FFmpegVideoStream protocol handler

The FFmpegVideoStream protocol handler allows reading, writing, streaming and receiving video data in different format with a lot of different options. It uses the FFmpeg Java port of JavaCV and is available in the video feature.

Options

- **streamurl**: The target or source URL. This is where a stream is sent to or received from, and a file is written or read from.
- **usedelay**: When reading from a file, applies a delay after reading each frame corresponding to the frame rate.
- **timestampmode**: Defines how image time stamps will be generated (default: start):
  - `start:` Frame timestamps will be extrapolated from the start time stamp specified as an extra option (`starttime`)
  - `filetime:` Frame timestamps will be extrapolated from the "file last modified" time stamp
  - `syncfile:` Not yet implemented. Timestamps for each frame will be read from a synchronization file, which is specifiead as an extra option (`syncfilename`)
  - `none:` Frame timestamps will be set to the current system time and will not contain an ending timestamp
- **framerate**: When streaming or writing, this specifies the frame rate of the stream. When writing, the framerate is written in the video file. (optional)
- **bitrate**: When streaming or writing, this specifies the bitrate of the stream. The quality and size of the resulting video can be controlled with this. (optional)
- **videocodec**: When streaming or writing, this specifies the video codec used for the stream. (default 13, see available video codecs)
- **videoquality**: When streaming or writing, this specifies the quality of the stream, where 0 is best and larger values increase compression. The size of the resulting video can be controlled with this. (optional)
- **format**: When streaming or writing, this specifies the underlying container format (for example "h264" for streaming, or "mp4" for writing). Useful if the format cannot be determined by the stream URL extension. (optional)
- **pixelformat**: When streaming or writing, specifies the pixel format of the stream. When reading or receiving, specifies the pixel format of the generated image. (optional)
- **framesizemultiple**: When writing or streaming, each frame will be extended so it's width and height is a multiple of **framesizemultiple**. Useful when writing images which are not correctly aligned for the current codec. (default 1)
- **bitsperpixel**: Defines the bits per pixel of the stream or of generated frames. (optional)

Codec options (FFmpeg only)

FFmpeg allows fine-tuning of codec parameters (See the FFmpeg streaming guide). To activate these options, use "**codec:**" in front of the parameter:

- **codec:preset**: Can be used to control encoding quality and speed. Possible values include **veryslow**, **slow**, **fast** and **ultrafast**.
- **codec:tune**: Can be used to control latency when streaming video. Possible values include **zerolatency**.

Schema

The output of the handler provides the following attributes as a schema. Attributes can be in arbitrary order and will be identified by the type.

Name	Type	Description
image	IMAGEJCV	The current frame of the video stream
starttimestamp	STARTTIMESTAMP	The start time stamp of the frame. Depends on the `timestampmode` option, will be omitted when set to 'none'
endtimestamp	ENDTIMESTAMP	The end time stamp of the frame. Depends on the `timestampmode` option, will be omitted when set to 'none'

Example

This example shows how to read a video file and stream it to a remote client in video speed with h264, ultrafast, zerolatency and 400kBit/s:

PQL

How to stream a video from file

```
video = ACCESS({source='Video',
                                wrapper='GenericPull',
                                transport='none',
                                protocol='FFmpegVideoStream',
                                datahandler='Tuple',
                                options=[
                                                        ['streamUrl', 'video.avi'],
                                                        ['timeStampMode', 'filetime'],
                                                        ['useDelay', 'true']
                                                ],
                                schema=         [
                                                        ['image',                       'IMAGEJCV'],
                                                        ['starttimestamp',         'STARTTIMESTAMP'],
                                                        ['endtimestamp',
'ENDTIMESTAMP']
                                                ]})

/// or shorter, as a source operator:
video = FFMPEGVIDEO({source='Video', options=[['streamUrl', 'video.avi'],
                                                ['timeStampMode', 'filetime'],
                                                ['useDelay', 'true']]})

output = SENDER({sink='Sink',
                                wrapper='GenericPush',
                                transport='none',
                                protocol='FFmpegVideoStream',
                                dataHandler='Tuple',
                                options=[       ['framerate', '30.0'],
                                                        ['streamUrl', 'udp://127.0.0.1:12345'],
                                                        ['bitrate', '400000'],
                                                        ['format', 'h264'],
                                                        ['codec:tune', 'zerolatency'],
                                                        ['codec:preset', 'ultrafast']
                                                ]
                                },
                                video)
```

This example shows how to receive a video stream and write it to an mp4 video file using the MPEG-4 codec (13).

PQL

How to receive a stream and encode it into a file

```
video = ACCESS({source='Video',
                                 wrapper='GenericPull',
                                 transport='none',
                                 protocol='FFmpegVideoStream',
                                 datahandler='Tuple',
                                 options=[
                                                         ['streamUrl', 'udp://127.0.0.1:12345'],
                                                         ['timeStampMode', 'none']
                                                 ],
                                schema=         [
                                                         ['image',
'IMAGEJCV']
                                                 ]})

/// or shorter, as a source operator:
video = FFMPEGVIDEO({source='Video', options=[['streamUrl', 'udp://127.0.0.1:12345'],
                                                  ['timeStampMode', 'none']]})

output = SENDER({sink='File',
                                wrapper='GenericPush',
                                transport='none',
                                protocol='FFmpegVideoStream',
                                dataHandler='Tuple',
                                options=[        ['framerate', '30.0'],
                                                         ['streamUrl', 'video.mp4'],
                                                         ['videoCodec', '13']
                                                   ]
                                }, video)
```