UNIFIED VIDEO ENGINE API v0.5

Overview

The goal of the external interface of the Unified Video Engine is to provide a uniform control interface for a variety of video technologies (e.g. HLS, MPEG-DASH, VPAID, etc) for use by any Player SDKs.

The Unified Player shall support APIs for three business models and two stream types:

	Ad Supported	Transaction Supported	Subscription Supported
VOD	AVOD	TVOD	SVOD
Linear	Ad-supported linear	*NA*	Subscription-supported linear

All UI/UX concerns are out-of-scope.

Timecodes

Timecodes are all in milliseconds, to support playback of any type of media.

For live/linear, ms are based from the start of playback.

For VOD, ms are based on zero as the start.

API

The Unified Video Engine shall have the core methods needed to playback an asset.

1. IVideoEngine interface:

IVideoEngine interface includes,

- **load()**

- URI of the Media being played by the Video Engine
- Parameters: uri of type String
- Return Value: None
- Usage:

```
load(uri: string);
```

- **initConfig()**
  - IConfig Object with key value pair of launch configuration
  - Parameters: IConfig of type Object
  - Return Value: None
  - Usage:

```
initConfig(IConfig: Object[]);
```

- **play()**
  - Starts playback when enough data is buffered at playhead
  - Parameters: None
  - Return Value: None
  - Usage:

```
play(): void;
```

- **pause()**
  - Pauses playback
  - Parameters: None
  - Return Value: None
  - Usage:

```
pause(): void;
```

- **stop()**
    - Stop playback and free resources
    - Parameters: None
    - Return Value: None
    - Usage:

```
stop(): void;
```

- **seek()**
    - Performs a seek
    - Parameters: timeSec the time in seconds to seek
    - Return Value: None
    - Usage:

```
seek(timeSec: number): void;
```

- **getCurrentState()**
    - Gets the duration of the content in seconds
    - This defaults to PlayState.PAUSED
    - Parameters: None
    - Return Value: PlayState
    - Usage:

```
getCurrentState(): PlayState;
```

- **getDurationSec()**
    - Gets the duration of the content in seconds
    - Parameters: None
    - Return Value: Number
    - Usage:

```
getDurationSec(): number;
```

- **getCurrentPosition()**
  - Gets the current user time in seconds
  - Parameters: None
  - Return Value: Number
  - Usage:

```
getCurrentPosition(): number;
```

- **getVideoBitrates()**
  - Gets available video bitrates
  - Parameters: None
  - Return Value: Number
  - Usage:

```
getVideoBitrates(): number;
```

- **getAudioBitrates()**
  - Gets available audio bitrates
  - Parameters: None
  - Return Value: Number
  - Usage:

```
getAudioBitrates(): number;
```

- **getCurrentVideoBitrate()**
  - Gets the current video bitrate

- Parameters: None
- Return Value: Number
- Usage:

```
getCurrentVideoBitrate(): number;
```

- **setVideoBitrate()**
  - Sets the current video bitrate
  - Parameters: bitrate of type Number
  - Return Value: None
  - Usage:

```
setVideoBitrate(bitrate: number): void;
```

- **getCurrentAudioBitrate()**
  - Gets the current audio bitrate
  - Parameters: None
  - Return Value: Number
  - Usage:

```
getCurrentAudioBitrate(): number;
```

- **setAudioBitrate()**
  - Sets the current audio bitrate
  - Parameters: bitrate of type Number
  - Return Value: None
  - Usage:

```
setAudioBitrate(bitrate: number): void;
```

- **getAudioTrack()**
  - Gets the current audio track
  - Parameters: None
  - Return Value: IAudioTrack or NULL
  - Usage:

```
getAudioTrack(): IAudioTrack | null;
```

- **setAudioTrack()**
  - Sets the current audio track
  - Parameters: track of type IAudioTrack
  - Return Value: None
  - Usage:

```
setAudioTrack(track: IAudioTrack): void;
```

- **getTextTrack()**
  - Gets the current text track
  - Parameters: None
  - Return Value: ITextTrack or NULL
  - Usage:

```
getTextTrack(): ITextTrack | null;
```

- **setTextTrack()**
  - Sets the current text track
  - Parameters: track of type ITextTrack
  - Return Value: None
  - Usage:

```
setTextTrack(track: ITextTrack | null): void;
```

- **getVolume()**
  - Gets the current volume (value between 0 and 1)
  - Parameters: None
  - Return Value: Number
  - Usage:

```
getVolume(): number;
```

- **setVolume()**
  - Sets the current volume (value between 0 and 1)
  - Parameters: volume of type Number
  - Return Value: None
  - Usage:

```
setVolume(volume: number): void;
```

- **getPlaybackRate()**
  - Gets the current playback rate
  - Parameters: None
  - Return Value: Number
  - Usage:

```
getPlaybackRate(): number;
```

- **setPlaybackRate()**
  - Sets the current playback rate

- Parameters: rate of type Number
- Return Value: None
- Usage:

```
setPlaybackRate(rate: number): void;
```

- **getSupportedKeySystems()**
  - Returns a list of key system the video engine supports
  - Parameters: None
  - Return Value: String[]
  - Usage:

```
getSupportedKeySystems(): string[];
```

- **setProtectionSchemeInterface()**
  - Enables Player to specify the handle to the the IProtectionSchemeHandler
  - Parameters: IProtectionSchemeInterface of type Object
  - Return Value: None
  - Usage:

```
setProtectionSchemeInterface(IProtectionSchemeInterface: object);
```

- **setVideoMute()**
  - Black out the video for parental controls
  - Parameters: enabled of type Boolean
  - Return Value: None
  - Usage:

```
setVideoMute(enabled: boolean): void;
```

- **addEventListener()**
    - The method attaches an event handler
    - Parameters: name of type String, handler of type Function
    - Return Value: None
    - Usage:

```
addEventListener(name: string, handler: Function);
```

- **removeEventListener()**
    - The method removes the event handler
    - Parameters: name of type String, handler of type Function
    - Return Value: None
    - Usage:

```
removeEventListener(name: string, handler: Function);
```

2. IConfig interface:

IConfig interface includes the following properties,

- **initialBitrate**
    - Maximum initial bitrate (kbps)
    - Usage:

```
initialBitrate: number;
```

- **initialBuffer**
    - Minimum amount of buffer needed before playback (seconds)
    - Usage:

```
initialBuffer: number;
```

- **playbackBuffer**
  - Maximum amount of buffer during playback (seconds)
  - Usage:

```
playbackBuffer: number;
```

- **offset**
  - Start position offset (ms)
  - Usage:

```
offset?: number;
```

- **networkTimeout**
  - Network request timeout (ms)
  - Usage:

```
networkTimeout: number;
```

- **downloadBuffer**
  - Maximum amount of time to download ahead of playhead (seconds)
  - For example, with a downloadBuffer of 10s there will be 10 seconds of video or audio stored in javascript memory and not in a playback buffer
  - Usage:

```
 downloadBuffer: number;
```

- **minBitrate**
  - Minimum amount of bitrate (kbps)
  - Usage:

```
minBitrate: number;
```

- **maxBitrate**
  - Maximum amount of bitrate (kbps)
  - Usage:

```
maxBitrate: number;
```

- **preferredAudioLanguage**
  - The preferred audio language
  - Usage:

```
preferredAudioLanguage: string;
```

- **timeShiftBufferLength**
  - TSB length in seconds
  - Value of 0 means it is disabled
  - Usage:

```
timeShiftBufferLength: number;
```

- **drm**
  - DRM of type DRMConfig
  - Usage:

```
drm: DRMConfig;
```

Pending

- DRMConfig definition & examples
- Client and Server side Digital Ad Insertion (DAI)