[RFC5245\]](#). The pacing of STUN transactions (Section B.1 of [\[RFC5245\]](#)) SHALL be used across all media streams that are part of the same RTSP session.

Aggressive nomination SHOULD be used with RTSP during initial SETUP for a resource. This doesn't have all the negative impact that it has in offer/answer as media playing only starts after issuing a PLAY request. Thus the issue with a change of the media path being used for delivery can be avoided by not issuing a PLAY request while STUN connectivity checks are still outstanding. Aggressive nomination can result in multiple candidate pairs having their nominated flag set but according to [Section 8.1.1.2](#) of ICE [\[RFC5245\]](#) when the PLAY request is sent the media will arrive on the pair with the highest priority. Note, different media resources may still end up with different foundations.

The above does not change ICE and its handling of aggressive nomination. When using aggressive nomination, a higher priority candidate pair with an outstanding connectivity check message can move into the Succeeded state and the candidate pair will have its Nominated flag set. This results in the higher priority candidate



pair being used instead of the previous pair, which is also in the Succeeded state.

To avoid this occurring during actual media transport, the RTSP client can add additional logic when the ICE processing overall is completed to indicate if there is still higher priority connectivity checks outstanding. If some check is still outstanding, the implementation can choose to wait until some additional timeout triggers or the outstanding checks completes before progressing with a PLAY request. An alternative is to accept the risk for a path change during media delivery and start playing immediately.

RTSP clients that want to ensure that each media resource uses the same path can use regular nomination where both the ICE processing completion criteria can be controlled in addition to which media streams being nominated for use. This does not affect the RTSP server, as its role is the one of being controlled.

#### **6.8. Client Connectivity Checks Complete**

When the client has concluded all of its connectivity checks and has nominated its desired candidate pair for a particular media stream, it MAY issue a PLAY request for that stream. Note, that due to the aggressive nomination, there is a risk that any outstanding check may nominate another pair than what was already nominated. The candidate pair with the highest priority will be used for the media. If the client has locally determined that its checks have failed it may try providing an extended set of candidates and update the server candidate list by issuing a new SETUP request for the media stream.

If the client concluded its connectivity checks successfully and therefore sent a PLAY request but the server cannot conclude successfully, the server will respond with a 480 (ICE Processing Failed). Upon receiving the 480 (ICE Processing Failed) response, the client may send a new SETUP request assuming it has any new information that can be included in the candidate list. If the server is still performing the checks when receiving the PLAY request it will respond with a 150 (ICE connectivity checks in progress) response to indicate this.

#### **6.9. Server Connectivity Checks Complete**

When the RTSP server receives a PLAY request, it checks to see that the connectivity checks have concluded successfully and only then will it play the stream. If the PLAY request is for a particular media stream, the server only needs to check that the connectivity checks for that stream completed successfully. If the server has not concluded its connectivity checks, the server indicates that by



sending the 150 (ICE connectivity checks in progress) ([Section 4.5.1](#)). If there is a problem with the checks, then the server sends a 480 response to indicate a failure of the checks. If the checks are successful then the server sends a 200 OK response and starts delivering media.

#### **[6.10.](#) Freeing Candidates**

Both server and client MAY free its non selected candidates as soon as a 200 PLAY response has been issued/received and no outstanding connectivity checks exist.

Clients and servers MAY free all their gathered candidates after having received or sent respectively a 480 response to a SETUP request. Client's will likely free their candidates first after having tried any additional actions that may resolve the issue, e.g. verifying the address gathering, or use additional STUN or TURN servers. Thus a server will have to weigh the cost of doing address gathering versus maintaining the gathered address for some time to allow any new SETUP request to be issued by the client.

If the 480 response is received or sent in response to a PLAY request, the server MUST NOT free its gathered candidates. Instead it will have to wait for additional actions from the client, or to terminate the RTSP session due to inactivity.

#### **[6.11.](#) Steady State**

The client and server SHALL use STUN to send keep-alive messages for the nominated candidate pair(s) following the rules of [Section 10](#) of ICE [[RFC5245](#)]. This is important as normally RTSP play mode sessions only contain traffic from the server to the client so the bindings in the NAT need to be refreshed by the client to server traffic provided by the STUN keep-alive.

#### **[6.12.](#) Re-SETUP**

A client that decides to change any parameters related to the media stream setup will send a new SETUP request. In this new SETUP request the client MAY include a new different ICE username fragment and password to use in the ICE processing. New ICE username and password SHALL cause the ICE processing to start from the beginning again, i.e., an ICE restart ([Section 9.1.1.1 of \[RFC5245\]](#)). The client SHALL in case of ICE restart gather candidates and include the candidates in the transport specification for D-ICE.

ICE restarts may be triggered due to changes of client or server attachment to the network, i.e., changes to the media streams



destination or source address or port. Most RTSP parameter changes would not require an ICE restart; instead existing mechanisms in RTSP for indicating from where in the RTP stream they apply is used. These include: Performing a pause prior to the parameter change and then resume; or assuming the server supports using SETUP during the PLAY state, using the RTP-Info header (Section 18.45 of [\[I-D.ietf-mmusic-rfc2326bis\]](#)) to indicate from where in the media stream the change shall apply.

Even if otherwise not supporting SETUP during PLAY state for other purposes, the server SHALL support SETUP requests in PLAY state, as long as the SETUP changes only the ICE parameters, which are: ICE-Password, ICE-ufrag and the content of ICE candidates.

If the RTSP session is in playing state at the time of sending the SETUP request requiring ICE restart, then the ICE connectivity checks SHALL use Regular nomination. Any ongoing media delivery continues on the previously nominated candidate pairs until the new pairs have been nominated for the individual media stream. Once the nomination of the new candidate pair has completed, all unused candidates may be released. If the ICE processing fails and no new candidate pairs are nominated for use, then the media stream MAY continue to use the previously nominated candidate pairs while they still function. If they appear to fail to transport media packets anymore then the client can select between two actions. First, if it can attempt any actions available that might make ICE work, like trying another STUN/TURN server, or change the transport parameters. In that case, the client modifies the RTSP session, and if ICE is still to be used, the client restarts ICE once more. If the client is unable to modify the transport or ICE parameters, it MUST NOT restart the ICE processing, and it SHOULD terminate the RTSP session.

#### **6.13. Server Side Changes After Steady State**

A Server may require an ICE restart because of server side load balancing or a failure resulting in an IP address and a port number change. In that case the server SHALL use the PLAY\_NOTIFY method to inform the client ([Section 13.5 \[I-D.ietf-mmusic-rfc2326bis\]](#)) with a new Notify-Reason header: ice-restart. The server will identify if the change is for a single media or for the complete session by including the corresponding URI in the PLAY\_NOTIFY request.

Upon receiving and responding to this PLAY\_NOTIFY with ice-restart reason the client SHALL gather new ICE candidates, and send SETUP requests for each media stream part of the session. The server provides its candidates in the SETUP response the same way as for the first time ICE processing. Both server and client SHALL provide new





ICE user names and passwords. The client MAY issue the SETUP request while the session is in PLAYING state.

If the RTSP session is in PLAYING state when the client issues the SETUP request, the client SHALL use regular nomination. If not the client will use the same procedures as for when first creating the session.

Note that keepalive messages on the previous set of candidate pairs SHOULD continue until all new candidate pairs have been nominated. After having nominated a new set of candidate pairs, the client may continue to receive media for some additional time. Even if the server stops delivering media over that candidate pair at the time of nomination, media may arrive for up to one maximum segment lifetime as defined in TCP (2 minutes). Unfortunately, if the RTSP server is divided into a separate controller and media stream, a failure may result in continued media delivery for a longer time than the maximum segment lifetime, thus source filtering is RECOMMENDED.

For example:

S->C: PLAY\_NOTIFY rtsp://example.com/fizzle/foo RTSP/2.0

CSeq: 854

Notify-Reason: ice-restart

Session: uZ3ci0K+Ld

Server: PhonyServer 1.1

C->S: RTSP/2.0 200 OK

CSeq: 854

User-Agent: PhonyClient/1.2

C->S: SETUP rtsp://server.example.com/fizzle/foo/audio RTSP/2.0

CSeq: 314

Session: uZ3ci0K+Ld

Transport: RTP/AVP/D-ICE; unicast; ICE-ufrag=Kl1C;

ICE-Password=H4sICGjBsEcCA3Rlc3RzLX; candidates="

1 1 UDP 2130706431 10.0.1.17 8998 typ host;

2 1 UDP 1694498815 192.0.2.3 51456 typ srflx

raddr 10.0.1.17 rport 9002"; RTCP-mux,

RTP/AVP/UDP; unicast; dest\_addr=":6970"/":6971",

RTP/AVP/TCP;unicast;interleaved=0-1

Accept-Ranges: NPT, UTC

User-Agent: PhonyClient/1.2

C->S: SETUP rtsp://server.example.com/fizzle/foo/video RTSP/2.0

CSeq: 315

Session: uZ3ci0K+Ld

Transport: RTP/AVP/D-ICE; unicast; ICE-ufrag=hZv9;



```

        ICE-Password=JAhA9myMHETTFNCrPtg+kJ; candidates="
        1 1 UDP 2130706431 10.0.1.17 9000 typ host;
        2 1 UDP 1694498815 192.0.2.3 51576 typ srflx
           raddr 10.0.1.17 rport 9000"; RTCP-mux,
        RTP/AVP/UDP; unicast; dest_addr=":6972"/":6973",
        RTP/AVP/TCP;unicast;interleaved=0-1
    Accept-Ranges: NPT, UTC
    User-Agent: PhonyClient/1.2

S->C: RTSP/2.0 200 OK
    CSeq: 314
    Session: uZ3ci0K+Ld
    Transport: RTP/AVP/D-ICE; unicast; RTCP-mux; ICE-ufrag=CbDm;
               ICE-Password=OfdXHws9XX0eBr6j2zz9Ak; candidates="
               1 1 UDP 2130706431 192.0.2.56 50234 typ host"
    Accept-Ranges: NPT
    Date: 11 March 2011 13:17:46 GMT
    Server: PhonyServer 1.1

S->C: RTSP/2.0 200 OK
    CSeq: 315
    Session: uZ3ci0K+Ld
    Transport: RTP/AVP/D-ICE; unicast; RTCP-mux; ICE-ufrag=jigs;
               ICE-Password=Dgx6fPj2lsa2WI8b7oJ7+s; candidates="
               1 1 UDP 2130706431 192.0.2.56 47233 typ host"
    Accept-Ranges: NPT
    Date: 11 March 2011 13:17:47 GMT
    Server: PhonyServer 1.1

```

## 7. ICE and Proxies

RTSP allows for proxies which can be of two fundamental types depending on whether they relay and potentially cache the media or not. Their differing impact on the RTSP NAT traversal solution, including backwards compatibility, is explained below.

### 7.1. Media Handling Proxies

An RTSP proxy that relays or caches the media stream for a particular media session can be considered to split the media transport into two parts: A media transport between the server and the proxy according to the proxy's need, and delivery from the proxy to the client. This split means that the NAT traversal solution will need to be run on each individual media leg according to need.

It is RECOMMENDED that any media handling proxy support the media NAT traversal defined within this specification. This is for two reasons: Firstly, to enable clients to perform NAT traversal for the



media between the proxy and itself, and secondly to allow the proxy to be topology independent to support performing NAT traversal (to the server) for non-NAT traversal capable clients present in the same address domain as the proxy.

For a proxy to support the media NAT traversal defined in this specification a proxy will need to implement the solution fully and be able to act as both a controlling and a controlled ICE peer. The proxy also SHALL include the "setup.ice-d-m" feature tag in any applicable capability negotiation headers, such as "Proxy-Supported".

### **7.2. Signalling Only Proxies**

A signalling only proxy handles only the RTSP signalling and does not have the media relayed through proxy functions. This type of proxy is not likely to work unless the media NAT traversal solution is in place between the client and the server, because the Denial of Service (DoS) protection measures, as discussed in [Section 21.2.1](#) of RTSP 2.0 [[I-D.ietf-mmusic-rfc2326bis](#)], usually prevent media delivery to addresses other than from where the RTSP signalling arrives at the server.

The solution for the Signalling Only proxy is that it must forward the RTSP SETUP requests including any transport specification with the "D-ICE" lower layer and the related transport parameters. A proxy supporting this functionality SHOULD indicate its capability by always including the "setup.ice-d-m" feature tag in the "Proxy-Supported" header.

### **7.3. Non-supporting Proxies**

A media handling proxy that doesn't support the ICE media NAT traversal specified here is assumed to remove the transport specification and use any of the lower prioritized transport specifications if provided by the requester. The specification of such a non ICE transport enables the negotiation to complete, although with a less preferred method since a NAT between the proxy and the client may result in failure of the media path.

A non-media handling proxy is expected to ignore and simply forward all unknown transport specifications, however, this can only be guaranteed for proxies following the RTSP 2.0 specification [[I-D.ietf-mmusic-rfc2326bis](#)].

Unfortunately, the usage of the "setup.ice-d-m" feature tag in the Proxy-Require will have contradicting results. For a non ICE supporting but media handling proxy, the inclusion of the feature tag will result in aborting the setup and indicating that it isn't



supported, which is desirable if you want to provide other fallbacks or other transport configurations to handle the situation. For non-supporting non-media handling proxies the result will also result in aborting the setup, however, setup might have worked if the feature tag wasn't present in the Proxy-Require header. This variance in results is the reason we don't recommend the usage of the Proxy-Require header. Instead we recommend the usage of the Supported header to force proxies to include the feature tags for the intersection of what the proxy chain support in the Proxy-Supported header. This will provide a positive indication when all proxies in the chain between the client and server support the functionality.

If a proxy doesn't support the setup.ice-d-m feature, but that proxy is not a media handling proxy, the ICE based setup could still work, since such a proxy may do passthrough on any transport parameters. Unfortunately the Proxy-Require and Proxy-Supported RTSP headers failed to provide that information. The only way of finding if this is the case or not is to try perform an SETUP including a Transport header with transport specifications using ICE.

## **8. RTP and RTCP Multiplexing**

"Multiplexing RTP Data and Control Packets on a Single Port" [[RFC5761](#)] specifies how and when RTP and RTCP can be multiplexed on the same port. This multiplexing SHALL be combined with ICE as it makes RTP and RTCP need only a single component per media stream instead of two, so reducing the load on the connectivity checks. For details on how to negotiate RTP and RTCP multiplexing, see [Appendix C](#) of RTSP 2.0 [[I-D.ietf-mmusic-rfc2326bis](#)].

Multiplexing RTP and RTCP has the benefit that it avoids the need for handling two components per media stream when RTP is used as the media transport protocol. This eliminates at least one STUN check per media stream and will also reduce the time needed to complete the ICE processing by at least the time it takes to pace out the additional STUN checks of up to one complete round trip time for a single media stream. In addition to the protocol performance improvements, the server and client side complexities are reduced as multiplexing halves the total number of STUN instances and holding the associated state. Multiplexing will also reduce the combinations and length of the list of possible candidates.

The implementation of RTP and RTCP multiplexing is additional work required for this solution. However, when implementing the ICE solution a server or client will need to implement a de-multiplexer between the STUN, and RTP or RTCP packets below the RTP/RTCP implementation anyway, so the additional work of one new





demultiplexing point directly connected to the STUN and RTP/RTCP seems small relative to the benefits provided.

Due to the above mentioned benefits, RTSP servers and clients that support "D-ICE" lower layer transport in combination with RTP SHALL also implement RTP and RTCP multiplexing as specified in this section and [[RFC5761](#)].

## **9. Fallback and Using Partial ICE functionality to improve NAT/Firewall traversal**

The need for fallback from ICE in RTSP should be less than for SIP using ICE in SDP offer/answer where a default destination candidate is very important to enable interworking with non-ICE capable endpoints. In RTSP, capability determination for ICE can happen prior to the RTSP SETUP request. This means a client should normally not need to include fallback alternatives when offering ICE, as the capability for ICE will already be determined. However, as described in this section, clients may wish to use part of the ICE functionality to improve NAT/Firewall traversal where the server is non-ICE capable.

[Section 4.1.4](#) of the ICE [[RFC5245](#)] specification does recommend that the default destination, i.e. what is used as fallback if the peer isn't ICE capable, is a candidate of relayed type to maximize the likelihood of successful transport of media. This is based on the peer in SIP using SDP offer/answer is almost as likely as the RTSP client to be behind a NAT. For RTSP the deployment of servers are much more heavily weighted towards deployment with public reachability. In fact since publicly reachable servers behind NAT either need to support ICE or have static configurations that allow traversal, one can assume that the server will have a public address or support ICE. Thus, the selection of the default destination address for RTSP can be differently prioritized.

As an ICE enabled client behind a NAT needs to be configured with a STUN server address to be able to gather candidates successfully, this can be used to derive a server reflexive candidate for the client's port. How useful this is for a NAT'ed RTSP client as a default candidate depends on the properties of the NAT. As long as the NAT uses an address independent mapping, then using a STUN derived reflexive candidate is likely to be successful. This is however brittle in several ways, and the main reason why the original specification of STUN [[RFC3489](#)] and direct usage for NAT traversal was obsoleted. First, if the NATs behavior is attempted to be determined using STUN as described in [[RFC3489](#)], the determined behavior might not be representative of the behavior encountered in another mapping. Secondly, filter state towards the ports used by



the server needs to be established. This requires that the server actually includes both address and ports in its response to the SETUP request. Thirdly messages need to be sent to these ports for keep-alive at a regular interval. How a server reacts to such unsolicited traffic is unknown. This brittleness may be accepted in fallback due to lack of support on the server side.

To maximize the likelihood that an RTSP client is capable of receiving media a relay based address should be chosen as the default fallback address. However, for RTSP clients lacking a relay server, like a TURN server, or where usage of such a server has significant cost associated with it the usage of a STUN derived server reflexive address as client default has a reasonable likelihood of functioning and may be used as an alternative.

Fallback addresses need to be provided in their own transport specification using a specifier that does not include the "D-ICE" lower layer transport. Instead the selected protocol, e.g. UDP needs to be explicitly or implicitly indicated. Secondly the selected default candidate needs to be included in the SETUP request. If this candidate is server reflexive or relayed the aspect of keep-alive needs to be ensured.

## **10. IANA Considerations**

This document requests registration in a number of registries, both for RTSP and SDP. For all the below registrations the contact person on behalf of the IETF WG MMUSIC is Magnus Westerlund; Postal address: Farogatan 6, 164 80 Stockholm, Sweden; Email: [magnus.westerlund@ericsson.com](mailto:magnus.westerlund@ericsson.com).

RFC-Editor Note: Please replace any occurrence of RFCXXXX in the below with the RFC number this specification is assigned.

### **10.1. RTSP Feature Tags**

This document request that one RTSP 2.0 feature tag is registered in the "RTSP 2.0 Feature-tags" registry:

setup.ice-d-m A feature tag representing the support of the ICE based establishment of datagram media transport that is capable of transport establishment through NAT and Firewalls. This feature tag applies to clients, servers and proxies and indicates support of all the mandatory functions of this specification.



## **10.2. Transport Protocol Specifications**

This document needs to register a number of transport protocol combinations in the RTSP 2.0 "Transport Protocol Specifications" registry.

"RTP/AVP/D-ICE" RTP using the AVP profile over an ICE established datagram flow.

"RTP/AVPF/D-ICE" RTP using the AVPF profile over an ICE established datagram flow.

"RTP/SAVP/D-ICE" RTP using the SAVP profile over an ICE established datagram flow.

"RTP/SAVPF/D-ICE" RTP using the SAVPF profile over an ICE established datagram flow.

## **10.3. RTSP Transport Parameters**

This document requests that 3 transport parameters are registered in the RTSP 2.0's "Transport Parameters" registry:

"candidates": Listing the properties of one or more ICE candidate. See [Section 4.2](#) of RFCXXXX.

"ICE-Password": The ICE password used to authenticate the STUN binding request in the ICE connectivity checks. See [Section 4.3](#) of RFCXXXX.

"ICE-ufrag": The ICE username fragment used to authenticate the STUN binding requests in the ICE connectivity checks. See [Section 4.3](#) of RFCXXXX.

## **10.4. RTSP Status Codes**

This document requests that 2 assignments are done in the "RTSP 2.0 Status Codes" registry. The values are:

150: The 150 response code indicates that ICE connectivity checks are still in progress and haven't concluded. This response SHALL be sent within 200 milliseconds of receiving a PLAY request that currently can't be fulfilled because ICE connectivity checks are still running. Subsequently, every 3 seconds after the previous sent one, a 150 reply SHALL be sent until the ICE connectivity checks conclude either successfully or in failure, and a final response for the request can be provided.



480: The 480 response code is used in cases when the request can't be fulfilled due to a failure in the ICE processing, such as all the connectivity checks have timed out. This error message can appear either in response to a SETUP request to indicate that no candidate pair can be constructed or to a PLAY request that the server's connectivity checks resulted in failure.

### **10.5. Notify-Reason value**

This document requests that one assignment is done in the RTSP 2.0 Notify-Reason header value registry. The defined value is:

ice-restart: Server notifying the client about the need for an ICE restart. See [Section 4.6](#).

### **10.6. SDP Attribute**

The registration of one SDP attribute is requested:

SDP Attribute ("att-field"):

Attribute name:	rtsp-ice-d-m
Long form:	ICE for RTSP datagram media NAT traversal
Type of attribute:	Session-level only
Subject to charset:	No
Purpose:	RFC XXXX, <a href="#">Section 4.7</a>
Values:	No values defined
Contact:	Magnus Westerlund
	E-mail: magnus.westerlund@ericsson.com
	phone: +46 10 714 82 87

## **11. Security Considerations**

ICE [[RFC5245](#)] and ICE TCP [[RFC6544](#)] provide an extensive discussion on security considerations which apply here as well.

### **11.1. ICE and RTSP**

A long-standing risk with transmitting a packet stream over UDP is that the host may not be interested in receiving the stream. On today's Internet, many hosts are behind NATs or operate host firewalls which do not respond to unsolicited packets with an ICMP port unreachable error. Thus, an attacker can construct RTSP SETUP requests with a victim's IP address and cause a flood of media packets to be sent to a victim. The addition of ICE, as described in this document, provides protection from the attack described above. By performing the ICE connectivity check, the media server receives confirmation that the RTSP client wants the media. While this





protection could also be implemented by requiring the IP addresses in the SDP match the IP address of the RTSP signaling packet, such a mechanism does not protect other hosts with the same IP address (such as behind the same NAT), and such a mechanism would prohibit separating the RTSP controller from the media play-out device (e.g., an IP-enabled remote control and an IP-enabled television); it also forces RTSP proxies to relay the media streams through them, even if they would otherwise be only signalling proxies.

To protect against the attacks in ICE based on signalling information RTSP signalling SHOULD be protected using TLS to prevent eavesdropping and modification of information.

The STUN amplification attack described in [Section 18.5.2](#) in ICE [[RFC5245](#)] needs consideration. Servers that are able to run according to the high-reachability option have good mitigation against this attack as they only send connectivity checks towards an address and port pair they have received an incoming connectivity check from. This means an attacker requires both the capability to spoof source addresses and to signal the RTSP server a set of ICE candidates. Independently an ICE agent needs to implement the mitigation to reduce the volume of the amplification attack as described in the ICE specification.

## **[12.](#) Acknowledgements**

The authors would like to thank Remi Denis-Courmont for suggesting the method of integrating ICE in RTSP signalling, Dan Wing for help with the security section and numerous other issues, Ari Keranen for review of the document and its ICE details. Flemming Andreassen for a thorough review. In addition Bill Atwood have provided comments and suggestions for improvements.

## **[13.](#) References**

### **[13.1.](#) Normative References**

- [I-D.ietf-mmusic-rfc2326bis]  
Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M.,  
and M. Stiemerling, "Real Time Streaming Protocol 2.0  
(RTSP)", [draft-ietf-mmusic-rfc2326bis-39](#) (work in  
progress), January 2014.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate  
Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.



- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, [RFC 3986](#), January 2005.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", [RFC 4566](#), July 2006.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, [RFC 5234](#), January 2008.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", [RFC 5245](#), April 2010.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", [RFC 5389](#), October 2008.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", [RFC 5761](#), April 2010.
- [RFC6544] Rosenberg, J., Keranen, A., Lowekamp, B., and A. Roach, "TCP Candidates with Interactive Connectivity Establishment (ICE)", [RFC 6544](#), March 2012.

### **13.2. Informative References**

- [I-D.ietf-mmusic-rtsp-nat-evaluation] Westerlund, M. and T. Zeng, "The Evaluation of Different Network Address Translator (NAT) Traversal Techniques for Media Controlled by Real-time Streaming Protocol (RTSP)", [draft-ietf-mmusic-rtsp-nat-evaluation-13](#) (work in progress), February 2014.
- [RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", [RFC 2326](#), April 1998.
- [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", [RFC 3022](#), January 2001.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", [RFC 3261](#), June 2002.



- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", [RFC 3264](#), June 2002.
- [RFC3489] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", [RFC 3489](#), March 2003.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", [RFC 4340](#), March 2006.

#### Authors' Addresses

Jeff Goldberg  
Cisco  
11 New Square, Bedfont Lakes  
Feltham,, Middx TW14 8HA  
United Kingdom

Phone: +44 20 8824 1000  
Email: [jgoldber@cisco.com](mailto:jgoldber@cisco.com)

Magnus Westerlund  
Ericsson  
Farogatan 6  
Stockholm SE-164 80  
Sweden

Phone: +46 8 719 0000  
Email: [magnus.westerlund@ericsson.com](mailto:magnus.westerlund@ericsson.com)

Thomas Zeng  
Nextwave Wireless, Inc.  
12670 High Bluff Drive  
San Diego, CA 92130  
USA

Phone: +1 858 480 3100  
Email: [thomas.zeng@gmail.com](mailto:thomas.zeng@gmail.com)

