FS - Stream

Content

History

Version	Date	Adjustments
A	2006-10-06	First revision. (MBEME)
B	2008-08-05	Updated regarding ptime, maxptime, RTCP and synchronization (Mystique). Changes for Brutele. Updated regarding RTCP for Mystique. Updated for Video Fast Update over RTCP feedback, FE27. Updated with information about SDP bandwidth modifiers. Updated with silence detection. (ESTBERG/EJAKGAR/EMATSHA)

1 Introduction

This document specifies the functionality of the Stream component.

1.1 How to read this document

This document uses the standards for streaming media such as RTP/RTCP and variations thereof and it is highly recommended that these standards are understood before reading this document.

2 Definitions

2.1 Media

In this context Media consists of an audio or video part that can be streamed on a media stream to and from the stream endpoint.

2.2 DTMF

DTMF events are transported over RTP and are handled differently from media (audio and video). DTMF is received by the inbound stream. The default behavior of the inbound stream is to transform RTP DTMF into control token events and issue them through the global event handling.

2.3 Media Stream

A stream is a media stream that consists of zero, one or two RTP streams. It also has a direction; inbound or outbound.

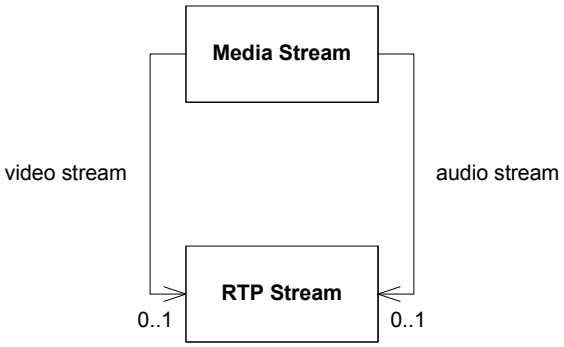


Figure 1 Media stream and RTP stream relationship

One of the RTP streams are for the audio part and the other is for the video part of the media stream. Note that the RTP streams are an internal object and are mainly shown to ease the understanding of the functionality of the Stream component.

2.4 Joined Media Stream

A media streams can be joined. The purpose is to transfer data from an inbound RTP stream to an outbound RTP stream. The direction of join is always from input to output.

During join an inbound media stream can be configured to ignore DTMF (no DTMF events are issued) and to transfer DTMF (RTP) to the outbound media stream.

3 Function Requirements (Commercial)

The following commercial requirements have been identified.

- Support for multiple codecs:
  - G.711 PCMU
  - G.711 PCMA
  - H.263 Baseline@Level 0
  - AMR (3GPP Release 6)
- Support for multiple file formats:
  - WAV
  - MOV
  - 3GPP (TS 26.244 version 6.3.0 Release 6)
- Support RTCP for synchronization and feedback.
- Support for SDP Bandwidth Modifiers as specified in [9].

4 Function Specification (Design Related)

4.1 Introduction

The Stream component handles the life cycle of the RTP streams, streaming and synchronization of video streams. It also handles transfer of media from one inbound to one outbound stream, i.e. joined streams.

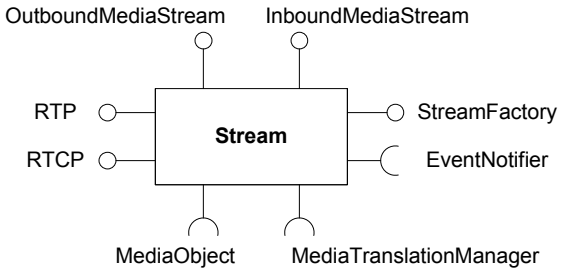


Figure 2 The exported and imported interfaces

4.2 Exported interfaces

4.2.1 RTP and RTCP interfaces (RTP, RTCP)

Through this interface RTP streams can be received and sent. For a full description of the RTP/RTCP protocols see [1].

4.2.2 The stream factory interface (StreamFactory)

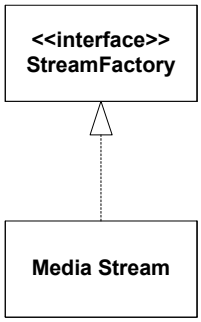


Figure 3 The StreamFactory interface

This interface is to retrieve a new inbound or outbound media stream.

4.2.2.1 Functions

4.2.2.1.1 Get inbound media stream ()

Get inbound media stream ()

Returns a new inbound media stream.

4.2.2.1.2 Get outbound media stream ()

Get outbound media stream ()

Returns a new outbound media stream.

4.2.3 The inbound media stream interface (IInboundMediaStream)

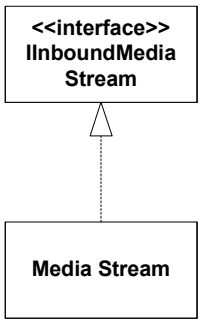


Figure 4 The InboundMediaStream interface

This interface is to create and control inbound media stream.

4.2.3.1 Functions

4.2.3.1.1 Create

Create (media mime types)

This function creates and connects a stream to its endpoint. The media mime types describe the media of the stream.

Create (video fast updater, media mime types)

This function creates and connects a stream to its endpoint. The media mime types describe the media of the stream. If a video fast updater instance is given, this instance is used to request an I-frame at the beginning of a record-operation.

Create (rtp payloads)

This function creates and connects a stream to its endpoint. The rtp payloads describe the media of the stream.

Create (video fast updater, rtp payloads)

This function creates and connects a stream to its endpoint. The rtp payloads describe the media of the stream. If a video fast updater instance is given, this instance is used to request an I-frame at the beginning of a record-operation.

4.2.3.1.2 Delete

Delete ()

This function closes the endpoints and makes the stream unavailable for recording.

4.2.3.1.3 Join

Join (outbound media stream)

Joins this inbound stream with the outbound stream. This causes the media that comes on the inbound stream to be redirected to the outbound stream.

Join (handle DTMF, outbound media stream, transfer DTMF)

Joins this inbound stream with the outbound stream. This causes the media that comes on the inbound stream to be redirected to the outbound stream.

4.2.3.1.4 Unjoin

Unjoin (outbound media stream)

This function removes the redirection of the inbound stream to the given outbound stream. When unjoin is done all data already received from the endpoint on the inbound stream must be sent to the outbound stream and then the outbound streams *unjoined* function is called.

4.2.3.1.5 Record

Record (requestId, play media object, recording properties)

The inbound stream records the inbound RTP data and stores it in the provided record media object as media data. The recording properties indicate how the recording is to be done with regards to for example max record time in milliseconds.

The recording will stop either when the max recording time is exceeded, when silence detected (if enabled) or when stop is called.

Record (requestId, play media object, outbound media stream, record media object, recording properties)

The inbound stream records the inbound RTP data and stores it in the provided record media object as media data. The recording properties indicate how the recording is to be done with regards to for example max record time in milliseconds.

If a play media object is included, it is played on the given outbound media stream until the recording starts.

The recording will stop either when the max recording time is exceeded, when silence detected (if enabled) or when stop is called.

4.2.3.1.6 Stop

Stop (requestId)

The recording matching the received requestId is stopped.

4.2.3.1.7 Set skew

Set skew (skewmethod, milliseconds)

This function sets the skew between the audio and video stream for a stream object. The skew is the number of milliseconds the audio is ahead of the video.

The default skew is defined in the configuration.

4.2.3.1.8 setCNAME

setCNAME(name)

Sets the canonical name used as identifier in the RTP packets sent by this stream. See RFC 1550 for more information about CNAME.

4.2.3.1.9 Get lost RTP packets

Returns statistics for the stream regarding lost RTP packets since the stream was created.

getCumulativePacketLost()

Current number of lost packets.

getFractionLost()

The loss fraction is defined as the number of packets lost, divided by the number expected. It is expressed as the integer part after multiplying the loss fraction by 256. Possible values are 0-255. If duplicates exist and the number of received packets are greater than the number expected, the loss fraction is set to zero.

Example: If 1/4 of the packets were lost, the loss fraction would be 1/4*256=64.

4.2.3.1.10 Get stream properties

getPTime()

Retrieves the ptime for this inbound stream.

getMaxPTime()

Retrieves the maxptime for this inbound stream.

4.2.3.2 Parameter Type Description

4.2.3.2.1 RequestId

RequestId identifies a specific request and is included in all events triggered by the request.

4.2.3.2.2 skewmethod

The skewmethod can be one of the following:

- Local
- RTCP
- Local and RTCP

For a description on skewmethod, please see the skewmethod configuration chapter 4.6.20.2.

4.2.3.2.3 Media mime types

The codec and bit rate describing the properties of the media stream.

4.2.3.2.4 Rtp payloads

This is a list of rtp payloads describing the properties of the media stream such as codec, bitrate, etc.

4.2.3.2.5 Media object

This is the actual media that is recorded on this stream (see [5] for details on the media object).

4.2.3.2.6 Recording properties

The recording properties[1] inform the media stream how to record a stream.

1. Max wait time before recording has started.
2. Max recording time, i.e. how long a record can be at max.
3. If the record should wait for recording to finish before returning.
4. Max silence duration before recording is aborted.

4.2.3.2.7 Handle DTMF

It is possible to specify if the received DTMF events should generate control token events or not.

4.2.3.2.8 Transfer DTMF

It is possible to specify if incoming DTMF should be transferred or not (from input to output RTP during join).

4.2.4 The outbound media stream interface (IOutboundMediaStream)

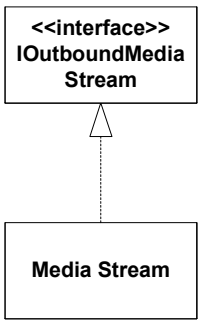


Figure 5 The OutboundMediaStream interface

This interface is to create and control outbound media streams.

[1] *Note: "minimum recording duration" is handled altogether by the VVA, this is a difference compared to the functionality in MVAS, where the Flow Manager transfers this property to Hlink.*

4.2.4.1 Functions

4.2.4.1.1 Create

Create (Supported payloads, connection properties, RTCPFeedback, pointer to InboundMediaStream)

This function creates and connects a stream to its endpoint. The supported payloads describe the media of the stream and the connection properties the endpoint location and transport mechanism such as IP, port, transport, frame size etc.

In order to support sending RTCP SR with RR block(s) included, the pointer to the inbound media stream is provided to facilitate fetching of statistical data from inbound stream.

Also retrieves the RTCP feedback to use in the stream, like Video Fast Update over RTCP.

4.2.4.1.2 Delete

Delete ()

This function closes the endpoints and makes the stream unavailable for playing.

4.2.4.1.3 Send

Send (control tokens)

This function sends the tokens on the RTP stream; DTMF is only supported so far as DTMF RTP payload.

4.2.4.1.4 Play

Play (requestId, media object, cursor, play option)

Plays the media object on the stream using the play option and starts from the cursor. If the media object is the same as the previously stopped media object any internal data such as packetization etc. is reused.

For a video media object the media stream must locate the closest intra frame to start streaming from.

Play on a joined outbound stream will fail.

Play (requestId, array of media objects, cursor, play option)

Plays the media objects on the stream using the play option and starts from the cursor.

For a video media object the media stream must locate the closest intra frame to start streaming from.

Play on a joined outbound stream will fail.

4.2.4.1.5 Stop

Stop (requestId)

Stops the ongoing play of a media object (i.e. the play matches the received requestId) and returns the cursor. The media stream retains all internal data of the media object in case a new play with the same media object is issued.

4.2.4.1.6 Unjoined

Unjoined ()

This function tells the outbound stream that the joined inbound stream has been removed and no more data will arrive from the inbound stream.

4.2.4.1.7 Joined

Joined ()

This function tells the outbound stream that an inbound stream has joined this outbound stream and data will arrive from the inbound stream.

4.2.4.1.8 isJoined

isJoined ()

Tells if this stream is currently joined.

4.2.4.1.9 Set skew

Set skew (method, milliseconds)

This function sets the skew between the audio and video stream for a stream object. The skew is the number of milliseconds the audio is ahead of the video. For a negative number the video will be ahead of audio.

4.2.4.1.10 Cancel

Cancel ()

Any ongoing streaming can be canceled with this function.

4.2.4.1.11 Translation failed

translationFailed(cause)

A translation has failed. When this occurs, the ongoing play-operation will end with a PlayFailed-event.

4.2.4.1.12 Translation Done

translationDone ()

Notifies the outbound stream that no more translated data will be sent from a joined stream.

translationDone(media object)

Sends a translated media object to the endpoint.

4.2.4.1.13 Setters

setCNAME(name)

Sets the canonical name used as identifier in the RTP packets sent by this stream. See RFC 1550 for more information about CNAME.

setEventDispatcher(event dispatcher)

This method sets the event dispatcher to be used by this stream.

4.2.4.2 Parameter Type Description

4.2.4.2.1 RequestId

RequestId identifies a specific request and is included in all events triggered by the request.

4.2.4.2.2 Supported payloads

Contains properties (codec and bit rate etc.) for all supported media of the media stream.

4.2.4.2.3 Connection properties

The connection properties of the stream, such as location, port number and transport mechanism and frame size.

4.2.4.2.4 Media object

The actual media that is played on this stream (see [5] for details on the media object).

4.2.4.2.5 Play option

The option on how the media is to be played, it can be either **wait for media** or **do not wait for media**. See the Stream play scenarios.

4.2.4.2.6 Cursor

The position in a media object where the streaming is currently taking place or where to start playing. The cursor is in milliseconds.

4.2.4.2.7 Control token

A control token to send over the stream. The control token consists of:

- The actual token
- Volume
- Duration

4.2.4.2.8 Cause

This parameter describes the cause of the play failure.

4.2.4.2.9 RTCPFeedback

This parameter describes the RTCP feedback functions to use.

4.3 Imported Interfaces

4.3.1 EventNotifier

The EventNotifier is used for issuing events.

4.3.2 MediaTranslationManager

The MediaTranslationManager is used for translating text media to audio media.

4.3.3 MediaObject

The MediaObject is used as data container for both play and record.

4.4 Events

4.4.1 Generated

4.4.1.1 Play finished

This event is sent when the play has finished, i.e. when the media object has been streamed to the other endpoint. For a media object containing video, this event is sent when both the voice and the video streams have completed streaming.

If a play is ongoing when the stream is deleted, the play is stopped and this event is sent.

This event includes the requestId that was specified when the request was issued.

Please note that play finished is issued when the requested play has finished. Hence, when requesting play of a sequence of media objects the play finished is issued when all media objects have been played.

4.4.1.2 Play failed

This event is sent if play fails and includes the requestId that was specified when the request was issued. A message describing the cause of failure is also included.

4.4.1.3 Record finished

This event is sent when the record has finished and includes the requestId that was specified when the request was issued and the length of the recording. It also contains a cause code for finishing the recording. It can be either timeout when recording has not started, max record time reached, max silence duration reached, stopped, max record time is exceeded or stream deleted.

4.4.1.4 Record failed

This event is sent if record fails and includes the requestId that was specified when the request was issued. A message describing the cause of failure is also included.

4.4.1.5 Control

Sent whenever a DTMF digit arrive on the RTP stream.

4.4.1.6 Abandoned Stream Detected

This event is sent if no packets are received on the inbound media stream for a configured amount of time. The stream must have been abandoned by the peer.

The abandoned stream is included in the event.

4.5 Consumed events

4.5.1.1 Configuration has changed

The configuration has changed and the media stream needs to reread its configuration.

4.6 Function Specification

This FS specifies the following scenarios:

1. Retrieve a new stream
2. Stream creation
3. Stream deletion
4. Stream play
5. Stop an ongoing play
6. Stream record
7. Stop an ongoing record
8. Stream joining/unjoining
9. Stream inbound DTMF
10. Stream outbound DTMF
11. RTCP Support
12. Synchronization

The scenarios above are described in sections below. Other functional requirements not related to a scenario are also described in sections below.

4.6.1 Retrieve a new stream

A new stream instance can be retrieved using a stream factory.

Figure 6 Retrieve a new inbound stream

The scenario:

1. Any component retrieves a new instance of a stream by calling get inbound media stream method or get outbound media stream method depending on direction.

4.6.2 Stream creation

Creating streams is straightforward. The needed data are the media mime types and connection properties (for outbound streams).

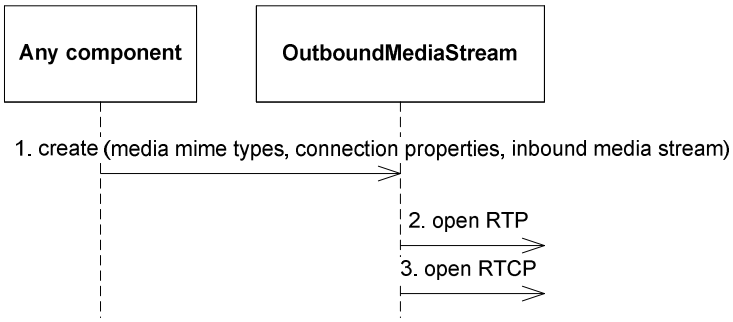


Figure 7 Creation of an outbound stream

In the example above an outbound stream is used but the scenario is valid for an inbound as well (except for the connection properties which are not included then). After the creation the stream can be used for playing, recording, joining etc depending on direction.

1. Any component creates the stream by calling the create function with the required media mime types and connection properties.

2. The RTP stream is opened (two in case of video) to the endpoint to be able to send or receive RTP frames based on if it is an inbound or outbound stream.

3. The RTCP port is opened (two in case of video) to be able to send or receive SR depending on if it is an inbound or outbound stream.

4.6.3 Stream deletion

The deletion of streams is straightforward.

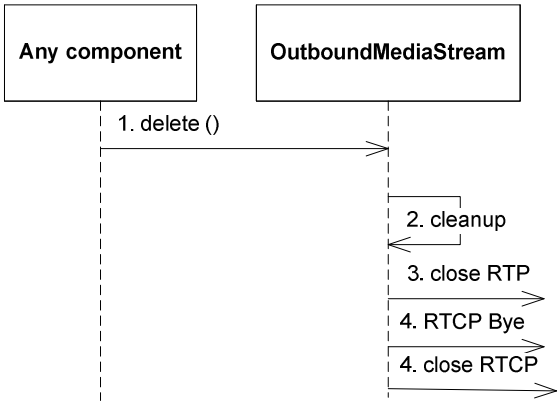


Figure 8 Deletion of an outbound stream

In the example above an outbound stream is used but the scenario is valid for an inbound as well. After the deletion the stream cannot be used any longer.

1. Any component can request the deletion of the stream by calling the delete function.
2. The cleanup can result in a play finished or record finished event if there was an ongoing play or record when the delete occurred.
3. The RTP streams if present are closed.
4. The RTCP streams if present are closed.

Note that joined streams are unjoined before deletion.

4.6.4 Stream play

When a play is issued the media stream will try to play the media as quick as possible taking into account if it is voice or video. Play will not return until the streaming has finished based on the play options during play.

Table 1 Play options

Play Option	Explanation
Wait for audio to finish	The play returns as soon as the audio part of the media has finished playing.
Wait for video to finish	The play returns as soon as the video part of the media has finished playing.
Wait for video and audio to finish	The play returns as soon as both audio and video part has finished playing.
Do not wait	The play returns as soon as possible.

The play options shown in the table above are used by the stream in order to know when to signal that a play has finished.

The cursor describes in milliseconds where in the media object to start streaming.

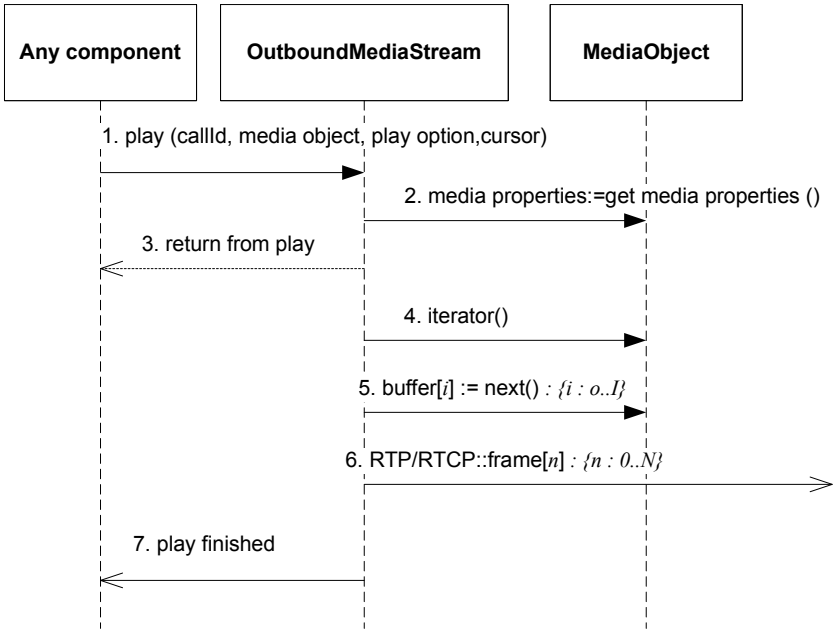


Figure 9 Play of media object

Play of a media object is initiated with the play function on the stream object.

1. Any component issues a play of a media object with requestId, play options and the cursor to the media stream.

2. The media stream checks the media properties of the receiving stream to see if any translation has to be done. In this scenario, translation is not needed.

3. The play returns.

4. The media stream gets the data iterator from the media object.

5. The media stream retrieves a chunk of the data buffer.

6. The media stream packetizes the buffers and sends them over RTP. RTCP are sent at intervals calculated so that RTCP traffic is 5% of the total session bandwidth. 5 and 6 is repeated until the entire data is sent (or play is stopped). If an Inbound Stream exists, the SR might include RR blocks(s) immediately following the SR fields.

7. The stream signals that the play has finished when there are no more buffers available from the media object and all buffers are sent on the RTP streams.

In the case where an array of media objects is played it is handled, within stream, as multiple calls of play, one difference though, play finished is only issued upon play finish of the last media object.

In the case when playing a text media object the stream object must handle the cursor in the way shown in FS Media Translation Manager, see [3].

4.6.5 Stop an ongoing play

When any component wants to stop an ongoing play it calls the stop function.

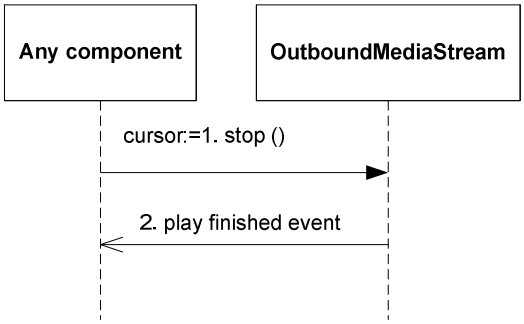


Figure 10 Stop an ongoing play

The scenario:

1. The stop function is called from any component to the media stream. The currently playing media stream is stopped and stored for later use in case the component wants to forward, rewind or pause the media stream by reissuing a play with the same (semantically same) media object.

2. A play finished event is generated.

4.6.6 Stream record

Recording of an inbound stream is done using a media object. The stream puts the received data into the media object.

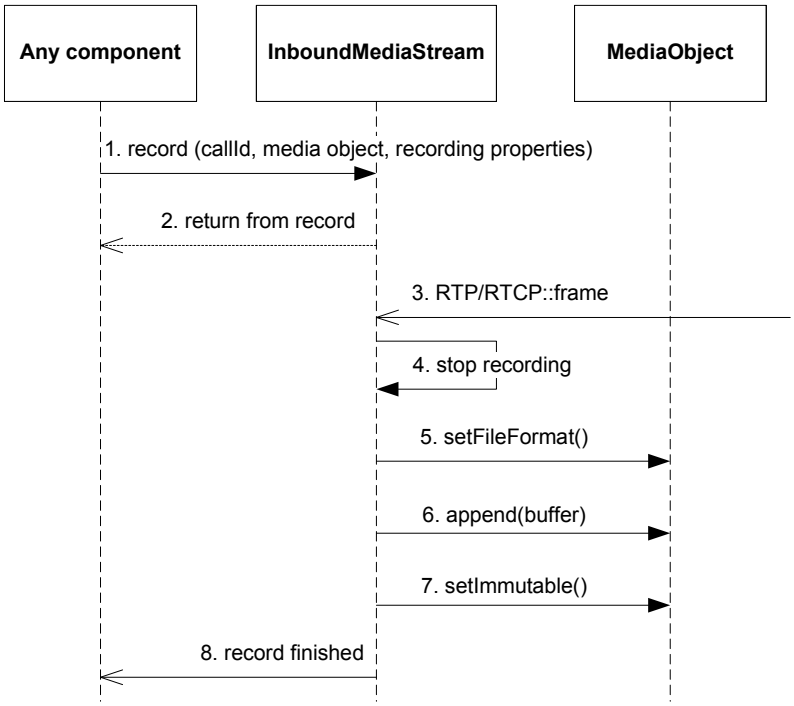


Figure 11 Recording of an inbound stream

The recording of an inbound stream is done with a media object based on recording properties.

1. Any component who wants to record an inbound stream uses the record function and gives the stream a requestId, a media object and recording properties. The inbound stream requests an I-frame from the video fast updater.
2. Record returns.
3. The RTP and RTCP frames are received by the media stream object. RTP frames are checked for energy level (if Silence Detection is enabled)
4. The recording is stopped either by exceeding the max record time, max silence duration or when stop is called by any component.
5. The media stream sets the file format of the media object. The file format is chosen based on configuration for the media mime types used for the stream.
6. The data from the media stream is appended to the media object. This is done for each chunk of data received.
7. When the record has finished, stream sets the media object to immutable, i.e. its data is not allowed to be modified.
8. The media stream object sends a record finished event with the reason code "max record time exceeded" or "stopped".

4.6.6.1 Record start

In the record, any one who wants to initiate a record has the possibility to add a media object that shall be played prior to record.

For audio streams a record is started after a possible play. Silence Detection functionality will be invoked (if enabled). Silence Detection functionality is suppressed if the audio stream belongs to a video call.

For video streams a record is started after a possible play; when the first I-frame has been received or when a preconfigured time has expired.

4.6.6.2 Ignoring first part of record

When record is done, the first received audio data shall be ignored for a configurable amount of time. For an audio stream, that data shall just be skipped in the recorded media object. For a video stream, the received video data are kept but the audio data during this time is replaced with silence.

The reason for skipping/replacing the initial audio data is to remove the possibility of recording the beep played before record is started.

For this purpose, there are two configuration properties (see 4.6.20):

- Amount of milliseconds of audio to skip for audio streams
- Amount of milliseconds of audio to replace with silence for video streams

4.6.6.3 Ignoring silent part of record

If silence is detected, recording is aborted. Silent data, at end of recording, will be removed from memory (stripped from recorded data), except for 900 ms of silence that is retained in order to prevent any loss of spoken audio.

Note that silence is only removed at end of recording, i.e. the max silence duration must be a sufficiently long period of time to allow for inter-word silence and a short silent period at the start of recording.

4.6.6.4 Silence detection (detailed description)

This chapter contains in principle the same description as the OM MAS (Stream part), but has more details on implementation and added figure to clarify the functionality, especially during dynamic silence detection.

Silence Detection Functionality
Silence detection in MAS is default disabled for both audio and video calls. When silence detection is enabled it gets enabled for audio calls. Currently, the silence detection is not enforced on the audio stream of a video call. It is necessary that the silent audio packets are sent by the gateway or end-point all the time (e.g. in case the user mutes his audio), silence detection will not work if MAS does not receive an audio stream. If no audio packets are received there will be no recording.

Three silence detection modes are supported in MAS:

- No Silence Detection mode
- Static Silence Detection mode
- Dynamic Silence Detection mode

No Silence Detection mode
To turn off the Silence Detection, the SilenceDetectionMode must be set to 0 (See stream part of OM – MAS, ref [6] for parameter description). When the Silence Detection is turned off, the recording is continued until one of the following occurs:

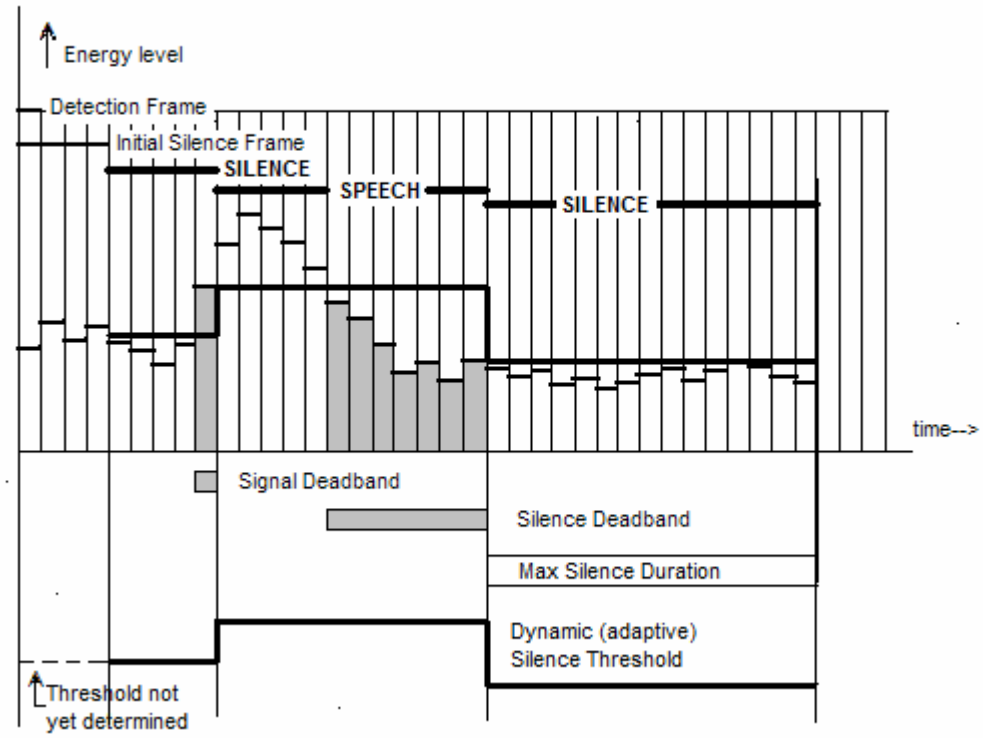
- Maximum recording duration is reached.
- Caller has hung up.
- Caller has pressed the specified DTMF to interrupt.

Static Silence Detection mode
To set the Silence Detection to Static Silence Detection mode, the SilenceDetectionMode parameter must be set to 1 (See stream part of OM - MAS, ref [6] for parameter description). This mode uses the configuration attribute “SilenceThreshold” to decide on the silence frames. Every PCM sample value in an audio frame is compared with this threshold value. A PCM sample value is deemed to be silent if it is below the threshold. A complete frame is deemed silent if 90% of the samples are silent.

Dynamic Silence Detection mode
To set the Silence Detection mode to Dynamic Silence Detection mode, the “SilenceDetectionMode” parameter must be set to 2 (See stream part of OM - MAS, ref [6] for parameter description). In this option, the silence detection works on an adaptive threshold value. Here the threshold value is not for each PCM sample; rather it is for the average energy level of a frame used for detection. Here in this mode following configuration parameters are used:

- InitialSilenceFrames
- DetectionFrames
- SilenceDeadband
- SignalDeadband

The following figure shows how the configuration parameters relate to the adaptive threshold value. Below the figure, the subsequent sections describe the parameters in detail.



Note that the figure is simplified since there would normally be a lot more transitions between speech to silence and vice versa, because of the inter word silent period.

The initial threshold value is calculated by averaging the energy level of first incoming audio frame(s) of length (in msec) as specified by the attribute “InitialSilenceFrames”, at the start of recording. Note: Currently this attribute is not used and the initial threshold value is based only on the first RTP frame of 40ms (provided that is the RTP packet size). The intention of the attribute is for the silence detector to look into an initial number of RTP frames to determine the average energy value to use as a first reference threshold value.

The subsequent incoming frames of length as specified by the attribute “DetectionFrames” are averaged for energy and compared with the current threshold. The default detection frame size is chosen as 10 ms since it gives a better granularity while averaging for energy and better zero crossing rate for voice signal. If an RTP packet is larger than 10ms, it is broken into smaller packets (e.g. 4 packets) and the energy is calculated for each of them. This breakup into smaller packets enables the silence detector to catch the energy level when the lower and higher energy samples are having higher statistical variance within a single packet. The mean value over the complete packet of 40ms will fail to capture this variance. By increasing this value the sensitivity towards the lower energy values will decrease.

The threshold value is lowered when a transition occurs from speech to silence and vice versa. The increment in threshold value is less monotonic than the decrement since we want to quickly come back to better silence threshold in the event of transition from noise or speech burst.

In order to make silence transition more conservative, we keep a silence dead band as specified by the attribute “SilenceDeadBand”. A transition from speech to silence occurs when

consecutive silent (as decided by comparing with the current threshold value) audio frames cross the “SilenceDeadBand” value. The default value is 150 ms. The default value is chosen since the inter word silence and hysteresis on an average speaker is less than 150 ms.

Similarly, a “SignalDeadBand” value is used to decide on transition from silence to speech. The speech transition is not constrained and the default speech dead band is 10 ms (that is 1 detection frame size). The transition from silence to speech occurs when we detect consecutive speech audio frames equal to “SignalDeadBand”

If silence has reached the max allowed silence duration, the Silence Detection functionality will abort the recording (or rather inform the Record Job class to abort recording).

4.6.7 Stop an ongoing record

When any component wants to stop an ongoing record it calls the stop function.

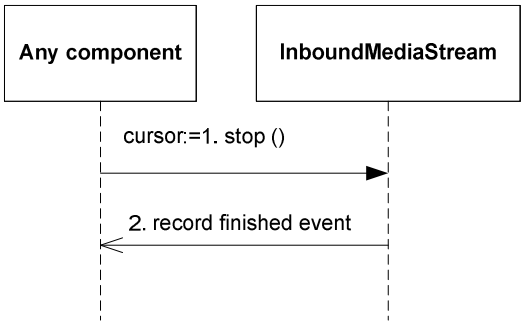


Figure 12 Stop an ongoing record

The stop scenario:

1. The stop function is called on the media stream object.
2. A record finished event is generated.

4.6.8 Stream joining and unjoining

When dealing with call transfer or ASR/TTS engines it usually involves transferring media from one inbound stream to one outbound stream. This is accomplished with the join and unjoin functions.

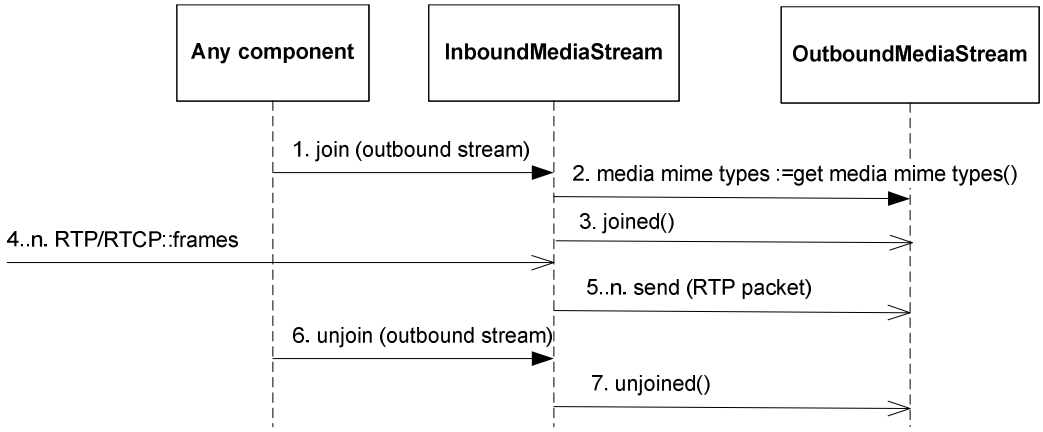


Figure 13 Joining and unjoining streams

Any component can join two streams; the join is performed on the inbound stream to an outbound stream.

1. Any component calls the join function on the inbound stream giving the outbound stream as a parameter.
2. The inbound streams checks the media mime types of the two streams, they must match.
3. If the media mime types matches, the outbound stream is informed of the join.
4. The arriving RTP/RTCP frames are received on the inbound stream.
5. The inbound stream sends the received raw RTP packets to the outbound stream. The outbound stream must perform RTP and RTCP SR packet header rewrites to fit the outbound RTP streams counters etc.
6. When any component no longer wants to join the two streams the function unjoin with the outbound stream as a parameter is called.
7. The inbound stream informs the outbound stream that an unjoin has been performed. This must be done after all buffers in the inbound stream have been sent to the outbound stream. The unjoined function call tells the outbound stream that no more data is arriving from the inbound stream.

4.6.9 Stream inbound DTMF

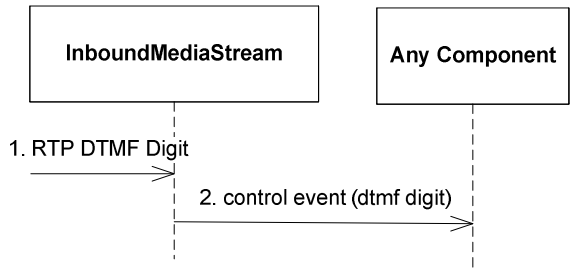


Figure 14 Inbound RTP DTMF digit

The DTMF digit arrives as a separate payload in the audio RTP stream and is converted to an event and sent to any component.

1. A DTMF digit arrives in the RTP stream as a separate RTP frame and payload, see [2].
2. The Stream component converts the digit to a control event and sends it to any component subscribing for that signal through the event dispatcher.

It is configurable in stream if the control event should be generated at key down or key up, see 4.6.20.

4.6.10 Stream outbound DTMF

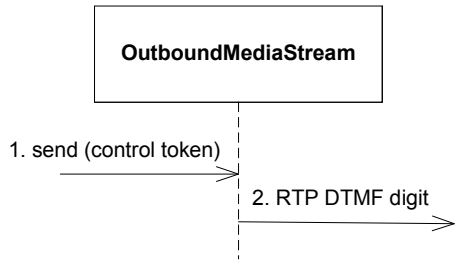


Figure 15 Outbound RTP DTMF digit

The DTMF digit can be sent on the audio RTP stream as a separate RTP frame with a special payload type.

1. Any component wants to send a digit/token to the endpoint and calls the send function with the control token.
2. The stream converts the token to an RTP frame with a special payload type and sends it in the RTP stream, see [2].

4.6.11 RTCP Support

The stream component supports sending and receiving of RTCP SR in addition to this it also supports all required RTCP messages as described in [1].

Stream sends RTCP RR as well but does not analyze received RR.

The only SDES item sent by stream is the mandatory CNAME item.

BYE is sent at the end of a RTP session (deletion of stream).

When negotiated over SDP, as specified in [8], Stream supports sending Video Fast Update requests (or Full Intra requests, FIRs) in the form of RTCP Feedback, as described in [6].

4.6.11.1 SDP Bandwidth modifiers

SDP Bandwidth modifiers can be used to limit the bandwidth for RR and RS within the RTCP protocol.

If the SDP offer contains bandwidth modifiers, they shall be compared to max/min values that may be configured for the specified payload, and if out of bounds, the offer shall be rejected.

Default values for bandwidth modifiers of each payload shall be possible to configure. If an SDP offer lacks bandwidth modifiers, the SDP answer may contain the default values. For more information, see [8] and [9].

4.6.12 Synchronization

Synchronization of voice and video is handled in one of three ways by the Stream component:

- RTCP Synchronization
- Local Synchronization
- Local and RTCP Synchronization

The one to use is configurable, see skewmethod 4.6.20.2.

It is assumed in this document that all media in the media objects are synchronized.

4.6.12.1 RTCP synchronization

The RTCP SR provides the correlation between an RTP Timestamp to a world clock. This correlation can be used to synchronize two RTP streams, e.g. audio and video streams.

RTCP synchronization means that the difference between the two inbound RTP streams in relation to the world clock is calculated from the RTCP SR.

RTCP synchronization cannot be applied on outbound streams.

4.6.12.2 Local synchronization

Local synchronization means that the difference between the two RTP streams in relation to the world clock is fixed instead of calculated from the RTCP SR. The fixed value to use is set by the component that created the media stream using the *setSkew* method.

On inbound streams synchronization is done during recording only, otherwise no synchronization is needed.

On outbound streams synchronization is done by rewriting RTP timestamps to compensate for the fixed delay between the streams.

4.6.12.3 Local and RTCP synchronization

Synchronization is done using both local and RTCP synchronization.

4.6.13 Detection of Abandoned Streams

Stream has a mechanism for detecting abandoned inbound streams, i.e. inbound streams where the peer has stopped transmitting. If no RTP packets are received for a configurable amount of time (see 4.6.20), the stream is considered abandoned. Stream generates the event Abandoned Stream Detected to warn other parts of the system.

4.6.14 Video Fast Update/Forced Update

An inbound media stream is passed a "video fast updater" when created. This object is used to request an I-frame before the stream starts to wait for packets. The recording starts when the first I-frame has been received or with the first frame after a configurable timeout.

4.6.15 Protocol Encapsulation

The implementation of the RTP/RTCP protocols is encapsulated. This is done to be able to exchange the current RTP/RTCP protocol stack with another one.

4.6.16 Codec Support

- G.711 PCMU
- G.711 PCMA
- H.263 Baseline@Level 0
- AMR (3GPP Release 6)

4.6.17 Use of File Format, File extension and Content Type

The file extension set in a Media Object is used to choose the appropriate parser/builder for the media. If no file extension is set, the content type is used to choose the parser/builder.

The content type is not sufficient, in all situations, to make the correct choice. For binary formats, the same content type might have different file extensions and should probably be read/written by different parsers/builders. This will, however, not be a problem with the currently supported formats supported by stream.

4.6.18 File Format Support

Stream currently supports the following file formats:

- WAV
- MOV
- 3GPP

The implementation of file formats is done to enable easy extension of new file formats. Although, adding new file formats to stream cannot be done pluggable during runtime.

4.6.18.1 WAV limitations

- Fact chunk is not supported.
  This chunk contains information about the number of samples in the data chunk. Because the WAV standard does not guarantee that this chunk is saved before the data chunk, this is not read. If this chunk is at the end of the file, the whole file must be read before the data can be streamed.
- Wave List chunk is not supported.
  The Wave List chunk is a list of several "sint" and "data" chunks and rarely used so therefore it is not supported.

In addition to this, the following list of chunks is not used. They are not needed when streaming a wav-file and there are currently no requirements that require them to be used when saving a wav-file either:

- Cue chunk "cue "
- Associated Data List chunk "list"
- Playlist chunk "plst"
- Label chunk "labl"
- Note Chunk "note"
- Labeled Text Chunk "ltxt"
- Sampler Chunk "smpl"
- Instrument Chunk "inst"

4.6.18.2 MOV limitations

MOV handling is limited to provide the functionality necessary to handle audio and video streaming:

- A MOV file is considered to contain three tracks: audio (PCMU/PCMA), video (H.263), video hint (RTP).
- The audio track is homogenous and consists of samples of same size (and duration). The size corresponds to the handled payload time (with other words the RTP payload data).
- The video track contains the H.263 frames where each frame is stored as a sample.
- The video hint track contains the information on how to transform each H.263 into RTP packets.

4.6.18.3 3GPP limitations

3GPP handling is reusing the MOV handling and hence suffering from the same limitations.

4.6.19 Logging

Stream does not log errors. Stream will issue warning and informational log events though. When not able to fulfill request due to a failure Stream should issue a failure event (e.g. play failed) describing the cause of the failure and log this as an informational log event. There are exceptions where it is impossible to issue any events, in these cases are logged by a warning log event.

4.6.20 Configuration

The configuration needs for a media stream are listed here. Illegal values in the configuration are logged as error and the default value is used.

4.6.20.1 RTP/RTCP

- The host name or address to use for the RTP/RTCP protocols.
  If no host is given, Stream chooses the IP address itself.

- The port number range to use for inbound media streams.
- The amount of time in milliseconds to wait before an inbound stream receiving no RTP packets are considered abandoned.
- The packet time (pTime) desired packet length (in milliseconds) for inbound streams. For outbound streams, the pTime to use is given by the peer party. Note that this is treated as a default value. If the peer party of an inbound stream chooses a value different value from the configured the peer party pTime is used.
- The maximum packet time (maxPTime) for inbound streams. Maximum amount of media that can be encapsulated in each packet, expressed as time in milliseconds. For frame-based codecs, the time should be an integer multiple of the frame size. For outbound streams, the maxPTime to use is given by the peer party.

4.6.20.2 skewmethod

The skewmethod is used to configure the synchronization of audio and video streams. Skewmethod can be:

- Local: A configurable delay is added to the audio or video stream.
- RTCP: When recording video, if RTCP information is available for both the audio and video stream, this is used to calculate the skew between the first audio packet and the first video packet. This skew is saved as an offset in the media file. Does not affect outbound streams.
- Local and RTCP: No difference from Local for an outbound stream. For an inbound, the result is the sum of skew according to local synchronization and RTCP synchronization.

Examples:

Outbound stream, local skew=100. Result: The first video packet is delayed 100 milliseconds relative to the video start offset.

Outbound stream, local skew=-400. Result: The first audio packet is delayed 400 milliseconds relative to the audio start offset.

Inbound stream, local skew=100, RTCP skew says first video packet is 300 ms ahead of first audio packet, that is, when playing the media, audio should be delayed by 300 ms relative to the first video packet. Result: Media is saved with an audio offset (audio delayed) of 200 ms.

4.6.20.3 Recording

- Amount of milliseconds of audio to skip for audio streams
- Amount of milliseconds of audio to replace with silence for video streams
- Timeout when waiting for an I-frame for video streams. After this timeout, the recording will start at the next frame.

4.6.20.4 DTMF

- Whether the DTMF detection should be made based on key down or key up.

4.6.20.5 Supported media

- Supported content types. Any Media Object sent to stream with a content type that is not set in the configuration file will be sent to MTM for translation.
- Default mappings for supported payloads

5 Glossary

RTCP SR	RTCP Sender Report
RTCP RR	RTCP Receiver Report
RTCP SDES	RTCP Source Description

6 References

[1] RTP/RTCP specification
RFC 3550

[2] DTMF in RTP
RFC 2833

[3] FS – Media Translation
15/FS-MAS0001

[4] CNG
RFC 3389

[5] FS – Media Object
13/FS-MAS0001

[6] OM – MAS
1/OM-MAS0001

[7] Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)
draft-ietf-avt-avpf-ccm-10
http://tools.ietf.org/html/draft-ietf-avt-avpf-ccm-10.html

[8] IWD – SIP Telephony
1/IWD-MAS0001 Uen

[9] SDP Bandwidth Modifiers
RFC 3556