# UVE ATSC Extensions

## Background & Design Principals

* UVE APIs are mature, in wide use and integrated with tooling/metrics.
* UVE APIs expose full functionality of our native IP player.
* Lightning Apps to work with UVE API directly as top Level API

## Stack Overview

|  |
| --- |
| [http://cdn.metrological.com/lightning-spark/<appname>/index.html](http://cdn.metrological.com/lightning-spark/%3Cappname%3E/index.html)Lightning App/Framework Launcher |
| appBundle.js // app code and resources |
| lightning.js – snapshot of lightning SDK | UVE APIsbindings (injected bundle, in-process)UVE APIs (out-of-process for DTT) |
| WPE Browser | AAMP | Mediarite/Futarq |
|  | GStreamer |
| AmLogic |

## Proposed App-Facing UVE APIs for integration with Futarque/Mediarite Stack

### Channel Map Scanning

 playerInstance.addEventListener("channelMapChanged", function );

Asynchronous notification that channel map is available or changed.

Note: most TVs expose “scanning” as an expensive, user-triggered operation.

How to expose state for scanning – i.e.

“scan-needed” version-based?

“scan-in-progress”

“scan-complete”

Need to collect and take into account UX requirements.

 Most recent scan will be persisted by lower layers across tv reboots; resident app not expected to have to independently cache/persist.

 with sample handler:

function \_channelMapChangedHandler( event )
{

 switch( event.state )

 {

 …

 }

}

playerInstance.getChannelMap ( filter )

 where filter is optional json object supporting:

 empty to request full channel map

 “sourceid” – sourceid or array of sourceids to filter to corresponding channels

Returns array of json channel descriptors including attributes:

[

{

 “locator”:<string>, // opaque locator, i.e. “atsc://freq=#&prog=3”

 “channel”:<string>, // unique channel number, suitable for onscreen presentation; can be simple number or two-part <major>.<minor>

 “name”:<string>, // short channel name, i.e. 7 character station call letters

 Other fields (optional – could be useful for troubleshooting, but <locator> should suffice)

“service type”:<string>, // DTV, NTSC, audio-only, etc. “sourceid”:<number>,

 “tsid”:<number>,

 “frequency”:<number>, // MHz

 “modulation mode”:<number>

 “program number”:<number>

 “hidden”:<bool> // relevant for OTA channels?

Note: some competitor TVs are advertised as “cable ready” – assumption – not relevant for PlatCo

},…

]

### Inband Guide Data

 playerInstance.addEventListener("guideDataChanged", function );

Asynchronous notification that new EIT guide data is available.

 playerInstance.getGuideData( filter )

 where filter is json object with:

 “channel” – channel or array of channels of interest (leave empty for all)

 “start” – start time (UTC milliseconds) for program data; program data that ends prior to “start” will be filtered

 “end” – end time (UTC milliseconds) for program data; program data that airs after “end” will be filtered

Returns unpacked array of upcoming program data matching passed filter as JSON array

[

 {

 “channel”:<number> // channel identifier from guideData or channelMap

 “title”:<string>

 “description”:<string>

 “start”:<time> // utc start time milliseconds

 “duration”:<duration> // program duration in seconds

 “rating”:<string>

 “audioTracks”:<string-array> // for display purposes – but source of truth is what’s actually available

 “textTracks”:<string-array> // for display purposes – but source of truth is what’s actually available

 },…

]

Note: getGuideData expected to be highly performant, uncompressing relevant chunks of program titles and descriptions on demand.

App free to use brute force to collect “all guide data” for the likely <100 channels, or a subset.

Open Question: should guide data persist (by lower level firmware) across reboots?

### Tuning, ATSC (OTA)

 playerInstance.load(<locator>); // using locator string from getChannelMap

…

 playerInstance.stop();

 playerInstance.release();

Discovery, Local Digital/Analog Inputs

 playerInstance.addEventListener("localInputsChanged", function );

Asynchronous notification that local input sources available or changed.

playerInstance.getLocalInputs ( filter )

Returns array of json local input sources including attributes:

[

{

 “locator”:<string>, // opaque tunable locator, i.e. “hdmi://1”

 “name”:<string>, // name for source, i.e. “HDMI1” (stack-level)

 “type”:<string>, // i.e. analog, digital

 “state”:<string>, // informs whether given physical input is active/inactive

}

];

Q: are we planning to allow users to overwrite input names?

We control physical names of hardware inputs

Propose having any remappings be managed resident app layers.

### Tuning, Local Input

playerInstance.load(<locator>); // using locator from getLocalInputs

### Caption and Alternate Audio Track Discovery

playerInstance.addEventListener("textTracksChanged", function );

Asynchronous notification that audio track options have changed

getAvailableAudioTracks() // returns JSON formatted list of audio tracks

 “name” // unique

 “language”

other (IP Video Centric)

 “codec”

 “rendition”

 “characteristics”

 “channels”

playerInstance.addEventListener("textTracksChanged", function );

Asynchronous notification that text track options have changed.

getAvailableTextTracks() // returns JSON formatted list of text tracks

 “name“, // unique

“language“,

other (IP Video Centric):

“type” // CLOSED-CAPTIONS or SUBTITLES

“rendition“, // value of "GROUP-ID" attribute of X-MEDIA tag

“instreamId“, // refer X-MEDIA tag <https://tools.ietf.org/html/rfc8216>

“characteristics“ // refer X-MEDIA tag [https://tools.ietf.org/html/rfc8216](https://tools.ietf.org/html/rfc8216#page-54)

Is above relevant for HDMI inputs? Probably not. Example: on xbox would expect xbox controls to have to be used here.

### Caption and Alternate Audio Track Selection Selection

setAudioLanguage(lang) // legacy, using language code for selection

setAudioTrack(trackid) // pass name from getAvailableAudioTracks

setTextTrack(trackid) // pass name from getAvailableTextTracks

getAudioTrack() // currently presenting audio track name

getTextTrack() // currently presenting text track name

Video Output Configuration

### initConfig( config )

 // new parameters to be defined

Open Discussion:

* Q: is the Platco device is going to be advertised as “Cable Ready” or “Digital Cable Ready?” That may impose some regulatory requirements. Cable Ready means it can tune unencrypted QAM channels. Digital Cable Ready means it can support a cablecard.
* UVE apis can be shared with variant that works on desktop browsers, to support rapid prototyping in Chrome/Safari
* MediaRite/Futarque and integration needs to be open source, not a binary
* Assumption: injected bundle on Flex, with routing under the hood to out of process Futarque/MediaRite layers using JSON RPC
* Assumption: MediaRite APIs for DVR, Settings not in scope
* UVE APIs currently exposed using “AAMPMediaPlayer” name – will alias/change to something less IP Video-Centric, i.e. RDKMediaPlayer
* Llama Integration with Application Services (AS) layer
	+ Currently C++ code integration
	+ Peter notes that JS to C is common pattern, but not other way around
	+ We have similar challenge with Whirlpool (JavaScript Player Platform integration)

## Contacts:

* Praveen (Praveen.BiddappaAppachanda@sky.uk) – lead for ATSC
* Ann Harding – tracking ATSC; currently aligned with 2006\_sprint
* Sharma, Sudhanshu <Sudhanshu\_Sharma@cable.comcast.com>
* Peter Stieglitz (peter.stieglitz@sky.uk)
* Pedersen, Jan jan.pedersen@sky.uk
* Unnithan, Sreejith (Sky - Guest) Sreejith.Unnithan@sky.uk
* Bennett, Andrew Andrew.Bennett@sky.uk
* Vinod.Vijayakumar@sky.uk
* Kaagaard, Jesper jesper.kaagaard@sky.uk
* Jensen, Jens Kristian jens.jensen@sky.uk
* Matt Barbour
* Ben Greenberg

## Appendix

<https://www.atsc.org/atsc-documents/a652013-program-and-system-information-protocol-for-terrestrial-broadcast-and-cable/>

<https://www.atsc.org/atsc-documents/a69-program-and-system-information-protocol-implementation-guidelines-for-broadcasters/>

### User Stories

[AMLOGIC-60](https://ccp.sys.comcast.net/browse/AMLOGIC-60) [PlatCo] ATSC Stack Integration Phase II

[CPC-2687](https://ccp.sys.comcast.net/browse/CPC-2687) “ATSC Tuning and SI/PG Extraction”

[RDK-26053](https://ccp.sys.comcast.net/browse/RDK-26053) Productize DTT for XiOne (RealTek)