Network Working Group
Internet-Draft
Updates: 5104 (if approved)
Intended status: Standards Track
Expires: March 6, 2016

B. Burman
A. Akram
Ericsson
R. Even
Huawei Technologies
M. Westerlund
Ericsson

September 3, 2015

# RTP Stream Pause and Resume
## draft-ietf-avtext-rtp-stream-pause-09

## Abstract

With the increased popularity of real-time multimedia applications, it is desirable to provide good control of resource usage, and users also demand more control over communication sessions. This document describes how a receiver in a multimedia conversation can pause and resume incoming data from a sender by sending real-time feedback messages when using Real-time Transport Protocol (RTP) for real time data transport. This document extends the Codec Control Messages (CCM) RTP Control Protocol (RTCP) feedback package by explicitly allowing and describing specific use of existing CCM messages and adding a group of new real-time feedback messages used to pause and resume RTP data streams. This document updates RFC 5104.

## 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 March 6, 2016.

Table of Contents

## 1. Introduction

As real-time communication attracts more people, more applications are created; multimedia conversation applications being one example. Multimedia conversation further exists in many forms, for example, peer-to-peer chat application and multiparty video conferencing controlled by central media nodes, such as RTP Mixers.

Multimedia conferencing may involve many participants; each has its own preferences for the communication session, not only at the start but also during the session. This document describes several scenarios in multimedia communication where a conferencing node or participant chooses to temporarily pause an incoming RTP [RFC3550] stream and later resume it when needed. The receiver does not need to terminate or inactivate the RTP session and start all over again by negotiating the session parameters, for example using SIP [RFC3261] with SDP [RFC4566] Offer/Answer [RFC3264].

Centralized nodes, like RTP Mixers or Multipoint Control Units (MCU), which either use logic based on voice activity, other measurements, or user input could reduce the resources consumed in both the sender and the network by temporarily pausing the RTP streams that aren't required by the RTP Mixer. If the number of conference participants are greater than what the conference logic has chosen to present simultaneously to receiving participants, some participant RTP streams sent to the RTP Mixer may not need to be forwarded to any other participant. Those RTP streams could then be temporarily paused. This becomes especially useful when the media sources are provided in multiple encoding versions (Simulcast)

[I-D.ietf-mmusic-sdp-simulcast] or with Multi-Session Transmission (MST) of scalable encoding such as SVC [RFC6190]. There may be some of the defined encodings or combination of scalable layers that are not used or cannot be used all of the time. As an example, a centralized node may choose to pause such unused RTP streams without being explicitly requested to do so, maybe due to temporarily limited network or processing resources. It may then also send an explicit indication that the streams are paused.

As the set of RTP streams required at any given point in time is highly dynamic in such scenarios, using the out-of-band signaling channel for pausing, and even more importantly resuming, an RTP stream is difficult due to the performance requirements. Instead, the pause and resume signaling should be in the media plane and go directly between the affected nodes. When using RTP [RFC3550] for media transport, using the Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF) [RFC4585] appears appropriate. No currently existing RTCP feedback message explicitly supports pausing and resuming an incoming RTP stream. As





this affects the generation of packets and may even allow the encoding process to be paused, the functionality appears to match Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF) [RFC5104]. This document defines the solution as a Codec Control Message (CCM) extension.

The Temporary Maximum Media Bitrate Request (TMMBR) message of CCM is used by video conferencing systems for flow control. It is desirable to be able to use that method with a bitrate value of zero for pause, whenever possible. This specification updates RFC 5104 to add the new Pause and Resume semantics to the TMMBR/TMMBN messages.

## 2. Definitions

### 2.1. Abbreviations

AVPF: Audio-Visual Profile with Feedback (RFC 4585)

CCM: Codec Control Messages (RFC 5104)

CNAME: Canonical Name (RTCP SDES)

CSRC: Contributing Source (RTP)

FCI: Feedback Control Information (AVPF)

FIR: Full Intra Refresh (CCM)

FMT: Feedback Message Type (AVPF)

MCU: Multipoint Control Unit

MTU: Maximum Transfer Unit

PT: Payload Type (RTP)

RTP: Real-time Transport Protocol (RFC 3550)

RTCP: RTP Control Protocol (RFC 3550)

RTCP RR: RTCP Receiver Report

RTCP SR: RTCP Sender Report

SDP: Session Description Protocol (RFC 4566)

SIP: Session Initiation Protocol (RFC 3261)





SSRC: Synchronization Source (RTP)

SVC: Scalable Video Coding

TMMBR: Temporary Maximum Media Bitrate Request (CCM)

TMMBN: Temporary Maximum Media Bitrate Notification (CCM)

UA: User Agent (SIP)

UDP: User Datagram Protocol (RFC 768)

## 2.2. Terminology

In addition to the following, the definitions from RTP [RFC3550],

AVPF [RFC4585], CCM [RFC5104], and RTP Taxonomy [I-D.ietf-avtext-rtp-grouping-taxonomy] also apply in this document.

Feedback Messages: CCM [RFC5104] categorized different RTCP feedback messages into four types, Request, Command, Indication and Notification. This document places the PAUSE and RESUME messages into Request category, PAUSED as Indication and REFUSED as Notification.

PAUSE: Request from an RTP stream receiver to pause a stream

RESUME: Request from an RTP stream receiver to resume a paused stream

PAUSED: Indication from an RTP stream sender that a stream is paused

REFUSED: Notification from an RTP stream sender that a PAUSE or RESUME request will not be honored

Mixer: The intermediate RTP node which receives an RTP stream from different endpoints, combines them to make one RTP stream and forwards to destinations, in the sense described in Topo-Mixer of RTP Topologies [I-D.ietf-avtcore-rtp-topologies-update].

Participant: A member which is part of an RTP session, acting as receiver, sender or both.

Paused sender: An RTP stream sender that has stopped its transmission, i.e. no other participant receives its RTP transmission, either based on having received a PAUSE request, defined in this specification, or based on a local decision.





Pausing receiver: An RTP stream receiver which sends a PAUSE request, defined in this specification, to other participant(s).

Stream: Used as a short term for RTP stream, unless otherwise noted.

Stream receiver: Short for RTP stream receiver; the RTP entity responsible for receiving an RTP stream, usually a Media Depacketizer.

Stream sender: Short for RTP stream sender; the RTP entity responsible for creating an RTP stream, usually a Media Packetizer.

## 2.3. Requirements Language

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

# 3. Use Cases

This section discusses the main use cases for RTP stream pause and resume.

RTCWEB WG's use case and requirements document [RFC7478] defines the following API requirements in Appendix A, used also by W3C WebRTC WG:

A8 The Web API must provide means for the web application to mute/ unmute a stream or stream component(s). When a stream is sent to a peer mute, status must be preserved in the stream received by the peer.

A9 The Web API must provide means for the web application to cease the sending of a stream to a peer.

This document provides means to optimize transport usage by stop sending muted streams and start sending again when unmuting. It is here assumed that "mute" above can be taken to apply also to other media than audio. At the time of publication for this specification, RTCWEB did not specify any pause / resume functionality.

## 3.1. Point to Point

This is the most basic use case with an RTP session containing two Endpoints. Each Endpoint sends one or more streams.

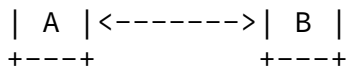



Figure 1: Point to Point

The usage of RTP stream pause in this use case is to temporarily halt delivery of streams that the sender provides but the receiver does not currently use. This can for example be due to minimized applications where the video stream is not actually shown on any display, and neither is it used in any other way, such as being recorded.

In this case, since there is only a single receiver of the stream, pausing or resuming a stream does not impact anyone else than the sender and the single receiver of that stream.

## 3.2. RTP Mixer to Media Sender

One of the most commonly used topologies in centralized conferencing is based on the RTP Mixer [I-D.ietf-avtcore-rtp-topologies-update]. The main reason for this is that it provides a very consistent view of the RTP session towards each participant. That is accomplished through the Mixer originating its own streams, identified by distinct SSRC values, and any RTP streams sent to the participants will be sent using those SSRC values. If the Mixer wants to identify the underlying media sources for its conceptual streams, it can identify them using CSRC. The stream the Mixer provides can be an actual mix of multiple media sources, but it might also be switching received streams as described in Sections 3.6-3.8 of [I-D.ietf-avtcore-rtp-topologies-update].

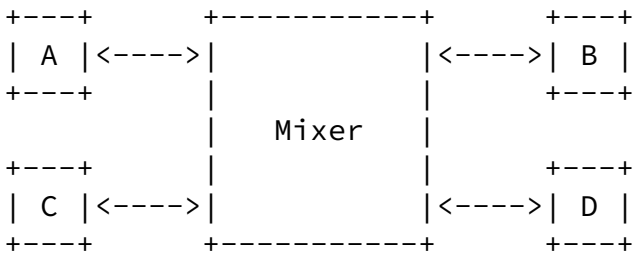



Figure 2: RTP Mixer in Unicast-only

Which streams from clients B, C and D that are delivered to a given receiver, A, can depend on several things. It can either be the RTP Mixer's own logic and measurements such as voice activity on the incoming audio streams. It can be that the number of sent media sources exceed what is reasonable to present simultaneously at any given receiver. It can also be a human controlling the conference

that determines how the media should be mixed; this would be more common in lecture or similar applications where regular listeners may be prevented from breaking into the session unless approved by the moderator. The streams may also be part of a Simulcast [I-D.ietf-mmusic-sdp-simulcast] or scalable encoded (for Multi-Session Transmission) [RFC6190], thus providing multiple versions that can be delivered by the RTP stream sender. These examples indicate that there are numerous reasons why a particular stream would not currently be in use, but must be available for use at very short notice if any dynamic event occurs that causes a different stream selection to be done in the Mixer.

Because of this, it would be highly beneficial if the Mixer could request the RTP stream sender to pause a particular stream. The Mixer also needs to be able to request the RTP stream sender to resume delivery with minimal delay.

In some cases, especially when the Mixer sends multiple RTP streams per receiving client, there may be situations that makes it desirable for the Mixer to pause some of its sent RTP streams, even without being explicitly asked to do so by the receiving client. Such situations can for example be caused by a temporary lack of available Mixer network or processing resources. An RTP stream receiver that no longer receives an RTP stream could interpret this as an error condition and try to take action to re-establish the RTP stream. Such action would likely be undesirable if the RTP stream was in fact deliberately paused by the Mixer. Undesirable RTP stream receiver actions could be avoided if the Mixer is able to explicitly indicate that an RTP stream is deliberately paused.

Just as for point-to-point (Section 3.1), there is only a single receiver of the stream, the RTP Mixer, and pausing or resuming a stream does not affect anyone else than the sender and single receiver of that stream.

## 3.3. RTP Mixer to Media Sender in Point-to-Multipoint

This use case is similar to the previous section, however the RTP Mixer is involved in three domains that need to be separated; the Multicast Network (including participants A and C), participant B, and participant D. The difference from above is that A and C share a multicast domain, which is depicted below.

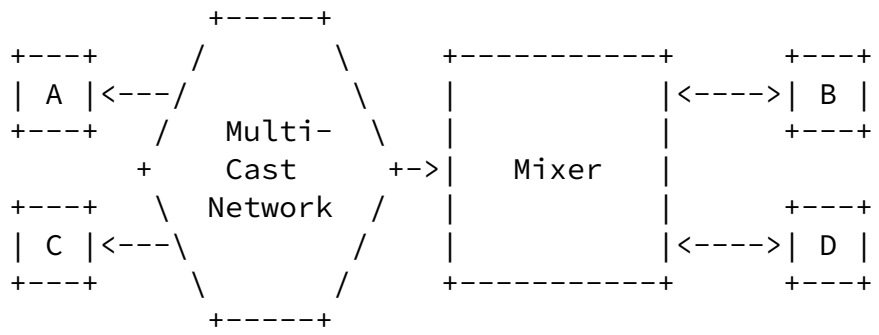



Figure 3: RTP Mixer in Point-to-Multipoint

If the RTP Mixer pauses a stream from A, it will not only pause the stream towards itself, but will also stop the stream from arriving to C, which C is heavily impacted by, might not approve of, and should thus have a say on.

If the Mixer resumes a paused stream from A, it will be resumed also towards C. In this case, if C is not interested it can simply ignore the stream and is not impacted as much as above.

In this use case there are several receivers of a stream and the Mixer must take special care so as not to pause a stream that is still wanted by some receivers.

## 3.4. Media Receiver to RTP Mixer

In this use case, the direction of the request to pause is the opposite compared to the two previous use cases. An Endpoint in Figure 2 could potentially request to pause the delivery of a given stream. Possible reasons include the ones in the point to point case (Section 3.1) above.

When the RTP Mixer is only connected to individual unicast paths, the use case and any considerations are identical to the point to point use case.

However, when the Endpoint requesting stream pause is connected to the RTP Mixer through a multicast network, such as A or C in

Figure 3, the use case instead becomes identical to the one in Section 3.3, only with reverse direction of the streams and pause/ resume requests.

## 3.5. Media Receiver to Media Sender Across RTP Mixer

An Endpoint, like A in Figure 2, could potentially request to pause the delivery of a given stream, like one of B's, over any of the SSRCs used by the Mixer by sending a pause request for the CSRC





identifying the stream. However, the authors are of the opinion that this is not a suitable solution, for several reasons:

1. The Mixer might not include CSRC in its stream indications.

2. An Endpoint cannot rely on the CSRC to correctly identify the stream to be paused when the delivered media is some type of mix. A more elaborate stream identification solution is needed to support this in the general case.

3. The Endpoint cannot determine if a given stream is still needed by the RTP Mixer to deliver to another session participant.

Due to the above reasons, we exclude this use case from further consideration.

# 4. Design Considerations

This section describes the requirements that this specification needs to meet.

## 4.1. Real-time Nature

The first section (Section 1) of this specification describes some possible reasons why a receiver may pause an RTP sender. Pausing and resuming is time-dependent, i.e. a receiver may choose to pause an RTP stream for a certain duration, after which the receiver may want the sender to resume. This time dependency means that the messages related to pause and resume must be transmitted to the sender in a timely fashion in order for them to be purposeful. The pause operation is arguably not as time critical as the resume operation, since it mainly provides a reduction of resource usage. Timely

handling of the resume operation is however likely to directly impact the end-user's perceived quality experience, since it affects the availability of media that the user expects to receive more or less instantly. It may also be highly desirable for a receiver to quickly learn that an RTP stream is intentionally paused on the RTP sender's own behalf.

## 4.2. Message Direction

It is the responsibility of an RTP stream receiver that wants to pause or resume a stream from the sender(s) to transmit PAUSE and RESUME messages. An RTP stream sender that wants to pause itself can often simply do it, but sometimes this will adversely affect the receiver and an explicit indication that the RTP stream is paused may then help. Any indication that an RTP stream is paused is the





responsibility of the RTP stream sender and may in some cases not even be needed by the stream receiver.

## 4.3. Apply to Individual Sources

The PAUSE and RESUME messages apply to single RTP streams identified by their SSRC, which means the receiver targets the sender's SSRC in the PAUSE and RESUME requests. If a paused sender starts sending with a new SSRC, the receivers will need to send a new PAUSE request in order to pause it. PAUSED indications refer to a single one of the sender's own, paused SSRC.

## 4.4. Consensus

An RTP stream sender should not pause an SSRC that some receiver still wishes to receive.

The reason is that in RTP topologies where the stream is shared between multiple receivers, a single receiver on that shared network must not single-handedly cause the stream to be paused without letting all other receivers to voice their opinions on whether or not the stream should be paused. Such shared networks can for example be multicast, a mesh with a joint RTP session, or a transport Translator based network. A consequence of this is that a newly joining receiver firstly needs to learn the existence of paused streams, and

secondly should be able to resume any paused stream. A newly joining receiver can for example be detected through an RTCP Receiver Report containing both a new SSRC and a CNAME that does not already occur in the session. Any single receiver wanting to resume a stream should also cause it to be resumed. An important exception to this is when the RTP stream sender is aware of conditions that makes it desirable or even necessitates to pause the RTP stream on its own behalf, without being explicitly asked to do so. Such local consideration in the RTP sender takes precedence over RTP receiver wishes to receive the stream.

## 4.5. Message Acknowledgments

RTP and RTCP does not guarantee reliable data transmission. It uses whatever assurance the lower layer transport protocol can provide. However, this is commonly UDP that provides no reliability guarantees. Thus it is possible that a PAUSE and/or RESUME message transmitted from an RTP Endpoint does not reach its destination, i.e. the targeted RTP stream sender. When PAUSE or RESUME reaches the RTP stream sender and are effective, i.e., an active RTP stream sender pauses, or a resuming RTP stream sender have media data to transmit, it is immediately seen from the arrival or non-arrival of RTP packets





for that RTP stream. Thus, no explicit acknowledgments are required in this case.

In some cases when a PAUSE or RESUME message reaches the RTP stream sender, it will not be able to pause or resume the stream due to some local consideration, for example lack of data to transmit. In this error condition, a negative acknowledgment may be needed to avoid unnecessary retransmission of requests (Section 4.6).

## 4.6. Request Retransmission

When the stream is not affected as expected by a PAUSE or RESUME request, the request may have been lost and the sender of the request will need to retransmit it. The retransmission should take the round trip time into account, and will also need to take the normal RTCP bandwidth and timing rules applicable to the RTP session into account, when scheduling retransmission of feedback.

When it comes to resume requests or unsolicited paused indications that are more time critical, the best performance may be achieved by repeating the message as often as possible until a sufficient number have been sent to reach a high probability of message delivery, or at an explicit indication that the message was delivered. For resume requests, such explicit indication can be delivery of the RTP stream being requested to be resumed.

## 4.7. Sequence Numbering

A PAUSE request message will need to have a sequence number to separate retransmissions from new requests. A retransmission keeps the sequence number unchanged, while it is incremented every time a new PAUSE request is transmitted that is not a retransmission of a previous request.

Since RESUME always takes precedence over PAUSE and are even allowed to avoid pausing a stream, there is a need to keep strict ordering of PAUSE and RESUME. Thus, RESUME needs to share sequence number space with PAUSE and implicitly references which PAUSE it refers to. For the same reasons, the explicit PAUSED indication also needs to share sequence number space with PAUSE and RESUME.

## 4.8. Relation to Other Solutions

A performance comparison between SIP/SDP and RTCP signaling technologies was made and included in draft versions of this specification. Using SIP and SDP to carry pause and resume information means that it will need to traverse the entire signaling path to reach the signaling destination (either the remote Endpoint





or the entity controlling the RTP Mixer), across any signaling proxies that potentially also has to process the SDP content to determine if they are expected to act on it. The amount of bandwidth required for a SIP/SDP-based signaling solution is in the order of at least 10 times more than an RTCP-based solution. Especially for UA sitting on mobile wireless access, this will risk introducing delays that are too long (Section 4.1) to provide a good user experience, and the bandwidth cost may also be considered infeasible compared to an RTCP-based solution. RTCP data is sent through the media path, which is likely shorter (contains fewer intermediate nodes) than the signaling path, may anyway have to traverse a few intermediate nodes.

The amount of processing and buffering required in intermediate nodes to forward those RTCP messages is however believed to be significantly less than for intermediate nodes in the signaling path. Based on those considerations, RTCP is chosen as signaling protocol for the pause and resume functionality.

## 5. Solution Overview

The proposed solution implements PAUSE and RESUME functionality based on sending AVPF RTCP feedback messages from any RTP session participant that wants to pause or resume a stream targeted at the stream sender, as identified by the sender SSRC.

This solution re-uses CCM TMMBR and TMMBN [RFC5104] to the extent possible, and defines a small set of new RTCP feedback messages where new semantics is needed.

A single Feedback message specification is used to implement the new messages. The message consists of a number of Feedback Control Information (FCI) blocks, where each block can be a PAUSE request, a RESUME request, PAUSED indication, a REFUSED notification, or an extension to this specification. This structure allows a single feedback message to handle pause functionality on a number of streams.

The PAUSED functionality is also defined in such a way that it can be used standalone by the RTP stream sender to indicate a local decision to pause, and inform any receiver of the fact that halting media delivery is deliberate and which RTP packet was the last transmitted.

Special considerations that apply when using TMMBR/TMMBN for pause and resume purposes are described in Section 5.6. This specification applies to both the new messages defined in herein as well as their TMMBR/TMMBN counterparts, except when explicitly stated otherwise. An obvious exception are any reference to the message parameters that are only available in the messages defined here. For example, any reference to PAUSE in the text below is equally applicable to TMMBR





0, and any reference to PAUSED is equally applicable to TMMBN 0. Therefore and for brevity, TMMBR/TMMBN will not be mentioned in the text, unless there is specific reason to do so.

This section is intended to be explanatory and therefore intentionally contains no mandatory statements. Such statements can instead be found in other parts of this specification.

## 5.1. Expressing Capability

An Endpoint can use an extension to CCM SDP signaling to declare capability to understand the messages defined in this specification. Capability to understand only a subset of messages is possible, to support partial implementation, which is specifically believed to be feasible for the RTP Mixer to Media Sender use case (Section 3.2). In that use case, only the RTP Mixer has capability to request the Media Sender to pause or resume. Consequently, in that same use case only the Media Sender has capability to pause and resume its sent streams based on requests from the RTP Mixer. Allowing for partial implementation of this specification is not believed to hamper interoperability, as long as the subsets are well defined and describe a consistent functionality, including a description of how a more capable implementation must perform fallback.

For the case when TMMBR/TMMBN are used for pause and resume purposes, it is possible to explicitly express joint support for TMMBR and TMMBN, but not for TMMBN only.

## 5.2. PauseID

All messages defined in this specification (Section 8) contain a PauseID, satisfying the design consideration on sequence numbering (Section 4.7). This PauseID is scoped by and thus a property of the targeted RTP stream (SSRC), not only a sequence number for individual messages. Instead, it numbers an entire "pause and resume operation" for the RTP stream, typically keeping PauseID constant for multiple, related messages. The PauseID value used during such operation is called the current PauseID. A new "pause and resume operation" is defined to start when the RTP stream sender resumes the RTP stream after it was being paused. The current PauseID is then incremented by one, in modulo arithmetic. In the subsequent descriptions below, it is sometimes necessary to refer to PauseID values that were already used as current PauseID, which is denoted as past PauseID. It should be noted that since PauseID uses modulo arithmetic, a past PauseID may have a larger value than the current PauseID. Since PauseID uses modulo arithmetic, it is also useful to define what PauseID values that are considered "past", to clearly separate it from what could be considered "future" PauseID values. Half of the

entire PauseID value range is chosen to represent past PauseID, while a quarter of the PauseID value range is chosen to represent future values. The remaining quarter of the PauseID value range is intentionally left undefined in that respect.

## 5.3. Requesting to Pause

An RTP stream receiver can choose to send a PAUSE request at any time, subject to AVPF timing rules.

The PAUSE request contains the current PauseID (Section 5.2).

When a non-paused RTP stream sender receives the PAUSE request, it continues to send the RTP stream while waiting for some time to allow other RTP stream receivers in the same RTP session that saw this PAUSE request to disapprove by sending a RESUME (Section 5.5) for the same stream and with the same current PauseID as in the PAUSE being disapproved. If such disapproving RESUME arrives at the RTP stream sender during the hold-off period before the stream is paused, the pause is not performed. In point-to-point configurations, the hold-off period may be set to zero. Using a hold-off period of zero is also appropriate when using TMMBR 0 and in line with the semantics for that message.

If the RTP stream sender receives further PAUSE requests with the current PauseID while waiting as described above, those additional requests are ignored.

If the PAUSE request is lost before it reaches the RTP stream sender, it will be discovered by the RTP stream receiver because it continues to receive the RTP stream. It will also not see any PAUSED indication (Section 5.4) for the stream. The same condition can be caused by the RTP stream sender having received a disapproving RESUME from a stream receiver A for a PAUSE request sent by a stream sender B, but that the PAUSE sender (B) did not receive the RESUME (from A) and may instead think that the PAUSE was lost. In both cases, a PAUSE request can be re-transmitted using the same current PauseID. If using TMMBR 0, the request MAY be re-transmitted when the requester fails to receive a TMMBN 0 confirmation.

If the pending stream pause is aborted due to a disapproving RESUME, the pause and resume operation for that PauseID is concluded, the current PauseID is updated, and any new PAUSE must therefore use the new current PauseID to be effective.

An RTP stream sender receiving a PAUSE not using the current PauseID informs the RTP stream receiver sending the ineffective PAUSE of this

condition by sending a REFUSED notification that contains the current PauseID value.

A situation where an ineffective PauseID is chosen can appear when a new RTP stream receiver joins a session and wants to PAUSE a stream, but does not yet know the current PauseID to use. The REFUSED notification will then provide sufficient information to create a valid PAUSE. The required extra signaling round-trip is not considered harmful, since it is assumed that pausing a stream is not time-critical (Section 4.1).

There may be local considerations making it impossible or infeasible to pause the stream, and the RTP stream sender can then respond with a REFUSED. In this case, if the used current PauseID would otherwise have been effective, REFUSED contains the same current PauseID as in the PAUSE request. Note that when using TMMBR 0 as PAUSE, that request cannot be refused (TMMBN > 0) due to the existing restriction in section 4.2.2.2 of [RFC5104] that TMMBN shall contain the current bounding set, and the fact that a TMMBR 0 will always be the most restrictive point in any bounding set, regardless of bounding set overhead value.

If the RTP stream sender receives several identical PAUSE for an RTP stream that was already at least once responded with REFUSED and the condition causing REFUSED remains, those additional REFUSED should be sent with regular RTCP timing. A single REFUSED can respond to several identical PAUSE requests.

## 5.4. Media Sender Pausing

An RTP stream sender can choose to pause the stream at any time. This can either be as a result of receiving a PAUSE, or be based on some local sender consideration. When it does, it sends a PAUSED indication, containing the current PauseID. Note that current PauseID in an unsolicited PAUSED (without having received a PAUSE), is incremented compared to previously sent PAUSED. It also sends the PAUSED indication in the next two regular RTCP reports, given that the pause condition is then still effective.

There is no reply to a PAUSED indication; it is simply an explicit

indication of the fact that an RTP stream is paused. This can be helpful for the RTP stream receiver, for example to quickly understand that transmission is deliberately and temporarily suspended and no specific corrective action is needed.

The RTP stream sender may want to apply some local consideration to exactly when the RTP stream is paused, for example completing some





media unit or a forward error correction block, before pausing the stream.

The PAUSED indication also contains information about the RTP extended highest sequence number when the pause became effective. This provides RTP stream receivers with firsthand information allowing them to know whether they lost any packets just before the stream paused or when the stream is resumed again. This allows RTP stream receivers to quickly and safely take into account that the stream is paused, in for example retransmission or congestion control algorithms.

If the RTP stream sender receives PAUSE requests with the current PauseID while the stream is already paused, those requests are ignored.

As long as the stream is being paused, the PAUSED indication MAY be sent together with any regular RTCP Sender Report (SR) or Receiver Report (RR). Including PAUSED in this way allows RTP stream receivers joining while the stream is paused to quickly know that there is a paused stream, what the last sent extended RTP sequence number was, and what the current PauseID is to be able to construct valid PAUSE and RESUME requests at a later stage.

When the RTP stream sender learns that a new Endpoint has joined the RTP session, for example by a new SSRC and a CNAME that was not previously seen in the RTP session, it should send PAUSED indications for all its paused streams at its earliest opportunity. It should in addition continue to include PAUSED indications in at least two regular RTCP reports.

## 5.5. Requesting to Resume

An RTP stream receiver can request the RTP stream sender to resume a stream with a RESUME request at any time, subject to AVPF timing rules. The RTP stream receiver must include the current PauseID in the RESUME request for it to be effective.

A pausing RTP stream sender that receives a RESUME including the current PauseID resumes the stream at the earliest opportunity. Receiving RESUME requests for a stream that is not paused does not require any action and can be ignored.

There may be local considerations at the RTP stream sender, for example that the media device is not ready, making it temporarily impossible to resume the stream at that point in time, and the RTP stream sender can then respond with a REFUSED containing the current PauseID. When receiving such REFUSED with a current PauseID





identical to the one in the sent RESUME, RTP stream receivers should avoid sending further RESUME requests for some reasonable amount of time, to allow the condition to clear. An RTP stream sender having sent a REFUSED SHOULD resume the stream through local considerations (see below) when the condition that caused the REFUSED is no longer true.

If the RTP stream sender receives several identical RESUME for an RTP stream that was already at least once responded with REFUSED and the condition causing REFUSED remains, those additional REFUSED should be sent with regular RTCP timing. A single REFUSED can respond to several identical RESUME requests.

A pausing RTP stream sender can apply local considerations and can resume a paused RTP stream at any time. If TMMBR 0 was used to pause the RTP stream, resumption is prevented by protocol, even if the RTP sender would like to resume due to local considerations. If TMMBR/ TMMBN signaling is used, if the RTP stream is paused due to local considerations (Section 5.4), and the RTP stream sender thus owns the TMMBN bounding set, the RTP stream can be resumed due to local considerations.

When resuming a paused stream, especially for media that makes use of temporal redundancy between samples such as video, it may not be appropriate to use such temporal dependency in the encoding between samples taken before the pause and at the time instant the stream is

resumed. Should such temporal dependency between media samples before and after the media was paused be used by the RTP stream sender, it requires the RTP stream receiver to have saved the samples from before the pause for successful continued decoding when resuming. The use of this temporal dependency of media samples from before the pause is left up to the RTP stream sender. If temporal dependency on samples from before the pause is not used when the RTP stream is resumed, the first encoded sample after the pause will not contain any temporal dependency on samples before the pause (for video it may be a so-called intra picture). If temporal dependency on samples from before the pause is used by the RTP stream sender when resuming, and if the RTP stream receiver did not save any sample from before the pause, the RTP stream receiver can use a FIR request [RFC5104] to explicitly ask for a sample without temporal dependency (for video a so-called intra picture), even at the same time as sending the RESUME.

## 5.6. TMMBR/TMMBN Considerations

As stated above, TMMBR/TMMBN may be used to provide pause and resume functionality for the point-to-point case. If the topology is not point-to-point, TMMBR/TMMBN cannot safely be used for pause or





resume. This use is expected to be mainly for interworking with implementations that don't support the messages defined in this specification (Section 8), but make use of TMMBR/TMMBN to achieve a similar effect.

This is a brief summary of what functionality is provided when using TMMBR/TMMBN:

TMMBR 0: Corresponds to PAUSE, without the requirement for any hold-off period to wait for RESUME before pausing the RTP stream.

TMMBR >0: Corresponds to RESUME when the RTP stream was previously paused with TMMBR 0. Since there is only a single RTP stream receiver, there is no need for the RTP stream sender to delay resuming the stream until after sending TMMBN >0, or to apply the hold-off period specified in [RFC5104] before increasing the bitrate from zero. The bitrate value used when resuming after pausing with TMMBR 0 is either according to known limitations, or based on starting a stream with the configured maximum for the

stream or session, for example given by b-parameter in SDP.

TMMBN 0: Corresponds to PAUSED when the RTP stream was paused with TMMBR 0, but may, just as PAUSED, also be used unsolicited. An unsolicited RTP stream pause based on local sender considerations uses the RTP stream's own SSRC as TMMBR restriction owner in the TMMBN message bounding set. Also corresponds to a REFUSED notification when a stream is requested to be resumed with TMMBR >0, thus resulting in the stream sender becoming the owner of the bounding set in the TMMBN message.

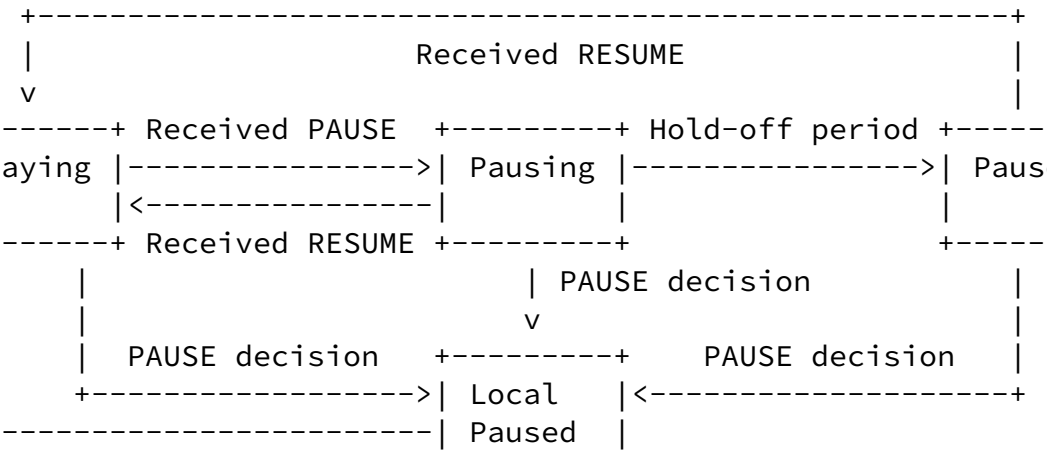
TMMBN >0: Cannot be used as REFUSED notification when a stream is requested to be paused with TMMBR 0, for reasons stated in Section 5.3.

# 6. Participant States

This document introduces three new states for a stream in an RTP sender, according to the figure and sub-sections below. Any references to PAUSE, PAUSED, RESUME and REFUSED in this section SHALL be taken to apply to the extent possible also when TMMBR/TMMBN are used (Section 5.6) for this functionality.







RESUME decision   +---------+

Figure 4: RTP Pause States in Sender

## 6.1. Playing State

This state is not new, but is the normal media sending state from [RFC3550]. When entering the state, the current PauseID MUST be incremented by one in modulo arithmetic. The RTP sequence number for the first packet sent after a pause SHALL be incremented by one compared to the highest RTP sequence number sent before the pause. The first RTP Time Stamp for the first packet sent after a pause SHOULD be set according to capture times at the source, meaning the RTP Time Stamp difference compared to before the pause reflects the time the RTP stream was paused.

## 6.2. Pausing State

In this state, the RTP stream sender has received at least one PAUSE message for the stream in question. The RTP stream sender SHALL wait during a hold-off period for the possible reception of RESUME messages for the RTP stream being paused before actually pausing RTP stream transmission. The hold-off period to wait SHALL be long enough to allow another RTP stream receiver to respond to the PAUSE with a RESUME, if it determines that it would not like to see the stream paused. This hold-off period is determined by the formula:

   2 * RTT + T_dither_max,

where RTT is the longest round trip known to the RTP stream sender and T_dither_max is defined in section 3.4 of [RFC4585]. The hold-off period MAY be set to 0 by some signaling (Section 9) means when it can be determined that there is only a single receiver, for example in point-to-point or some unicast situations.





If the RTP stream sender has set the hold-off period to 0 and receives information that it was an incorrect decision and that there are in fact several receivers of the stream, it MUST change the hold-off to instead be based on the above formula.

An RTP stream sender SHOULD use the following criteria to determine if there is only a single receiver, unless it has explicit and more reliable information:

- Observing only a single CNAME across all received SSRCs (CNAMEs for received CSRCs are insignificant), or

- If RTCP reporting groups [I-D.ietf-avtcore-rtp-multi-stream-optimisation] is used, observing only a single, endpoint external RTCP reporting group.

## 6.3. Paused State

An RTP stream is in paused state when the sender pauses its transmission after receiving at least one PAUSE message and the hold-off period has passed without receiving any RESUME message for that stream. Pausing transmission SHOULD only be done when reaching an appropriate place to pause in the stream, like a media boundary that avoids a media receiver to trigger repair or concealment actions.

When entering the state, the RTP stream sender SHALL send a PAUSED indication to all known RTP stream receivers, and SHALL also repeat PAUSED in the next two regular RTCP reports, as long as it is then still in paused state.

Pausing an RTP stream MUST NOT affect the sending of RTP keepalive [RFC6263][RFC5245] applicable to that RTP stream.

Following sub-sections discusses some potential issues when an RTP sender goes into paused state. These conditions are also valid if an RTP Translator is used in the communication. When an RTP Mixer implementing this specification is involved between the participants (which forwards the stream by marking the RTP data with its own SSRC), it SHALL be a responsibility of the Mixer to control sending PAUSE and RESUME requests to the sender. The below conditions also apply to the sender and receiver parts of the RTP Mixer, respectively.

### 6.3.1. RTCP BYE Message

When a participant leaves the RTP session, it sends an RTCP BYE message. In addition to the semantics described in section 6.3.4 and

6.3.7 of RTP [RFC3550], following two conditions MUST also be considered when an RTP participant sends an RTCP BYE message,

- If a paused sender sends an RTCP BYE message, receivers observing this SHALL NOT send further PAUSE or RESUME requests to it.

- Since a sender pauses its transmission on receiving the PAUSE requests from any receiver in a session, the sender MUST keep record of which receiver that caused the RTP stream to pause. If that receiver sends an RTCP BYE message observed by the sender, the sender SHALL resume the RTP stream. No receivers that were in the RTP session when the stream was paused objected that the stream was paused, but if there were so far undetected receivers added to the session during pause, those may not have learned about the existence of the paused stream, either because there was no PAUSED sent for the paused RTP stream or those receivers did not support PAUSED. Resuming the stream when the pausing party leaves the RTP session allows those potentially undetected receivers to learn that the stream exists.

### 6.3.2. SSRC Time-out

Section 6.3.5 in RTP [RFC3550] describes the SSRC time-out of an RTP participant. Every RTP participant maintains a sender and receiver list in a session. If a participant does not get any RTP or RTCP packets from some other participant for the last five RTCP reporting intervals it removes that participant from the receiver list. Any streams that were paused by that removed participant SSRC SHALL be resumed.

## 6.4. Local Paused State

This state can be entered at any time, based on local decision from the RTP stream sender. Pausing transmission SHOULD only be done when reaching an appropriate place to pause in the stream, like a media boundary that avoids a media receiver to trigger repair or concealment actions.

As with Paused State (Section 6.3), the RTP stream sender SHALL send a PAUSED indication to all known RTP stream receivers, when entering the state, unless the stream was already in paused state (Section 6.3). Such PAUSED indication SHALL be repeated a sufficient number of times to reach a high probability that the message is correctly delivered, stopping such repetition whenever leaving the state.

When using TMMBN 0 as PAUSED indication and when already in paused state, the actions when entering local paused state depends on the

bounding set overhead value in the received TMMBR 0 that caused the paused state, and the bounding set overhead value used in (the RTP stream sender's own) TMMBN 0:

TMMBN 0 overhead <= TMMBR 0 overhead: The RTP stream sender SHALL NOT send any new TMMBN 0 replacing that active (more restrictive) bounding set, even if entering local paused state.

TMMBN 0 overhead > TMMBR 0 overhead: The RTP stream sender SHALL send TMMBN 0 with itself in the TMMBN bounding set when entering local paused state.

The case above when using TMMBN 0 as PAUSED indication, being in local paused state, and having received a TMMBR 0 with a bounding set overhead value greater than the value the RTP stream sender would itself use in a TMMBN 0 requires further consideration and is for clarity henceforth referred to as "restricted local paused state".

As indicated in Figure 4, local paused state has higher precedence than paused state (Section 6.3) and RESUME messages alone cannot resume a paused RTP stream as long as the local decision still applies. An RTP stream sender in local paused state is responsible for leaving the state whenever the conditions that caused the decision to enter the state no longer apply.

If the RTP stream sender is in restricted local paused state, it cannot leave that state until the TMMBR 0 limit causing the state is removed by a TMMBR >0 (RESUME). If the RTP stream sender then needs to stay in local paused state due to local considerations, it MAY continue pausing the RTP stream by entering local paused state and MUST then act accordingly, including sending a TMMBN 0 with itself in the bounding set.

Pausing an RTP stream MUST NOT affect the sending of RTP keepalive [RFC6263][RFC5245] applicable to that RTP stream.

When leaving the local paused state, the stream state SHALL become Playing, regardless whether or not there were any RTP stream receivers that sent PAUSE for that stream during the local paused state, effectively clearing the RTP stream sender's memory for that stream.

## 7. Message Format

Section 6 of AVPF [RFC4585] defines three types of low-delay RTCP feedback messages, i.e. Transport layer, Payload-specific, and Application layer feedback messages. This document defines a new Transport layer feedback message, which is further sub-typed into





either a PAUSE request, a RESUME request, a PAUSED indication, or a REFUSED notification.

The Transport layer feedback messages are identified by having the RTCP payload type be RTPFB (205) as defined by AVPF [RFC4585]. This Transport layer feedback message, containing one or more of the sub-typed messages, is henceforth referred to as the PAUSE-RESUME message. The specific FCI format is identified by a Feedback Message Type (FMT) value in common packet header for feedback message defined in section 6.1 of AVPF [RFC4585]. The PAUSE-RESUME transport feedback message FCI is identified by FMT value = TBA1.

The Common Packet Format for Feedback Messages defined by AVPF [RFC4585] is:

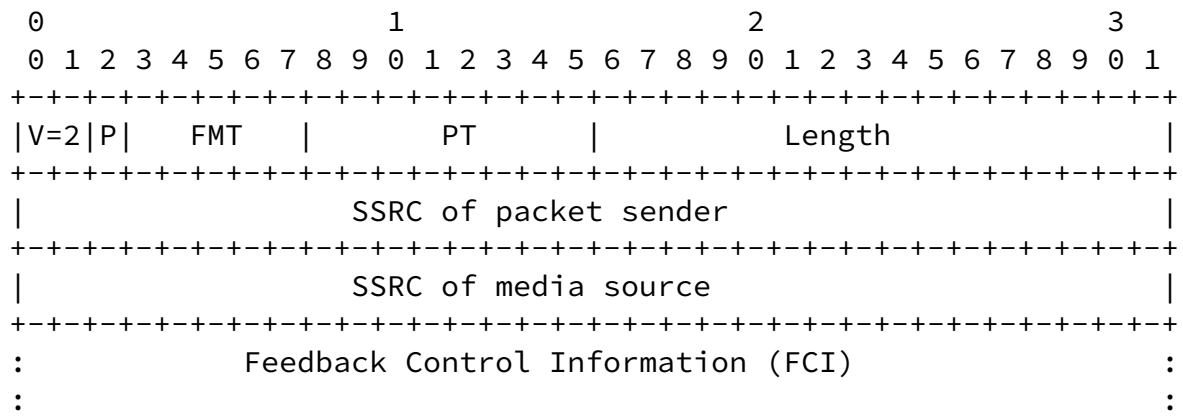



Figure 5: AVPF Common Feedback Message Packet Format

For the PAUSE-RESUME message defined in this memo, the following interpretations of the packet fields apply:

FMT: The FMT value identifying the PAUSE-RESUME FCI: TBA1

PT: Payload Type = 205 (RTPFB)

Length: As defined by AVPF, i.e. the length of this packet in 32-bit words minus one, including the header and any padding.

SSRC of packet sender: The SSRC of the RTP session participant sending the messages in the FCI. Note, for Endpoints that have multiple SSRCs in an RTP session, any of its SSRCs MAY be used to send any of the pause message types.

SSRC of media source: Not used, SHALL be set to 0. The FCI identifies the SSRC the message is targeted for.





The Feedback Control Information (FCI) field consists of one or more PAUSE, RESUME, PAUSED, REFUSED, or any future extension. These messages have the following FCI format:

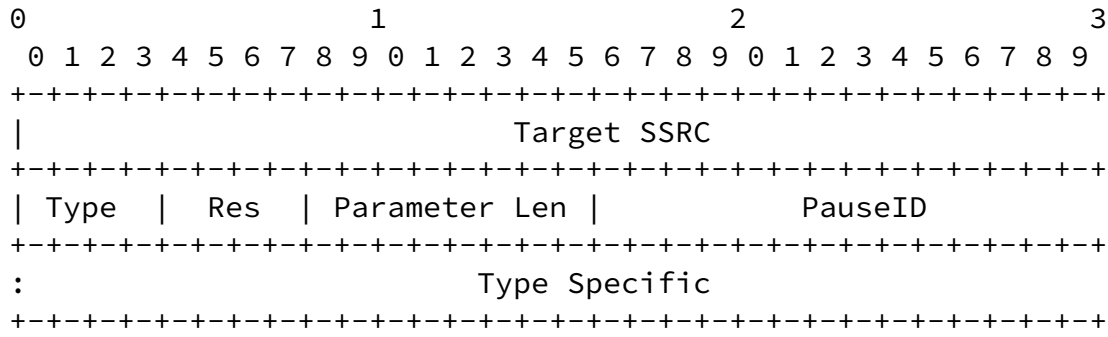



Figure 6: Syntax of FCI Entry in the PAUSE and RESUME message

The FCI fields have the following definitions:

Target SSRC (32 bits): For a PAUSE-RESUME message, this value is the SSRC that the request is intended for. For PAUSED, it MUST be the SSRC being paused. If pausing is the result of a PAUSE request, the value in PAUSED is effectively the same as Target SSRC in a related PAUSE request. For REFUSED, it MUST be the Target SSRC of the PAUSE or RESUME request that cannot change state. A CSRC MUST NOT be used as a target as the interpretation of such a request is unclear.

Type (4 bits): The pause feedback type. The values defined in this specification are as follows,

0: PAUSE request message.

1: RESUME request message.

2: PAUSED indication message.

3: REFUSED notification message.

4-15: Reserved for future use. FCI fields with these Type values SHALL be ignored on reception by receivers and MUST NOT be used by senders implementing this specification.

Res: (4 bits): Type specific reserved. SHALL be ignored by receivers implementing this specification and MUST be set to 0 by senders implementing this specification.





Parameter Len: (8 bits): Length of the Type Specific field in 32-bit words. MAY be 0.

PauseID (16 bits): Message sequence identification, as described in Section 5.2. SHALL be incremented by one modulo 2^16 for each new PAUSE message, unless the message is re-transmitted. The initial value SHOULD be 0. The PauseID is scoped by the Target SSRC, meaning that PAUSE, RESUME, and PAUSED messages therefore share the same PauseID space for a specific Target SSRC.

Type Specific: (variable): Defined per pause feedback Type. MAY be empty. A receiver implementing this specification MUST be able to skip and ignore any unknown Type Specific data, even for Type values defined in this specification.

## 8. Message Details

This section contains detailed explanations of each message defined in this specification. All transmissions of requests and indications are governed by the transmission rules as defined by Section 8.5.

Any references to PAUSE, PAUSED, RESUME and REFUSED in this section SHALL be taken to apply to the extent possible also when TMMBR/TMMBN are used (Section 5.6) for this functionality. TMMBR/TMMBN MAY be used instead of the messages defined in this specification when the effective topology is point-to-point. This use is expected to be mainly for interworking with implementations that don't support the messages defined in this specification, but make use of TMMBR/TMMBN to achieve a similar effect. If either sender or receiver learns that the topology is not point-to-point, TMMBR/TMMBN MUST NOT be used for pause/resume functionality. If the messages defined in this specification are supported in addition to TMMBR/TMMBN by all involved parties, pause/resume signaling MUST use messages from this specification. If the topology is not point-to-point and the messages defined in this specification are not supported, pause/resume functionality with TMMBR/TMMBN MUST NOT be used.

For the scope of this specification, a past PauseID (Section 5.2) is defined as having a value between and including (PauseID - 2^15) MOD 2^16 and (PauseID - 1) MOD 2^16, where "MOD" is the modulo operator. Similarly, a future PauseID is defined as having a value between and including (PauseID + 1) MOD 2^16 and (PauseID + 2^14) MOD 2^16. Future PauseID is intentionally not defined as the entire range that was not already defined as past PauseID. The remaining range of PauseID is simply "not current".





### 8.1. PAUSE

An RTP stream receiver MAY schedule PAUSE for transmission at any time.

PAUSE has no defined Type Specific parameters.

PauseID SHOULD be the current PauseID, as indicated by PAUSED (Section 8.2),REFUSED (Section 8.4), or implicitly determined by previously received PAUSE or RESUME (Section 8.3) requests. A randomly chosen PauseID MAY be used if it was not possible to retrieve current PauseID information, in which case the PAUSE will either succeed, or the current PauseID can be found in the returned REFUSED (Section 8.4).

It can be noted that as a result of what is described in Section 6.1, PauseID is incremented by one, in modulo arithmetic, for each PAUSE request that is not a retransmission, compared to what was used in the last PAUSED indication sent by the media sender. PauseID in the message is supposed to match current PauseID at the RTP stream sender.

If an RTP stream receiver that sent a PAUSE with a certain PauseID for a target SSRC receives a RESUME or a REFUSED with the same PauseID for the same target SSRC, it is RECOMMENDED that it refrains from scheduling further PAUSE requests for some appropriate time. This is because the RESUME indicates that there are other receivers that still wishes to receive the stream, and the REFUSED indicates that the RTP stream sender is currently not able to pause the stream. What is an appropriate time can vary from application to application and will also depend on the importance of achieving the bandwidth saving, but 2-5 regular RTCP intervals is expected to be appropriate.

If the targeted RTP stream does not pause, if no PAUSED indication with a future PauseID compared to the one used in PAUSE is received, and if no REFUSED with the current or a future PauseID is received within 2 * RTT + T_dither_max, the PAUSE MAY be scheduled for retransmission, using the same current PauseID. RTT is the observed round-trip to the RTP stream sender and T_dither_max is defined in section 3.4 of [RFC4585].

When an RTP stream sender in Playing State (Section 6.1) receives a PAUSE with the current PauseID, and unless local considerations currently makes it impossible to pause the stream, it SHALL enter Pausing State (Section 6.2) and act accordingly.





If an RTP stream sender receives a PAUSE with the current PauseID while in Pausing, Paused (Section 6.3) or Local Paused (Section 6.4) States, the received PAUSE SHALL be ignored.

## 8.2. PAUSED

The PAUSED indication, if supported, MUST be sent whenever entering

Paused State (Section 6.3) or Local Paused State (Section 6.4).

PauseID in the PAUSED message MUST contain the current PauseID that can be included in a subsequent RESUME (Section 8.3). For Local Paused State, this means that PauseID in the message is the current PauseID, just as if the RTP stream sender had sent a PAUSE to itself.

PAUSED SHALL contain a fixed-length 32-bit parameter at the start of the Type Specific field with the RTP extended highest sequence number (Section 6.4.1 of [RFC3550]) valid when the RTP stream was paused.

After having entered Paused or Local Paused State and thus having sent PAUSED once, PAUSED MUST also be included in (at least) the next two regular RTCP reports, given that the pause condition is then still effective.

PAUSED indications MAY be retransmitted, subject to transmission rules (Section 8.5), to increase the probability that the message reaches the receiver in a timely fashion. This can be especially important when entering Local Paused State. The number of repetitions to use could be tuned to observed loss rate and desired loss probability, for example based on RTCP reports received from the intended message target.

While remaining in Paused or Local Paused States, PAUSED MAY be included in all compound RTCP reports, as long as the negotiated RTCP bandwidth is not exceeded.

When in Paused or Local Paused States, whenever the RTP stream sender learns that there are Endpoints that did not previously receive the stream, for example by RTCP reports with an SSRC and a CNAME that was not previously seen in the RTP session, it is RECOMMENDED to send PAUSED at the earliest opportunity and also to include it in (at least) the next two regular RTCP reports, given that the pause condition is then still effective.

## 8.3. RESUME

An RTP stream receiver MAY schedule RESUME for transmission whenever it wishes to resume a paused stream, or to disapprove a stream from being paused.

PauseID SHOULD be the current PauseID, as indicated by PAUSED (Section 8.2) or implicitly determined by previously received PAUSE (Section 8.1) or RESUME requests. A randomly chosen PauseID MAY be used if it was not possible to retrieve current PauseID information, in which case the RESUME will either succeed, or the current PauseID can be found in a returned REFUSED (Section 8.4).

If an RTP stream receiver that sent a RESUME with a certain PauseID receives a REFUSED with the same PauseID, it is RECOMMENDED that it refrains from scheduling further RESUME requests for some appropriate time since the REFUSE indicates that it is currently not possible to resume the stream. What is an appropriate time can vary from application to application and will also depend on the importance of resuming the stream, but 1-2 regular RTCP intervals is expected to be appropriate.

RESUME requests MAY be retransmitted, subject to transmission rules (Section 8.5), to increase the probability that the message reaches the receiver in a timely fashion. The number of repetitions to use could be tuned to observed loss rate and desired loss probability, for example based on RTCP reports received from the intended message target. Such retransmission SHOULD stop as soon as RTP packets from the targeted stream are received, or a REFUSED with the current PauseID for the targeted RTP stream is received.

RESUME has no defined Type Specific parameters.

When an RTP stream sender in Pausing (Section 6.2), Paused (Section 6.3) or Local Paused State (Section 6.4) receives a RESUME with the current PauseID, and unless local considerations currently makes it impossible to resume the stream, it SHALL enter Playing State (Section 6.1) and act accordingly. If the RTP stream sender is incapable of honoring a RESUME request with the current PauseID, or if it receives a RESUME request with a PauseID that is not the current PauseID while in Paused or Pausing state, the RTP stream sender SHALL schedule a REFUSED message for transmission as specified below.

If an RTP stream sender in Playing State receives a RESUME containing either the current PauseID or a past PauseID, the received RESUME SHALL be ignored.

### 8.4. REFUSED

If an RTP stream sender receives a PAUSE (Section 8.1) or RESUME (Section 8.3) request containing the current PauseID, where the requested action cannot be fulfilled by the RTP stream sender due to some local consideration, it SHALL schedule transmission of a REFUSED

notification containing the current PauseID from the rejected request.

REFUSED has no defined Type Specific parameters.

If an RTP stream sender receives a PAUSE or RESUME request with a PauseID that is not the current PauseID, it SHALL schedule a REFUSED notification containing the current PauseID, except if the RTP stream sender is in Playing State and receives a RESUME with a past PauseID, in which case the RESUME SHALL be ignored.

If several PAUSE or RESUME that would render identical REFUSED notifications are received before the scheduled REFUSED is sent, duplicate REFUSED MUST NOT be scheduled for transmission. This effectively lets a single REFUSED respond to several ineffective PAUSE or RESUME requests.

An RTP stream receiver that sent a PAUSE or RESUME request and receives a REFUSED containing the same PauseID as in the request SHOULD refrain from sending an identical request for some appropriate time to allow the condition that caused REFUSED to clear. For PAUSE, an appropriate time is suggested in Section 8.1. For RESUME, an appropriate time is suggested in Section 8.3.

An RTP stream receiver that sent a PAUSE or RESUME request and receives a REFUSED containing a PauseID different from the request MAY schedule another request using the PauseID from the REFUSED notification.

## 8.5. Transmission Rules

The transmission of any RTCP feedback messages defined in this specification MUST follow the normal AVPF defined timing rules and depends on the session's mode of operation.

All messages defined in this specification, as well as TMMBR/TMMBN used for pause/resume purposes (Section 5.6), can use either Regular, Early or Immediate timings, but should make a trade-off between timely transmission (Section 4.1) and RTCP bandwidth consumption. This can be achieved by taking the following into consideration:

- It is recommended that PAUSE use Early or Immediate timing, except for retransmissions where RTCP bandwidth can motivate the use of

Regular timing.

- The first transmission of PAUSED for each (non-wrapped) PauseID is recommended to be sent with Immediate or Early timing, to stop unnecessary repetitions of PAUSE. It is recommended that





subsequent transmissions of PAUSED for that PauseID use Regular timing, to avoid that multiple PAUSE requests cause excessive PAUSED RTCP bandwidth.

- It is recommended that unsolicited PAUSED (sent when entering Local Paused State (Section 6.4)) always use Immediate or Early timing, until PAUSED for that PauseID is considered delivered at least once to all receivers of the paused RTP stream, to avoid that RTP stream receivers take unnecessary corrective action when the RTP stream is no longer received, after which it is recommended that PAUSE uses Regular timing (as for PAUSED triggered by PAUSE above).

- RESUME is often time-critical and it is recommended that it always uses Immediate or Early timing.

- The first transmission of REFUSED for each (non-wrapped) PauseID is recommended to be sent with Immediate or Early timing, to stop unnecessary repetitions of PAUSE or RESUME. It is recommended that subsequent REFUSED for that PauseID use Regular timing, to avoid that multiple unreasonable requests cause excessive REFUSED RTCP bandwidth.

## 9. Signaling

The capability of handling messages defined in this specification MAY be exchanged at a higher layer such as SDP. This document extends the rtcp-fb attribute defined in section 4 of AVPF [RFC4585] to include the request for pause and resume. This specification follows all the rules defined in AVPF [RFC4585] and CCM [RFC5104] for an rtcp-fb attribute relating to payload type in a session description.

This specification defines a new parameter "pause" to the "ccm" feedback value defined in CCM [RFC5104], representing the capability to understand the RTCP feedback message and all of the defined FCIs of PAUSE, RESUME, PAUSED and REFUSED.

Note: When TMMBR 0 / TMMBN 0 are used to implement pause and resume functionality (with the restrictions described in this specification), signaling rtcp-fb attribute with ccm tmmbr parameter is sufficient and no further signaling is necessary. There is however no guarantee that TMMBR/TMMBN implementations pre-dating this specification work exactly as described here when used with a bitrate value of 0.

The "pause" parameter has two optional attributes, "nowait" and "config":





- "nowait" indicates that the hold-off period defined in Section 6.2 can be set to 0, reducing the latency before the stream can be paused after receiving a PAUSE request. This condition occurs when there will only be a single receiver per direction in the RTP session, for example in point-to-point sessions. It is also possible to use in scenarios using unidirectional media. The conditions that allow "nowait" to be set (Section 6.2) also indicate that it would be possible to use CCM TMMBR/TMMBN as pause/resume signaling.

- "config" allows for partial implementation of this specification according to the different roles in the use cases section (Section 3), and takes a value that describes what sub-set is implemented:

  1 Full implementation of this specification. This is the default configuration. A missing config attribute MUST be treated equivalent to providing a config value of 1.

  2 The implementation intends to send PAUSE and RESUME requests for received RTP streams and is thus also capable of receiving PAUSED and REFUSED. It does not support receiving PAUSE and RESUME requests, but may pause sent RTP streams due to local considerations and then intends to send PAUSED for them.

  3 The implementation supports receiving PAUSE and RESUME requests targeted for RTP streams it sends. It will send PAUSED and REFUSED as needed. The node will not send any PAUSE and RESUME requests, but supports and desires receiving PAUSED if received

RTP streams are paused.

4 The implementation intends to send PAUSE and RESUME requests for received RTP streams and is thus also capable of receiving PAUSED and REFUSED. It cannot pause any RTP streams it sends, and thus does not support receiving PAUSE and RESUME requests, and also does not support sending PAUSED indications.

5 The implementation supports receiving PAUSE and RESUME requests targeted for RTP streams it sends. It will send PAUSED and REFUSED as needed. It does not support sending PAUSE and RESUME requests to pause received RTP streams, and also does not support receiving PAUSED indications.

6 The implementation supports sent and received RTP streams being paused due to local considerations, and thus supports sending and receiving PAUSED indications.





7 The implementation supports and desires to receive PAUSED indications for received RTP streams, but does not pause or send PAUSED indications for sent RTP streams. It does not support any other messages defined in this specification.

8 The implementation supports pausing sent RTP streams and sending PAUSED indications for them, but does not support receiving PAUSED indications for received RTP streams. It does not support any other messages defined in this specification.

All implementers of this specification are encouraged to include full support for all messages (config=1), but it is recognized that this is sometimes not meaningful for implementations operating in an environment where only parts of the functionality provided by this specification are needed. The above defined "config" functionality sub-sets provide a trade-off between completeness and the need for implementation interoperability, achieving at least a level of functionality corresponding to what is desired by the least capable party when used as specified here. Implementation of any other functionality sub-sets of this specification than the above defined is NOT RECOMMENDED.

When signaling a config value other than 1, an implementation MUST ignore non-supported messages on reception, and SHOULD omit sending messages not supported by the remote peer. One example where is can be motivated to send messages that some receivers do not support, is when there are multiple message receivers with different message support (different config values). That approach avoids that the least capable receiver limits the functionality provided to others. The below table summarizes per-message send and receive support for the different config attribute values ("X" indicating support and "-" indicating non-support):

| # | Send | | | | Receive | | | |
|---|---|---|---|---|---|---|---|---|
| | PAUSE | RESUME | PAUSED | REFUSED | PAUSE | RESUME | PAUSED | REFUSED |
| 1 | X | X | X | X | X | X | X | X |
| 2 | X | X | X | - | - | - | X | X |
| 3 | - | - | X | X | X | X | X | - |
| 4 | X | X | - | - | - | - | X | X |
| 5 | - | - | X | X | X | X | - | - |
| 6 | - | - | X | - | - | - | X | - |
| 7 | - | - | - | - | - | - | X | - |
| 8 | - | - | X | - | - | - | - | - |

Figure 7: Supported messages for different config values





In the above description of partial implementations, config=2 and 4 correspond to the RTP Mixer in the RTP Mixer to Media Sender use case (Section 3.2), and config=3 and 5 correspond to the Media Sender in that same use case. For that use case, it should be clear that an RTP Mixer implementing only config=3 or 5 will not provide a working solution. Similarly, for that use case, a Media Sender implementing only config=2 or 4 will not provide a working solution. Both the RTP Mixer and the Media Sender will of course work when implementing the full set of messages, corresponding to config=1.

A partial implementation is not suitable for pause / resume support between cascaded RTP Mixers, but would require support corresponding to config=1 between such RTP Mixers. This is because an RTP Mixer is then also Media Sender towards the other RTP Mixer, requiring support for the union of config=2 and 3 or config=4 and 5, which effectively

becomes config=1.

As can be seen from Figure 7 above, config=2 and 3 differ from config=4 and 5 only in that in the latter, the PAUSE / RESUME message sender (e.g. the RTP Mixer side) does not support local pause (Section 6.4) for any of its own streams and therefore also does not support sending PAUSED.

Partial implementations that only support local pause functionality can declare this capability through config=6-8.

Viable fallback rules between different config are described in Section 9.1 and Figure 9.

This is the resulting ABNF [RFC5234], extending existing ABNF in section 7.1 of CCM [RFC5104]:

```
rtcp-fb-ccm-param  =/ SP "pause" *(SP pause-attr)
pause-attr         = pause-config ; partial message support
                   / "nowait"     ; no hold-off
                   / byte-string  ; for future extensions
pause-config       = "config=" pause-config-value
pause-config-value = 1*2DIGIT
; byte-string as defined in RFC 4566
```

Figure 8: ABNF

An endpoint implementing this specification and using SDP to signal capability SHOULD indicate the new "pause" parameter with ccm signaling, but MAY instead use existing ccm tmmbr signaling [RFC5104] if the limitations in functionality when using TMMBR/TMMBN as described in this specification (Section 5.6) are considered





acceptable. In that case, no partial message support is possible. The messages from this specification (Section 8) SHOULD NOT be used towards receivers that did not declare capability to receive those messages.

The pause functionality can normally be expected to work independently of the payload type. However, there might exist situations where an endpoint needs to restrict or at least configure

the capabilities differently depending on the payload type carrying the media stream. Reasons for this might relate to capabilities to correctly handle media boundaries and avoid any pause or resume operation to occur where it would leave a receiver or decoder with no choice than to attempt to repair or discard the media received just prior to or at the point of resuming.

There MUST NOT be more than one "a=rtcp-fb" line with "pause" applicable to a single payload type in the SDP, unless the additional line uses "*" as payload type, in which case "*" SHALL be interpreted as applicable to all listed payload types that do not have an explicit "pause" specification. The "config" pause attribute MUST NOT appear more than once for each "pause" CCM parameter. The "nowait" pause attribute MUST NOT appear more than once for each "pause" CCM parameter.

## 9.1. Offer-Answer Use

An offerer implementing this specification needs to include "pause" CCM parameter with suitable configuration attribute ("config") in the SDP, according to what messages it intends to send and desires to receive in the session.

In SDP offer/answer, the "config" attribute and its message directions are interpreted based on the agent providing the SDP. The offerer is described in an offer, and the answerer is described in an answer.

An answerer receiving an offer with a "pause" CCM line and a config attribute with a certain value, describing a certain capability to send and receive messages, MAY change the config attribute value in the answer to another configuration. The permitted answers are listed in the below table.





SDP Offer config value | Permitted SDP Answer config values

| | |
|---|---|
| 1 | 1, 2, 3, 4, 5, 6, 7, 8 |
| 2 | 3, 4, 5, 6, 7, 8 |
| 3 | 2, 4, 5, 6, 7, 8 |
| 4 | 5, 6, 7, 8 |
| 5 | 4, 6, 7, 8 |
| 6 | 6, 7, 8 |
| 7 | 8 |
| 8 | 7 |

Figure 9: Config values in Offer/Answer

An offer or answer omitting the config attribute, MUST be interpreted as equivalent to config=1. Implementations of this specification MUST NOT use any other config values than the ones defined above in an offer or answer, and MUST remove the "pause" CCM line in the answer when receiving an offer with a config value it does not understand. In all cases the answerer MAY also completely remove any "pause" CCM line to indicate that it does not understand or desire to use any pause functionality for the affected payload types.

If the offerer believes that itself and the intended answerer are likely the only Endpoints in the RTP session, it MAY include the "nowait" attribute on the "pause" line in the offer. If an answerer receives the "nowait" attribute on the "pause" line in the SDP, and if it has information that the offerer and itself are not the only Endpoints in the RTP session, it MUST NOT include any "nowait" attribute on its "pause" line in the SDP answer. The answerer MUST NOT add "nowait" on the "pause" line in the answer unless it is present on the "pause" line in the offer. If both offer and answer contained a "nowait" parameter, then the hold-off period is configured to 0 at both offerer and answerer.

Unknown pause attributes MUST be ignored in the offer and MUST then be omitted from the answer.

If both "pause" and "tmmbr" are present in the offer, both MAY be included also in the answer, in which case TMMBR/TMMBN MUST NOT be used for pause/resume purposes (with a bitrate value of 0), to avoid signaling ambiguity.

## 9.2. Declarative Use

In declarative use, the SDP is used to configure the node receiving the SDP. This has implications on the interpretation of the SDP signaling extensions defined in this specification.

First, the "config" attribute and its message directions are interpreted based on the node receiving the SDP, and describes the RECOMMENDED level of operation. If the joining client does not support the indicated config value, some RTP session stream optimizations may not be possible in that some RTP streams will not be paused by the joining client, and/or the joining client may not be able to resume and receive wanted streams because they are paused.

Second, the "nowait" parameter, if included, is followed as specified. It is the responsibility of the declarative SDP sender to determine if a configured node will participate in a session that will be point to point, based on the usage. For example, a conference client being configured for an any source multicast session using SAP [RFC2974] will not be in a point to point session, thus "nowait" cannot be included. An RTSP [RFC2326] client receiving a declarative SDP may very well be in a point to point session, although it is highly doubtful that an RTSP client would need to support this specification, considering the inherent PAUSE support in RTSP.

Unknown pause attributes MUST be ignored.

If both "pause" and "tmmbr" are present in the SDP, TMMBR/TMMBN MUST NOT be used for pause/resume purposes (with a bitrate value of 0), to avoid signaling ambiguity.

## 10. Examples

The following examples shows use of PAUSE and RESUME messages, including use of offer-answer:

1. Offer-Answer

2. Point-to-Point session

3. Point-to-Multipoint using Mixer

4. Point-to-Multipoint using Relay

### 10.1. Offer-Answer

The below figures contains an example how to show support for pausing and resuming the streams, as well as indicating whether or not the hold-off period can be set to 0.

```
v=0
o=alice 3203093520 3203093520 IN IP4 alice.example.com
s=Pausing Media
t=0 0
c=IN IP4 alice.example.com
m=audio 49170 RTP/AVPF 98 99
a=rtpmap:98 G719/48000
a=rtpmap:99 PCMA/8000
a=rtcp-fb:* ccm pause nowait
```

Figure 10: SDP Offer With Pause and Resume Capability

The offerer supports all of the messages defined in this specification, leaving out the optional config attribute. The offerer also believes that it will be the sole receiver of the answerer's stream as well as that the answerer will be the sole receiver of the offerer's stream and thus includes the "nowait" sub-parameter for the "pause" parameter.

This is the SDP answer:

```
v=0
o=bob 293847192 293847192 IN IP4 bob.example.com
s=-
t=0 0
c=IN IP4 bob.example.com
m=audio 49202 RTP/AVPF 98
a=rtpmap:98 G719/48000
a=rtcp-fb:98 ccm pause config=2
```

Figure 11: SDP Answer With Pause and Resume Capability

The answerer will not allow its sent streams to be paused or resumed and thus restricts the answer to indicate config=2. It also supports pausing its own RTP streams due to local considerations, which is why config=2 is chosen rather than config=4. The answerer somehow knows that it will not be a point-to-point RTP session and has therefore

removed "nowait" from the "pause" line, meaning that the offerer must use a non-zero hold-off period when being requested to pause the stream.

When using TMMBR 0 / TMMBN 0 to achieve pause and resume functionality, there are no differences in SDP compared to CCM [RFC5104] and therefore no such examples are included here.





## 10.2. Point-to-Point Session

This is the most basic scenario, which involves two participants, each acting as a sender and/or receiver. Any RTP data receiver sends PAUSE or RESUME messages to the sender, which pauses or resumes transmission accordingly. The hold-off period before pausing a stream is 0.

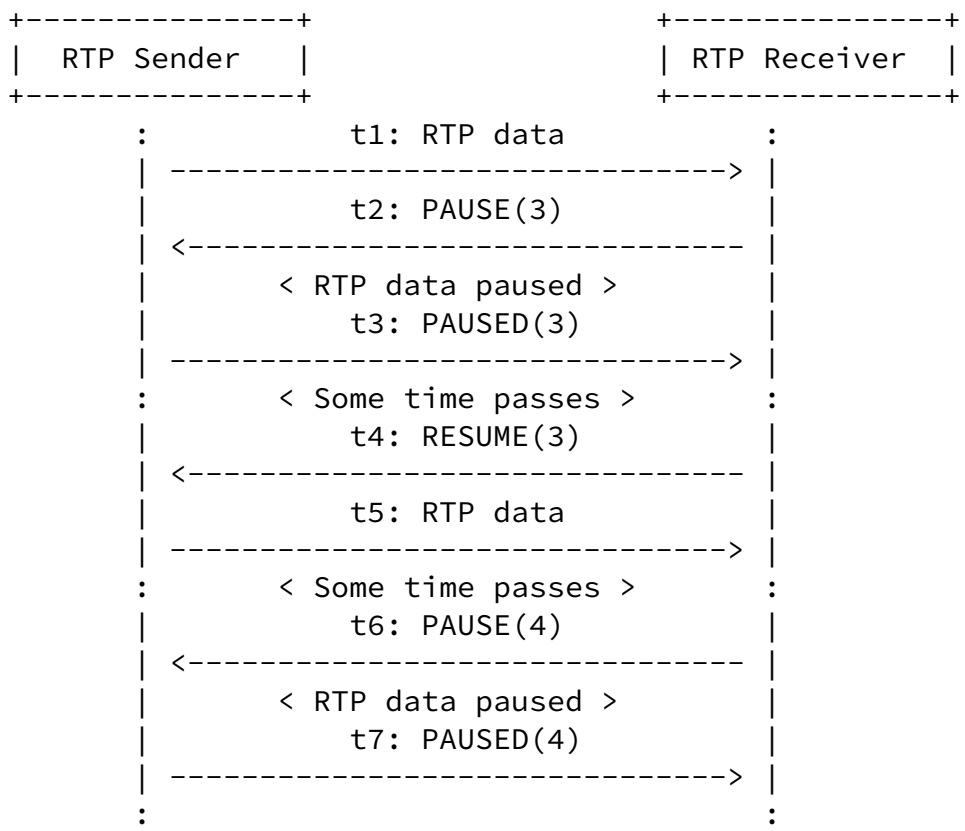


Figure 12: Pause and Resume Operation in Point-to-Point

Figure 12 shows the basic pause and resume operation in Point-to-Point scenario. At time t1, an RTP sender sends data to a receiver. At time t2, the RTP receiver requests the sender to pause the stream, using PauseID 3 (which it knew since before in this example). The sender pauses the data and replies with a PAUSED containing the same PauseID. Some time later (at time t4) the receiver requests the sender to resume, which resumes its transmission. The next PAUSE, sent at time t6, contains an updated PauseID (4), with a corresponding PAUSED being sent at time t7.





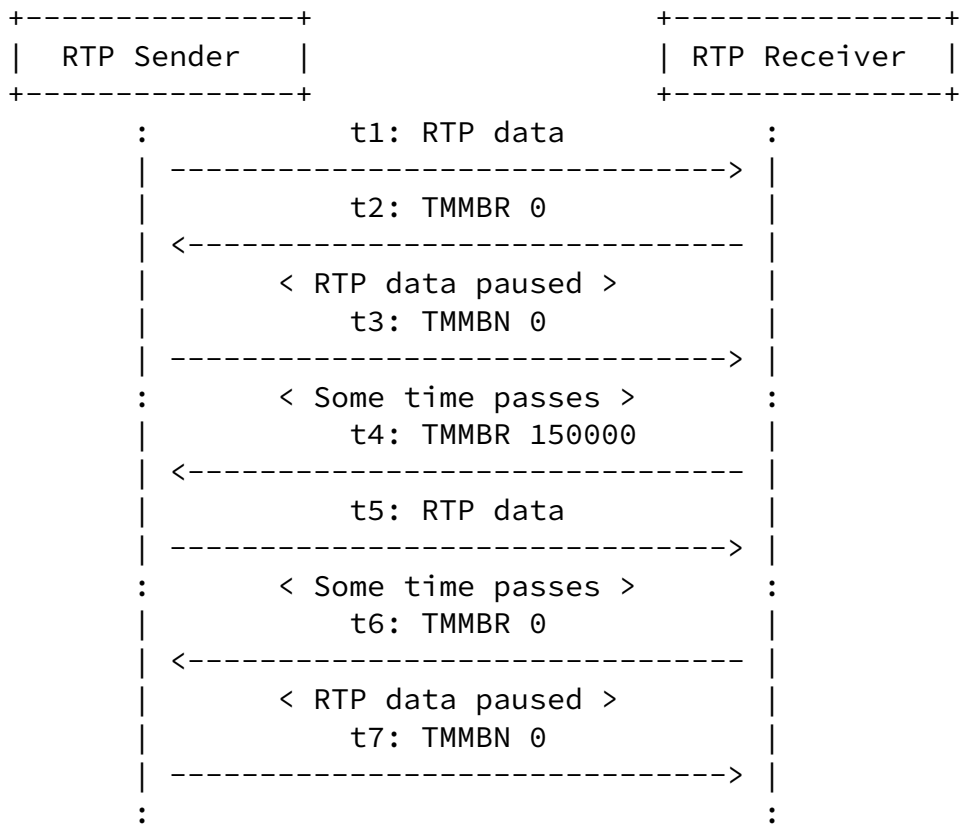
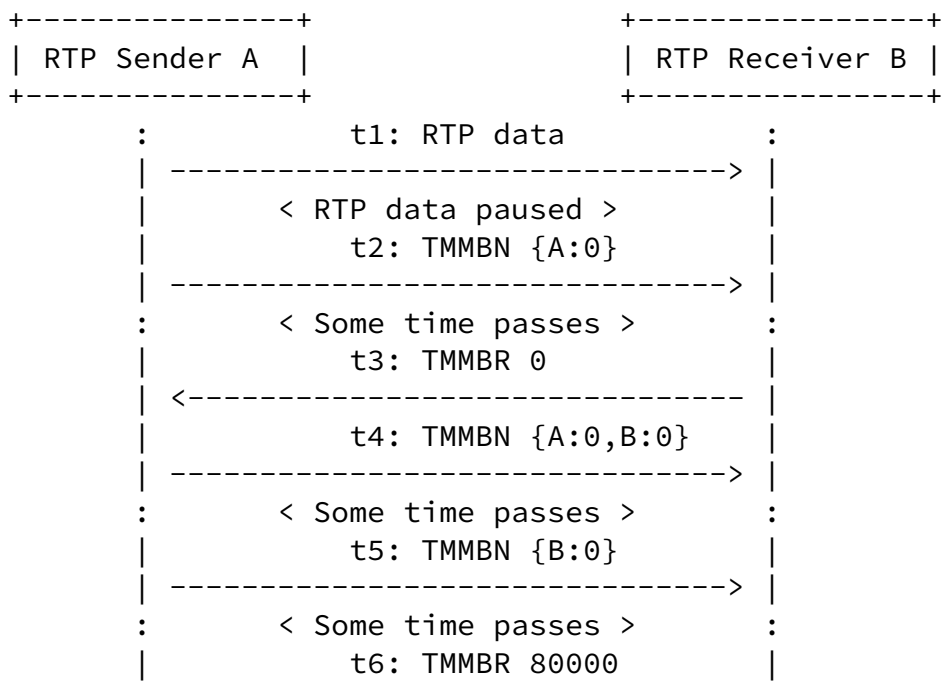



Figure 13: TMMBR Pause and Resume in Point-to-Point

Figure 13 describes the same point-to-point scenario as above, but using TMMBR/TMMBN signaling.







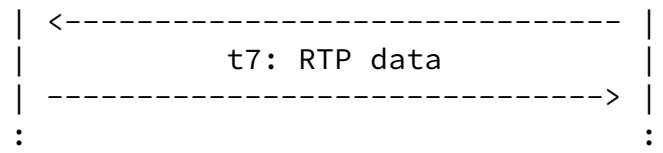
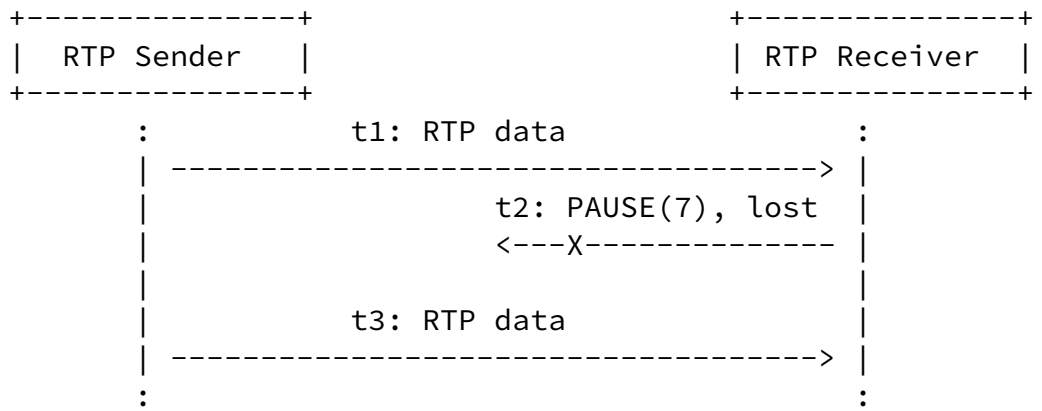



Figure 14: Unsolicited PAUSED using TMMBN

Figure 14 describes the case when an RTP stream sender (A) chooses to pause an RTP stream due to local considerations. Both the RTP stream sender (A) and the RTP stream receiver (B) use TMMBR/TMMBN signaling for pause/resume purposes. A decides to pause the RTP stream at time t2 and uses TMMBN 0 to signal PAUSED, including itself in the TMMBN bounding set. At time t3, despite the fact that the RTP stream is still paused, B decides that it is no longer interested to receive the RTP stream and signals PAUSE by sending a TMMBR 0. As a result of that, the bounding set now contains both A and B, and A sends out a new TMMBN reflecting that. After a while, at time t5, the local considerations that caused A to pause the RTP stream no longer apply, causing it to remove itself from the bounding set and to send a new TMMBN indicating this. At time t6, B decides that it is now interested to receive the RTP stream again and signals RESUME by sending a TMMBR containing a bitrate value greater than 0, causing A to resume sending RTP data.







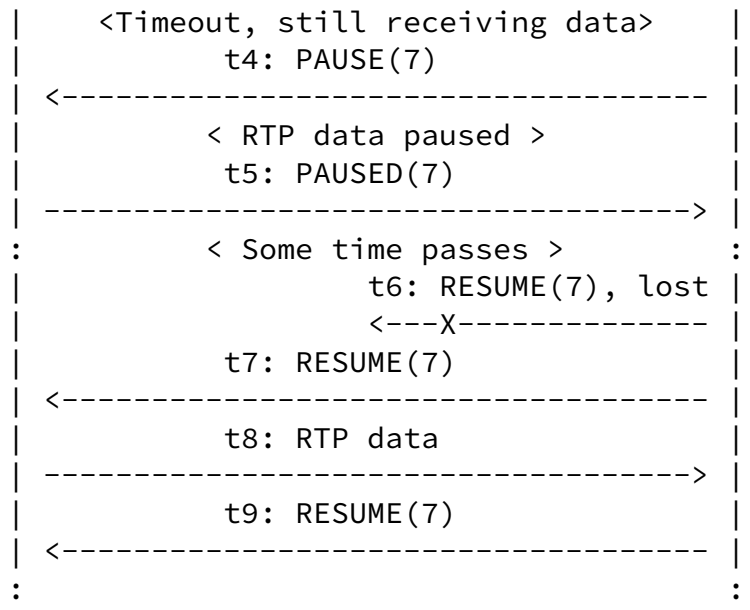



Figure 15: Pause and Resume Operation With Messages Lost

Figure 15 describes what happens if a PAUSE message from an RTP stream receiver does not reach the RTP stream sender. After sending a PAUSE message, the RTP stream receiver waits for a time-out to detect if the RTP stream sender has paused the data transmission or has sent PAUSED indication according to the rules discussed in Section 6.3. As the PAUSE message is lost on the way (at time t2), RTP data continues to reach to the RTP stream receiver. When the timer expires, the RTP stream receiver schedules a retransmission of the PAUSE message, which is sent at time t4. If the PAUSE message now reaches the RTP stream sender, it pauses the RTP stream and replies with PAUSED.

At time t6, the RTP stream receiver wishes to resume the stream again and sends a RESUME, which is lost. This does not cause any severe effect, since there is no requirement to wait until further RESUME are sent and another RESUME are sent already at time t7, which now reaches the RTP stream sender that consequently resumes the stream at





time t8. The time interval between t6 and t7 can vary, but may for example be one RTCP feedback transmission interval as determined by the AVPF rules.

The RTP stream receiver did not realize that the RTP stream was resumed in time to stop yet another scheduled RESUME from being sent at time t9. This is however harmless since RESUME contains a past PauseID and will be ignored by the RTP stream sender. It will also not cause any unwanted resume even if the stream was paused again based on a PAUSE from some other receiver before receiving the RESUME, since the current PauseID was updated compared to the one in the stray RESUME, which contains a past PauseID and will be ignored by the RTP stream sender.

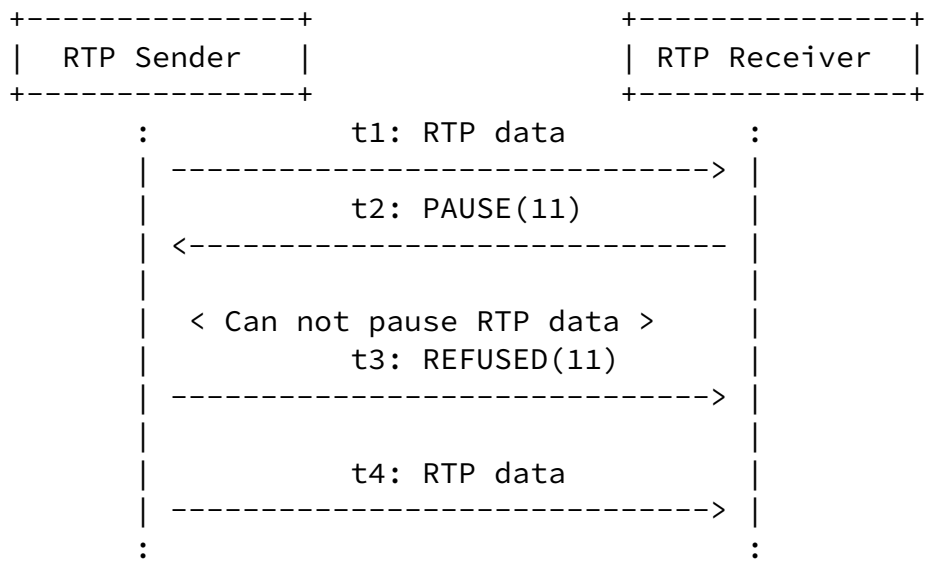



Figure 16: Pause Request is Refused in Point-to-Point

In Figure 16, the receiver requests to pause the sender, which refuses to pause due to some consideration local to the sender and responds with a REFUSED message.

## 10.3. Point-to-Multipoint using Mixer

An RTP Mixer is an intermediate node connecting different transport-level clouds. The Mixer receives streams from different RTP sources, selects or combines them based on the application's needs and forwards the generated stream(s) to the destination. The Mixer typically puts its' own SSRC(s) in RTP data packets instead of the original source(s).

The Mixer keeps track of all the streams delivered to the Mixer and how they are currently used. In this example, it selects the video

stream to deliver to the receiver R based on the voice activity of the RTP stream senders. The video stream will be delivered to R using M's SSRC and with an CSRC indicating the original source.

Note that PauseID is not of any significance for the example and is therefore omitted in the description.

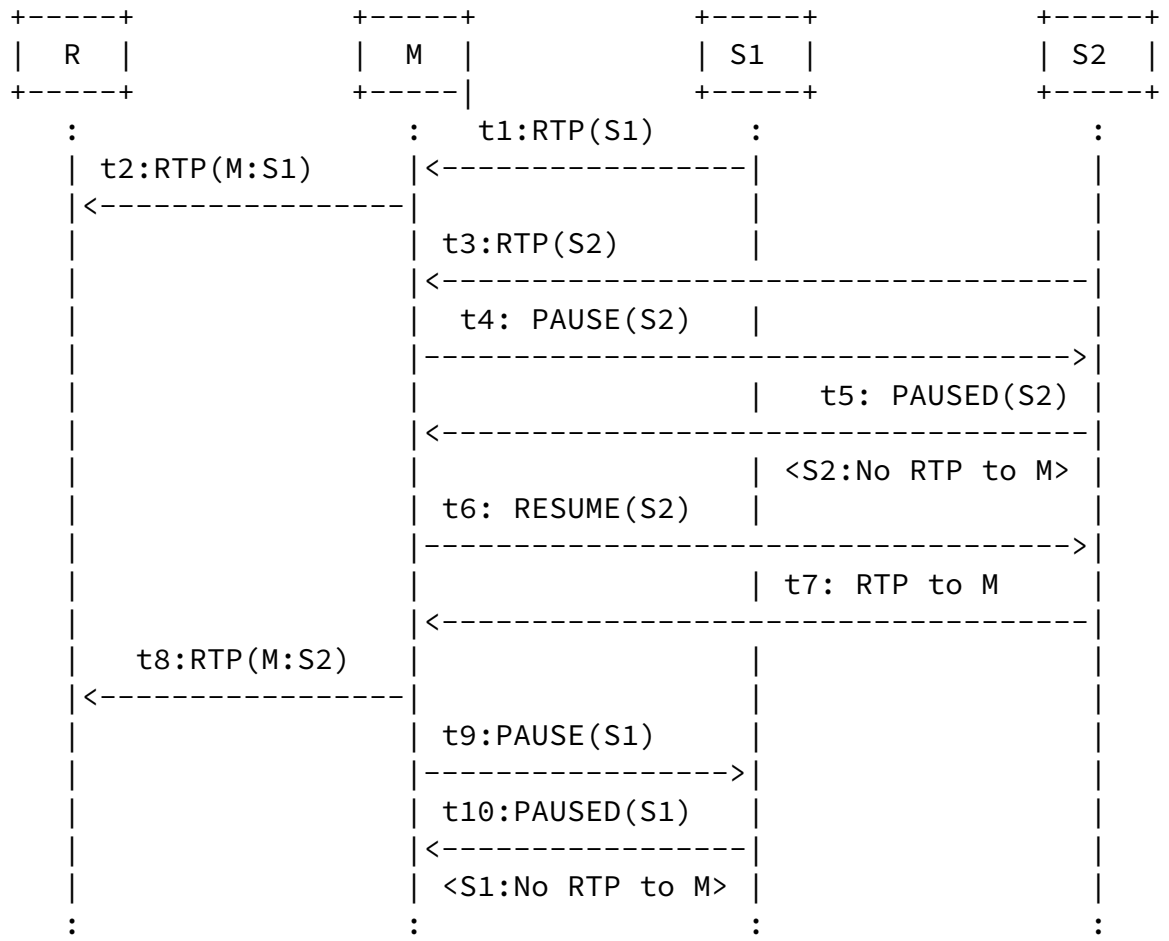



Figure 17: Pause and Resume Operation for a Voice Activated Mixer

The session starts at t1 with S1 being the most active speaker and thus being selected as the single video stream to be delivered to R (t2) using the Mixer SSRC but with S1 as CSRC (indicated after the colon in the figure). Then S2 joins the session at t3 and starts delivering an RTP stream to the Mixer. As S2 has less voice activity then S1, the Mixer decides to pause S2 at t4 by sending S2 a PAUSE request. At t5, S2 acknowledges with a PAUSED and at the same instant stops delivering RTP to the Mixer. At t6, the user at S2 starts speaking and becomes the most active speaker and the Mixer decides to switch the video stream to S2, and therefore quickly sends a RESUME request to S2. At t7, S2 has received the RESUME request

and acts on it by resuming RTP stream delivery to M. When the RTP





stream from t7 arrives at the Mixer, it switches this RTP stream into its SSRC (M) at t8 and changes the CSRC to S2. As S1 now becomes unused, the Mixer issues a PAUSE request to S1 at t9, which is acknowledged at t10 with a PAUSED and the RTP stream from S1 stops being delivered.

## 10.4. Point-to-Multipoint using Translator

A transport Relay in an RTP session forwards the message from one peer to all the others. Unlike Mixer, the Relay does not mix the streams or change the SSRC of the messages or RTP media. These examples are to show that the messages defined in this specification can be safely used also in a transport Relay case. The parentheses in the figures contains (Target SSRC, PauseID) information for the messages defined in this specification.

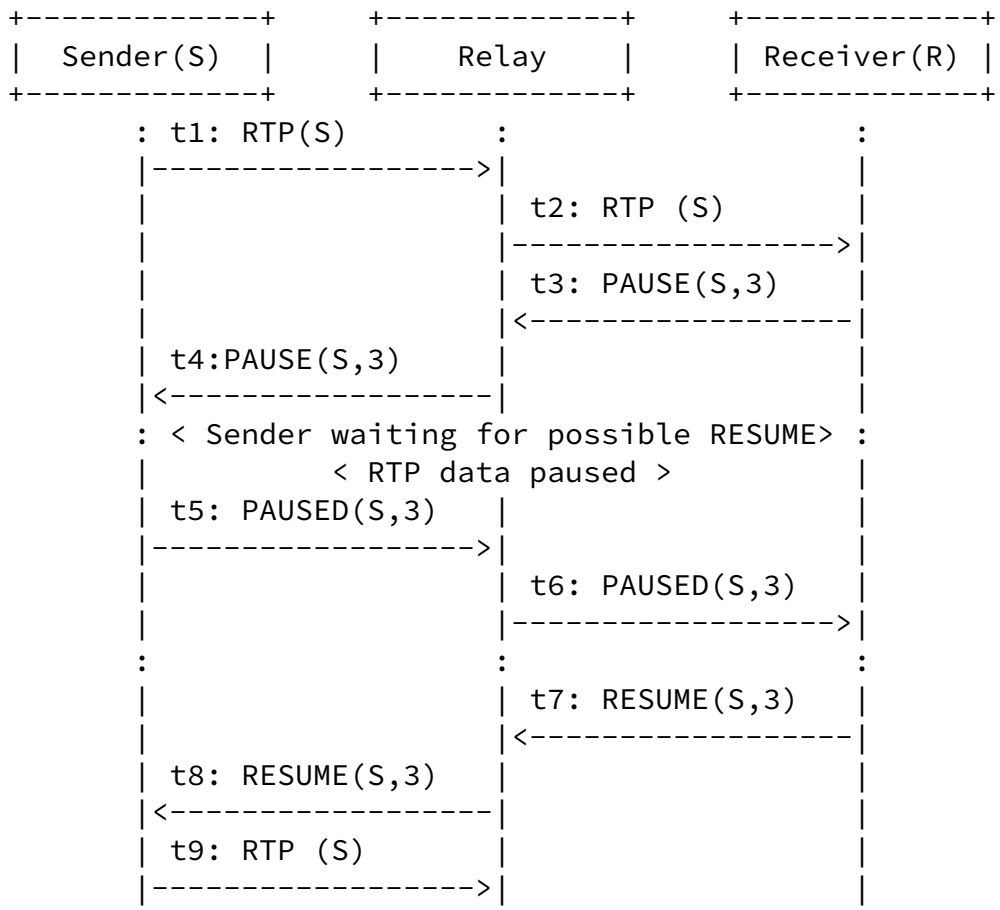
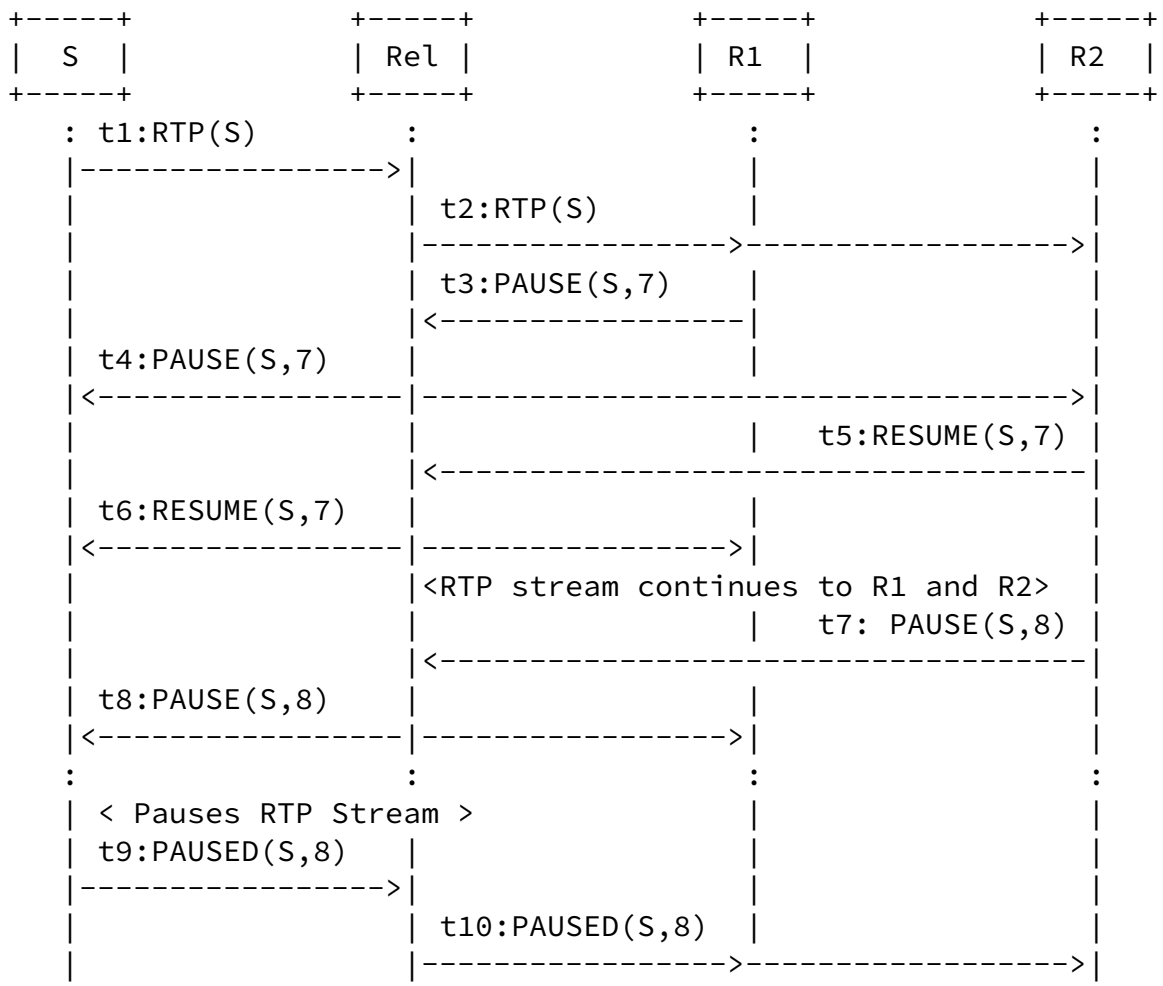



Figure 18: Pause and Resume Operation Between Two Participants Using a Relay





Figure 18 describes how a Relay can help the receiver in pausing and resuming the sender. The sender S sends RTP data to the receiver R through Relay, which just forwards the data without modifying the SSRCs. The receiver sends a PAUSE request to the sender, which in this example knows that there may be more receivers of the stream and waits a non-zero hold-off period to see if there is any other receiver that wants to receive the data, does not receive any disapproving RESUME, hence pauses itself and replies with PAUSED. Similarly the receiver resumes the sender by sending RESUME request through Relay. Since this describes only a single pause and resume operation for a single RTP stream sender, all messages uses a single PauseID, in this example 3.


S
Rel
R1
R2
t1:RTP(S)
t2:RTP(S)
t3:PAUSE(S,7)
t4:PAUSE(S,7)
t5:RESUME(S,7)
t6:RESUME(S,7)
<RTP stream continues to R1 and R2>
t7: PAUSE(S,8)
t8:PAUSE(S,8)
< Pauses RTP Stream >
t9:PAUSED(S,8)
t10:PAUSED(S,8)

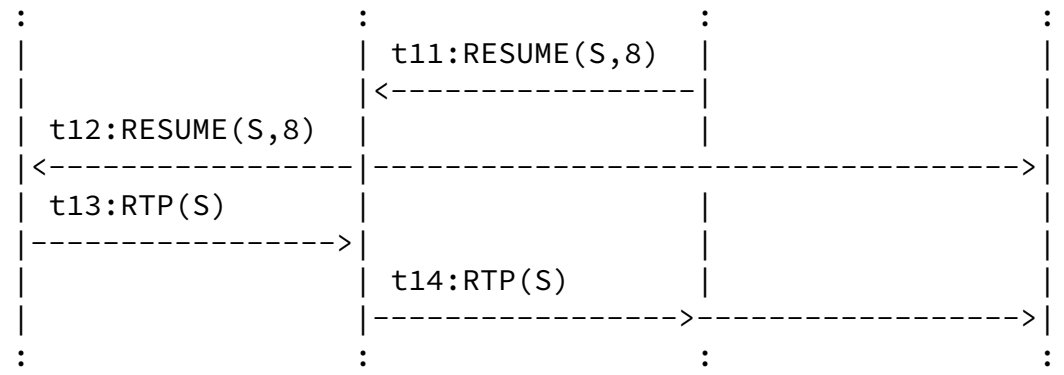



Figure 19: Pause and Resume Operation Between One Sender and Two Receivers Through Relay

Figure 19 explains the pause and resume operations when a transport Relay is involved between a sender and two receivers in an RTP session. Each message exchange is represented by the time it happens. At time t1, Sender (S) starts sending an RTP stream to the Relay, which is forwarded to R1 and R2 through the Relay, Rel. R1 and R2 receives RTP data from Relay at t2. At this point, both R1 and R2





will send RTCP Receiver Reports to S informing that they receive S's stream.

After some time (at t3), R1 chooses to pause the stream. On receiving the PAUSE request from R1 at t4, S knows that there are at least one receiver that may still want to receive the data and uses a non-zero hold-off period to wait for possible RESUME messages. R2 did also receive the PAUSE request at time t4 and since it still wants to receive the stream, it sends a RESUME for it at time t5, which is forwarded to the sender S by the Relay. The sender S sees the RESUME at time t6 and continues to send data to Relay which forwards to both R1 and R2. At t7, the receiver R2 chooses to pause the stream by sending a PAUSE request with an updated PauseID. The sender S still knows that there are more than one receiver (R1 and R2) that may want the stream and again waits a non-zero hold-off period, after which and not having received any disapproving RESUME, it concludes that the stream must be paused. S now stops sending the stream and replies with PAUSED to R1 and R2. When any of the receivers (R1 or R2) chooses to resume the stream from S, in this

example R1, it sends a RESUME request to the sender (also seen by R2). The RTP sender immediately resumes the stream.

Consider also an RTP session which includes one or more receivers, paused sender(s), and a Relay. Further assume that a new participant joins the session, which is not aware of the paused sender(s). On receiving knowledge about the newly joined participant, e.g. any RTP traffic or RTCP report (i.e. either SR or RR) from the newly joined participant, the paused sender(s) immediately sends PAUSED indications for the paused streams since there is now a receiver in the session that did not pause the sender(s) and may want to receive the streams. Having this information, the newly joined participant has the same possibility as any other participant to resume the paused streams.

## 11. IANA Considerations

This specification requests the following registrations from IANA:

1. A new value for media stream pause / resume to be registered with IANA in the "FMT Values for RTPFB Payload Types" registry located at the time of publication at: http://www.iana.org/assignments/rtp-parameters/rtp-parameters.xhtml#rtp-parameters-8

   Value: TBA1

   Name: PAUSE-RESUME

   Long Name: Media Pause / Resume





   Reference: This RFC

2. A new value "pause" to be registered with IANA in the "Codec Control Messages" registry located at the time of publication at: http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xhtml#sdp-parameters-19

   Value Name: pause

   Long Name: Media Pause / Resume

   Usable with: ccm

Reference: This RFC

## 12. Security Considerations

This document extends the CCM [RFC5104] and defines new messages, i.e. PAUSE, RESUME, PAUSED, and REFUSED. The exchange of these new messages have some security implications, which need to be addressed by the user.

The messages defined in this specification can have substantial impact on the perceived media quality if used in a malicious way. First of all, there is the risk for Denial of Service (DoS) on any RTP session that uses the PAUSE-RESUME functionality. By injecting one or more PAUSE requests into the RTP session, an attacker can potentially prevent any media from flowing, especially when the hold-off period is zero. The injection of PAUSE messages is quite simple, requiring knowledge of the SSRC and the PauseID. This information is visible to an on-path attacker unless RTCP messages are encrypted. Even off-path attacks are possible as signalling messages often carry the SSRC value, while the 16-bit PauseID have to be guessed or tried. The way of protecting the RTP session from these injections is to perform source authentication combined with message integrity, to prevent other than intended session participants from sending these messages. The security solution should provide replay protection, otherwise an attacker could for sessions that are long lived enough to wrap the PauseID replay old messages at the appropriate time to influence the media sender state. There exist several different choices for securing RTP sessions to prevent this type of attack. SRTP is the most common, but also other methods exist as discussed in "Options for Securing RTP Sessions" [RFC7201].

Most of the methods for securing RTP however do not provide source authentication of each individual participant in a multi-party use case. In case one of the session participants is malicious, it can wreck significant havoc within the RTP session and similarly cause a





DoS on the RTP session from within. That damage can also be attempted to be obfuscated by having the attacker impersonate other endpoints within the session. These attacks can be mitigated by using a solution that provides true source authentication of all participants' RTCP packets. However, that has other implications.

For multi-party sessions including a middlebox, that middlebox is RECOMMENDED to perform checks on all forwarded RTCP packets so that each participant only uses its set of SSRCs, to prevent the attacker utilizing another participant's SSRCs. An attacker that can send a PAUSE request without it reaching any other participant than the media sender can have its request being unopposed. This is mitigated in multi-party topologies that ensure that requests are seen by all or most of the RTP session participants, enabling these participants to send RESUME. In topologies with middleboxes that consume and process PAUSE requests, the middlebox can also mitigate such behavior as it will commonly not generate or forward a PAUSE message if it knows of another participant having use for the media stream.

The above text has been focused on using the PAUSE message as the tool for malicious impact on the RTP session. That is because of the greater impact from denying users access to RTP media streams. In contrast, if an attacker attempts to use RESUME in a malicious purpose, it will result in that the media streams are delivered. However, such an attack basically prevents the use the Pause and Resume functionality. Thus, potentially forcing a reduction of the media quality due to limitation in available resources, like bandwidth that must be shared.

The session establishment signalling is also a potential venue of attack, as that can be used to prevent the enabling of Pause and Resume functionality by modifying the signalling messages. The above mitigation of attacks based on source authentication also requires the signalling system to securely handle identities, and assert that only the intended identities are allowed into the RTP session and provided the relevant security contexts.

## 13. Contributors

Daniel Grondal contributed in the creation and writing of early versions of this specification. Christian Groves contributed significantly to the SDP config attribute and its use in Offer/ Answer.

## 14. Acknowledgements

Daniel Grondal made valuable contributions during the initial versions of this draft. The authors would also like to thank Emil Ivov, Christian Groves, David Madnelberg, Meral Shirazipour, Spencer

Dawkins, Bernard Aboba, and Ben Campbell, who provided valuable review comments.

## 15. References

### 15.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <http://www.rfc-editor.org/info/rfc3264>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <http://www.rfc-editor.org/info/rfc4566>.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <http://www.rfc-editor.org/info/rfc4585>.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, DOI 10.17487/RFC5104, February 2008, <http://www.rfc-editor.org/info/rfc5104>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, DOI 10.17487/RFC5245, April 2010, <http://www.rfc-editor.org/info/rfc5245>.

[RFC6263] Marjou, X. and A. Sollaud, "Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows", RFC 6263, DOI 10.17487/RFC6263, June 2011, <http://www.rfc-editor.org/info/rfc6263>.

## 15.2. Informative References

[I-D.ietf-avtcore-rtp-multi-stream-optimisation] Lennox, J., Westerlund, M., Wu, W., and C. Perkins, "Sending Multiple Media Streams in a Single RTP Session: Grouping RTCP Reception Statistics and Other Feedback", draft-ietf-avtcore-rtp-multi-stream-optimisation-06 (work in progress), July 2015.

[I-D.ietf-avtcore-rtp-topologies-update] Westerlund, M. and S. Wenger, "RTP Topologies", draft-ietf-avtcore-rtp-topologies-update-10 (work in progress), July 2015.

[I-D.ietf-avtext-rtp-grouping-taxonomy] Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", draft-ietf-avtext-rtp-grouping-taxonomy-08 (work in progress), July 2015.

[I-D.ietf-mmusic-sdp-simulcast] Burman, B., Westerlund, M., Nandakumar, S., and M. Zanaty, "Using Simulcast in SDP and RTP Sessions", draft-ietf-mmusic-sdp-simulcast-01 (work in progress), July 2015.

[RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, DOI 10.17487/RFC2326, April 1998, <http://www.rfc-editor.org/info/rfc2326>.

[RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, DOI 10.17487/RFC2974, October 2000, <http://www.rfc-editor.org/info/rfc2974>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,

A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <http://www.rfc-editor.org/info/rfc3261>.





[RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, DOI 10.17487/RFC6190, May 2011, <http://www.rfc-editor.org/info/rfc6190>.

[RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, DOI 10.17487/RFC7201, April 2014, <http://www.rfc-editor.org/info/rfc7201>.

[RFC7478] Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use Cases and Requirements", RFC 7478, DOI 10.17487/RFC7478, March 2015, <http://www.rfc-editor.org/info/rfc7478>.

## Appendix A. Changes From Earlier Versions

NOTE TO RFC EDITOR: Please remove this section prior to publication.

### A.1. Modifications Between Version -08 and -09

Changes based on IETF last call and IESG comments.

- Expanded some acronyms on first usage.
- Clarified why this document updated RFC 5104.
- Removed some unused abbreviations.
- Clarified when TMMBR/TMMBN is expected to be used in Section 5.6.
- Mandate using SHALL repetition of PAUSED indications in Section 6.4.
- Removed paragraph in Section 8.4 on PauseID and REFUSED as being redundant.

- Transmission rules in Section 8.5 was reworded and stoped using RFC 2119 terms, instead uses recommendations and lists motivations.

- In Section 9 (Signalling) it was clarified that one SHOULD only send messages supported by receivers, but provide example of cases when one may still send messages not supported by every receiver of that message. Also clarified that TMMBR/TMMBN for pause is monolithic in capability.

- Security consideration was updated to consider replay attacks.





- Security consideration was updated to describe the mitigation against malicous RTP session participants in multi-party cases that seeing all messages provide.

### A.2. Modifications Between Version -07 and -08

Changes based on IESG AD Evaluation.

- Moved the mentioning of RTCWEB RFC7478 API requirements out from 3.1 to section 3, adding a couple of clarifying sentences.

- Highlighted that the use case in section 3.4 deals with a different direction of the pause request than the previous use cases.

- Added text on partial capability and interoperability to section 5.1.

- Added an overview explanation of PauseID as a new section 5.2, and moved a few sentences on PauseID from other 5.x sections in there.

- Changed all occurrences of "available" and "valid" PauseID to the more clear "current" PauseID, and re-phrased sentences involving that to become more clear.

- Changed all occurrences of "smaller" and "larger" PauseID to "past" and "future", respectively, to better align with "current".

- Removed an incorrect sentence in 5.2 about when it is not feasible to send repeated PAUSE.

- Changed a few capitalized words that could be taken as normative text from section 5, which is intended to be a non-normative description.

- Added some explanatory text on why RTP stream is resumed when the stream receiver that paused the stream leaves the RTP session to last bullet in 6.3.1.

- Added caption to Figure 5.

- Moved the detailed description on what PauseID ranges are defined as "past" and "future" before section 8.1, instead of having it in section 8.1, and added a comment on the "not current" part of the value range.





- Added text in section 8.1 on appropriate time to wait between sending PAUSE, when the first PAUSE was rejected by a RESUME or a REFUSED.

- Added text in section 8.3 on appropriate time to wait between sending RESUME, when the first RESUME was rejected by a REFUSED.

- Added text in section 8.4 on time to wait before sending the REFUSED request again, referencing sections 8.1 and 8.3.

- Added a couple of paragraphs in section 9 on partial capability and interoperability, including a description on when different config values are expected to be useful, and when they are not.

- Added arrows in Figure 19 to highlight that the Relay sends out all received messages to all receivers, not only the first PAUSE message.

- Changed references to RFC3264 and RFC4566 to be normative.

- Updated ietf-rtcweb-use-cases-and-requirements reference to be

RFC7478.

- Editorial improvements and clarifications.

## A.3. Modifications Between Version -06 and -07

- Completely rewrote the Security Consideration section.

- Aligned text such that REFUSED is always referred to as a notification, not indication.

- Added and changed text in several places, clarifying the case when TMMBR/TMMBN bounding set overhead value matters, related to whether local RTP stream sender or remote RTP stream receiver owns the TMMBR 0 restriction, and the consequences this has on pause/resume logic.

- Moved text on when to stop media stream transmission from when receiving PAUSE and entering pausing state, to when entering paused or local paused states.

- Added text on how to determine if there is a single receiver or not, aligned with what is specified in draft-ietf-avtcore-multi-stream, adding a reference to draft-ietf-avtcore-multi-stream-optimisation to be able to use a single RTCP reporting group as one criteria.





- Added clarifying text on repeating PAUSED and RESUME messages only as long as remaining in the relevant state.

- Clarified that it is the RTP stream sender's responsibility to leave local paused state when the condition causing that state is no longer true.

- Added text to better allow for extensions to this specification, since there is already some text on extensions.

- Corrected and amended ABNF to make CCM pause parameters order-independent, allow for a larger config pause attribute value range, and added corresponding text to handle that additional flexibility.

- Added SDP rules on how to handle unknown pause attribute values.

- Clarified how to handle an SDP with both "ccm pause" and "ccm tmmbr".

- Changed from "Translator" to "Relay" in examples, to make it clearer in relation to the updated topologies draft.

- Editorial improvements.

### A.4. Modifications Between Version -05 and -06

- Clarified in Message Details section for PAUSED that retransmission of the message can be used to increase the probability that the message reaches the receiver in a timely fashion, and also added text that says the number of repetitions can be tuned to observed loss rate and the desired loss probability. Also removed Editor's notes on potential ACK for unsolicited PAUSED, since the issue is solved by the above.

- In the same section as above, added that PAUSED may be included in all compound RTCP reports, as long as the negotiated RTCP bandwidth is not exceeded.

- In Message Details section for RESUME, added text on retransmission similar to the one mentioned for PAUSED above. Also included text that says such retransmission SHOULD stop as soon as RTP packets or a REFUSED with a valid PauseID from the targeted stream are received.

- Changed simulcast reference, since that draft was moved from AVTCORE to MMUSIC and made WG draft.





- Changed End Point to Endpoint to reflect change in RTP Grouping Taxonomy draft.

- Editorial improvements.

### A.5. Modifications Between Version -04 and -05

- Added text in sections 4.1, 4.6, 6.4 and 8.5 on retransmission and timing of unsolicited PAUSED, to improve the message timeliness and probability of reception.

### A.6. Modifications Between Version -03 and -04

- Change of Copyright boilerplate

### A.7. Modifications Between Version -02 and -03

- Changed the section on SDP signaling to be more explicit and clear in what is supported, replacing the 'paused' parameter and the 'dir' attribute with a 'config' parameter that can take a value, and an explicit listing of what each value means.

- Added a sentence in section on paused state (Section 6.3) that pause must not affect RTP keepalive.

- Replaced REFUSE message name with REFUSED throughout, to better indicate that it is not a command but a notification.

- Added text in a few places, clarifying that PAUSED message may be used unsolicited due to RTP sender local considerations, and also clarified the interaction between this usage and an RTP stream receiver pausing the stream. Also added an example describing this case.

- Clarified that when TMMBN 0 is used as PAUSED message, and when sent unsolicited due to RTP sender local considerations, the TMMBN message includes the RTP stream sender itself as part of the bounding set.

- Clarified that there is no reply to a PAUSED indication.

- Improved the IANA section.

- Editorial improvements.

## A.8. Modifications Between Version -01 and -02

- Replaced most text on relation with other signaling technologies in previous section 5 with a single, summarizing paragraph, as discussed at IETF 90 in Toronto, and placed it as the last sub-section of section 4 (design considerations).

- Removed unused references.

## A.9. Modifications Between Version -00 and -01

- Corrected text in section 6.5 and 6.2 to indicate that a PAUSE signaled via TMMBR 0 cannot be REFUSED using TMMBN > 0

- Improved alignment with RTP Taxonomy draft, including the change of Packet Stream to RTP Stream

- Editorial improvements

## Authors' Addresses


Bo Burman
Ericsson
Kistavagen 25
SE - 164 80 Kista
Sweden

Email: bo.burman@ericsson.com

Azam Akram
Ericsson
Farogatan 6
SE - 164 80 Kista
Sweden

Phone: +46107142658
Email: muhammad.azam.akram@ericsson.com
URI: www.ericsson.com

Roni Even
Huawei Technologies
Tel Aviv
Israel

Email: roni.even@mail01.huawei.com

Magnus Westerlund
Ericsson
Farogatan 6
SE- 164 80 Kista
Sweden

Phone: +46107148287
Email: magnus.westerlund@ericsson.com