Adding W3C Media Source Extensions and Encrypted Media Extensions to GStreamer

Jordan Yelloz
Senior Software Engineer

About Me

- Working on GStreamer projects at Collabora since 2022
- Previously worked at Amazon Video and a few much smaller companies
  - Projects ranging from digital print automation, GStreamer, Linux audio drivers, web services
- Based in Fort Collins, Colorado, USA

- Media Source Extensions (MSE) Introduction
- Encrypted Media Extensions (EME) Introduction
- GStreamer MSE Library
- GStreamer EME Interfaces
- GStreamer EME Implementations
- Development Challenges

COLLABORA

Media Source Extensions

Introduction

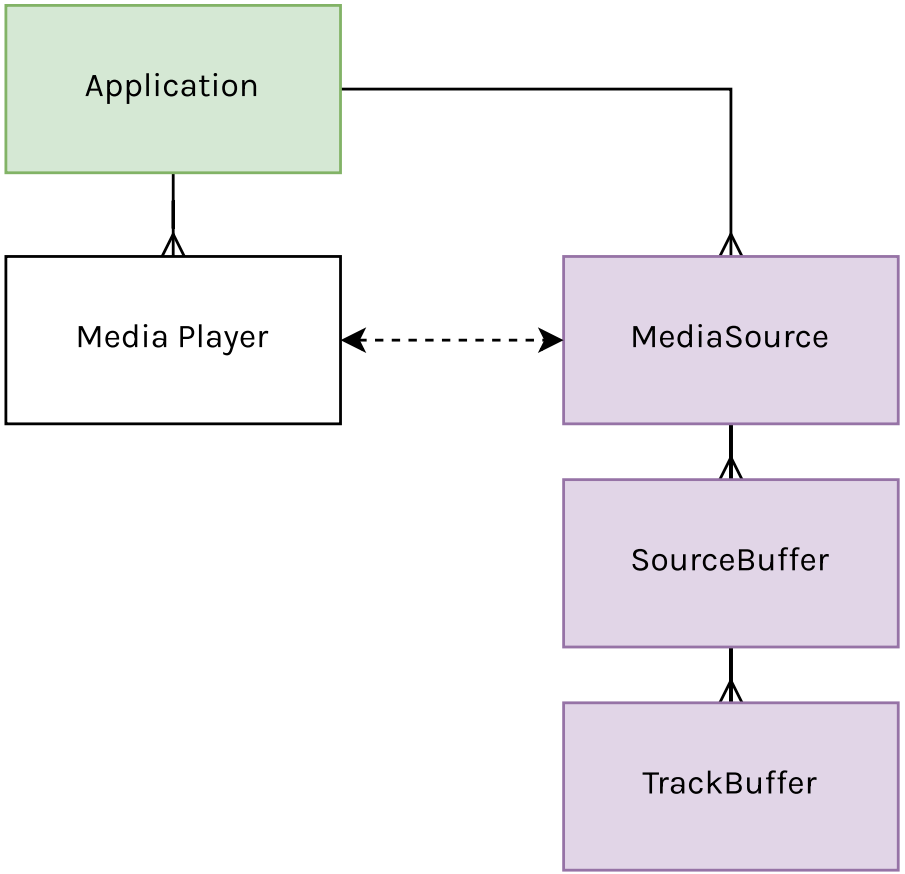
Media Source Extensions - Intro

- Web technology allowing programmatic input to `<audio>`/`<video>` elements
- Demuxes and parses raw data into timecoded samples
- Stores samples in time-addressable data structures
- Supported formats:
  - Fragmented MP4
  - WebM
  - MPEG-TS
  - MP3/M4A audio

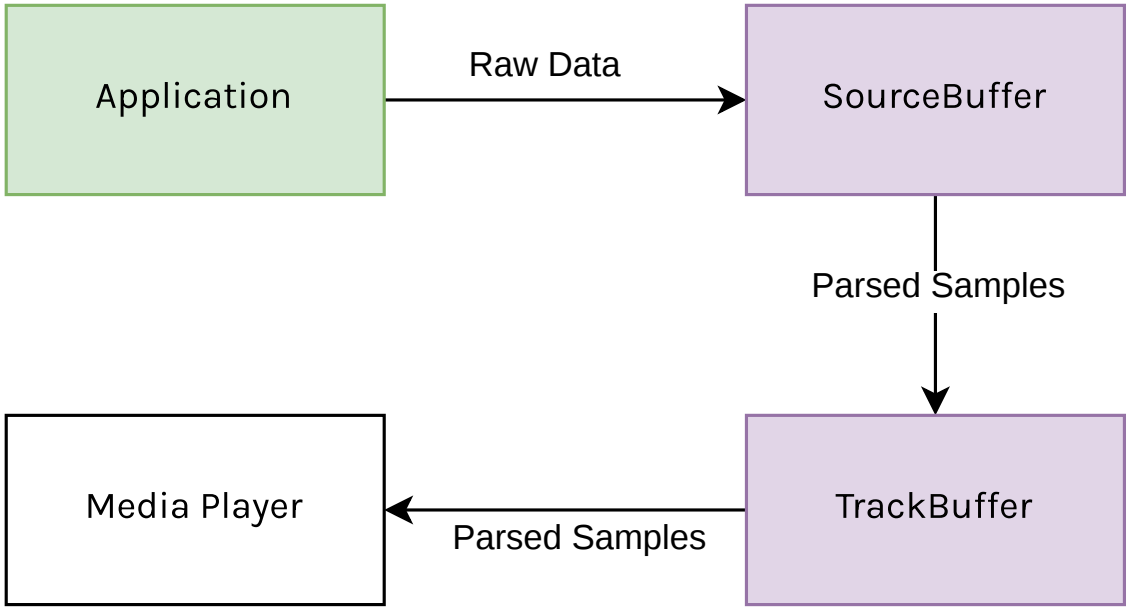
Media Source Extensions - Intro

- HTML Media Element
  - Playback component
- Media Source
  - Entry point to MSE API, group of Source Buffers, attached to Media Element
- Source Buffer
  - Single byte stream of content
  - Bytes are parsed and organized into Track Buffers
- Track Buffer
  - Parsed timeline of encoded samples for a single track, may have gaps
  - Feeds media into playback component

MSE - Structural Relationships



MSE – Data Flow



COLLABORA

Encrypted Media Extensions
Introduction

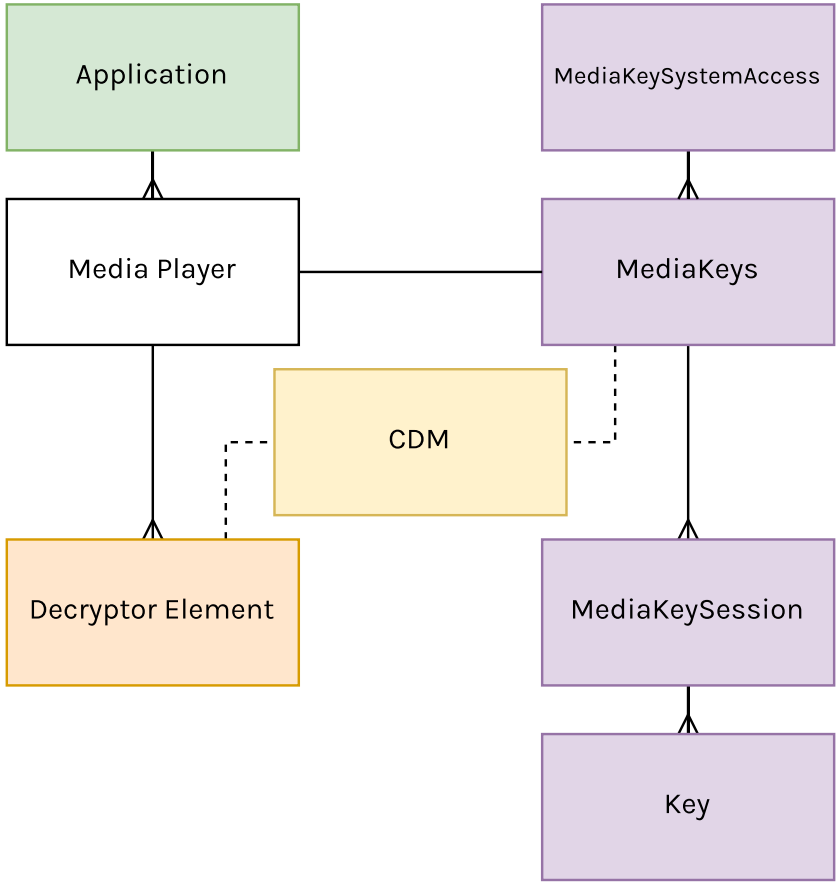
Encrypted Media Extensions - Intro

- Web technology for decryption of encrypted media
- Primarily defines communications pattern between Application, License Authority, and Content Decryption Module (CDM)
- Supported container formats:
  - MP4, WebM
- Relies on Common Encryption (CENC) scheme for each supported container
  - Allows the same encrypted media to be processed by multiple CDMs
  - Initialization Data within container informs system which keys are needed to decrypt a span of media
- Specifies "Clear Key" decryption system for evaluation purposes
- Web browsers integrate commercial CDMs

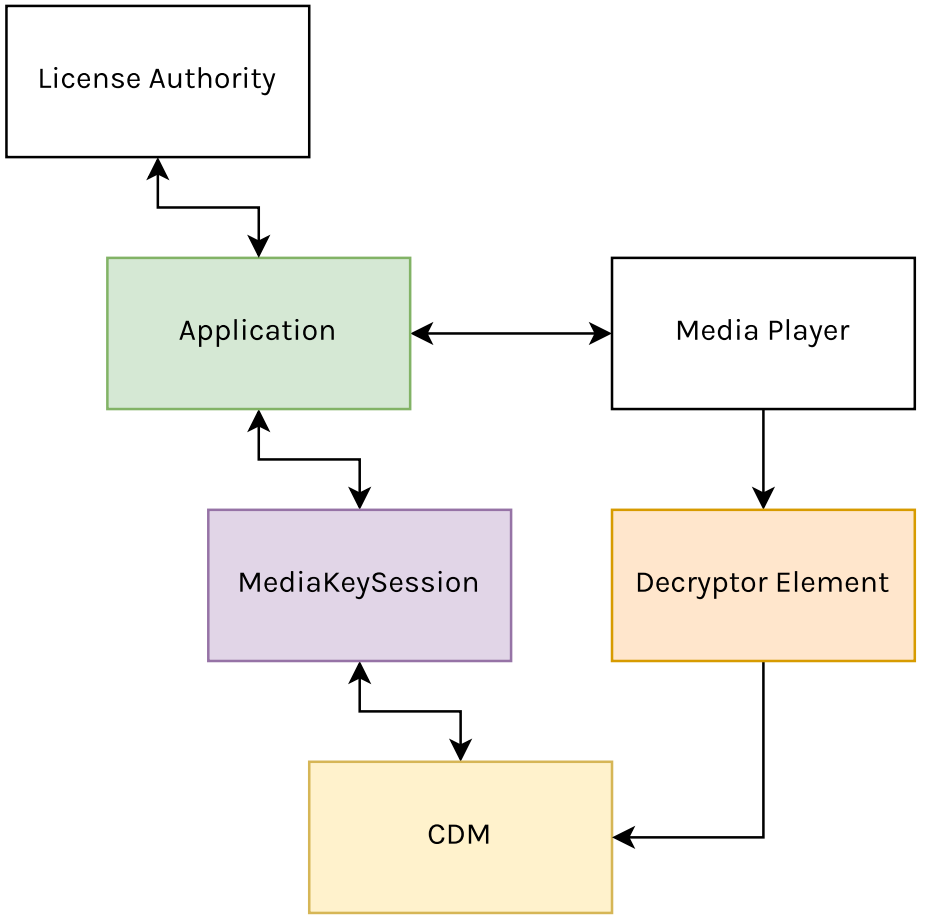
Encrypted Media Extensions - Intro

- MediaKeySystemAccess
  - Builds Media Keys instance when possible
- MediaKeys
  - Wrapper for underlying CDM instance, maintains Sessions
- MediaKeySession
  - Represents the keys referenced in a single unit of Initialization Data

EME - Structural Relationships



EME – Data Flow



COLLABORA

GStreamer MSE Library

GStreamer MSE Library

- Allows applications to use MSE API without a web browser library
- Based on existing WebKit implementation
  - Converted from C++ to GObject C and simplified
- Implementation relies on `appsrc`, `parsebin`, and `appsink`
- Adds custom `msesrc` element handling `mse://` URI scheme
- Integrates with `GstPlay` / `playbin`

GStreamer MSE – Application Usage

- Create Pipeline with `msesrc` element
  - `playbin3 uri=mse://` should be enough
- Create Media Source, attach to `msesrc`
- Add Source Buffers to Media Source
- Play pipeline
- Feed Source Buffers with data

COLLABORA

GStreamer EME Library

Protected Media in GStreamer

- What exists now inside GStreamer?
  - Demuxers
    - Tag buffers with `GstProtectionMeta`
    - Raise `GST_EVENT_PROTECTION`
    - Supported demuxers: MP4, WebM, DASH, MSS
  - Decryptors?

GStreamer EME Library

- A set of Interfaces and Data Models
  - `GstMediaKeySystemAccess` - Provides `GstMediaKeys` instance
  - `GstMediaKeys` - CDM instance wrapper, manages lifecycle of sessions
  - `GstMediaKeySession` - Groups related keys, manages lifecycle of keys
- Also defines a convention for Content Decryption Module plugins
  - Protection System - Entry point
- API relies heavily on `GstPromise`, matching W3C EME's use of JS Promises

GStreamer EME – Application Usage

- Set up pipeline with decryptor element or just use `GstPlay`
- Instantiate supported protection system(s)
  - Request `GstMediaKeySystemAccess`
  - Create `GstMediaKeys`
- Watch the Bus for `GST_MESSAGE_NEED_CONTEXT` and inform origin element of preferred protection system
- Watch the Bus for `eme-encrypted` message from decryptor element
- Asynchronously answer contained promise with appropriate `GstMediaKeys` instance
- Create session for each new unit of Initialization Data
- Request License from License Authority
- Feed License Authority's response back to Session

GStreamer EME – CDM Integration

- Multiple options
  - Write a plugin:
    - Implement `GstMediaKeySystemAccess`, `GstMediaKeys`, `GstMediaKeySession`
    - Implement custom decryptor element
  - Re-use an OpenCDM plugin

GStreamer EME – CDM Integration

- Using Widevine CDM included with Web Browsers
- Create OpenCDM module wrapper
  - Module C++ Headers are distributed in browser source tree
  - Discover local installation path
  - Link at runtime using GModule
- Application
  - Instantiate CDM
  - Handle Messages
  - Communicate with License Authority

COLLABORA

Development Challenges

Development Challenges - MSE

- Porting from WebKit
  - Removal of HTML/DOM/JavaScript concepts
  - Removal of threading model of WebKit
  - Simplification of design - WebKit is designed to support multiple implementations of MSE for platforms that don't rely on GStreamer
- Conformance Testing
  - Existing Web Platform Tests rely on web browser

Development Challenges - EME

- Reliance on `GstPromise`: lots of utility code to pack/unpack `GstStructure` fields
- Decryptor elements: GStreamer elements must advertise supported key systems statically
  - CDMs might not have a mechanism to enumerate supported key systems
  - Issue for implementations that are abstractions over multiple CDMs

Development Status

- MSE:
  - https://gitlab.freedesktop.org/gstreamer/gstreamer/-/merge_requests/2992
- EME:
  - Coming Soon

Thank you!

We are hiring

col.la/careers