Fraunhofer FOKUS Institute for Open Communication Systems

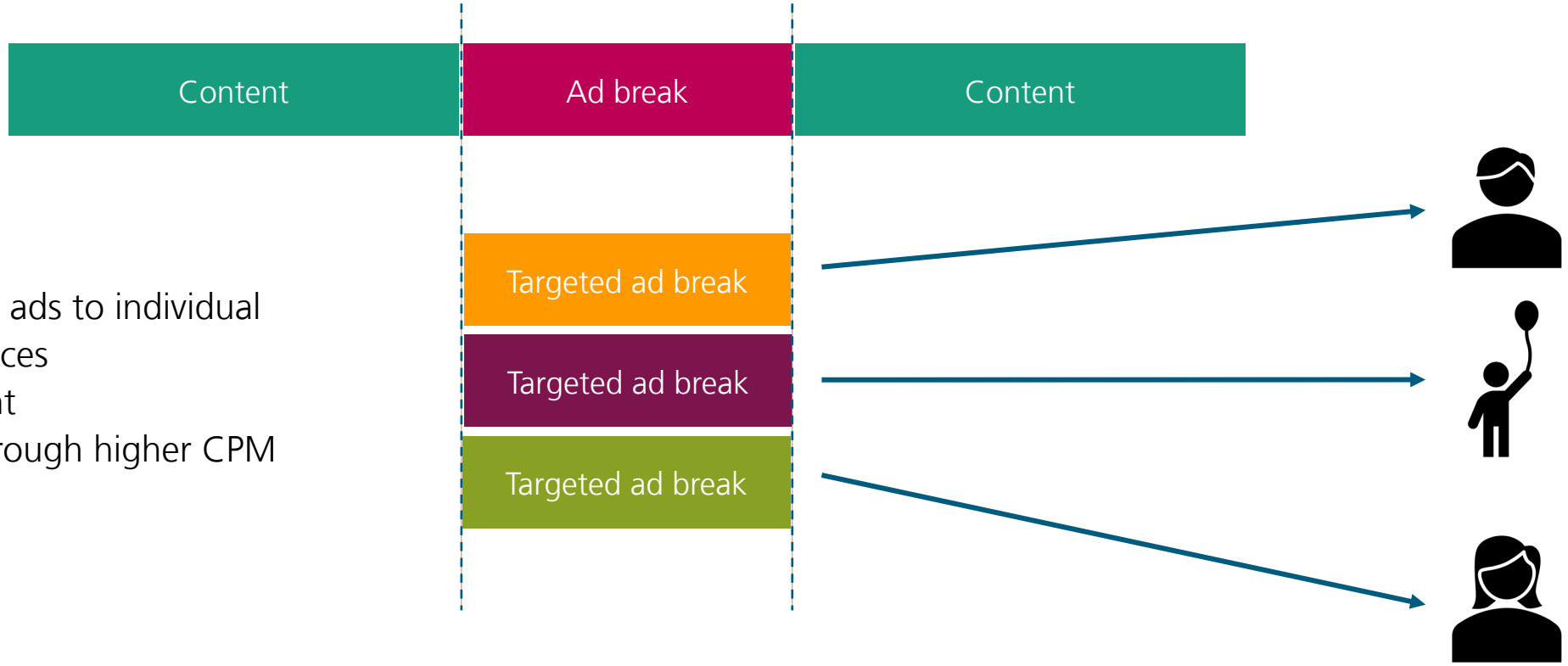
Advertisement in Broadcast and Broadband Environments

Robert Seeliger
robert.seeliger@fokus.fraunhofer.de

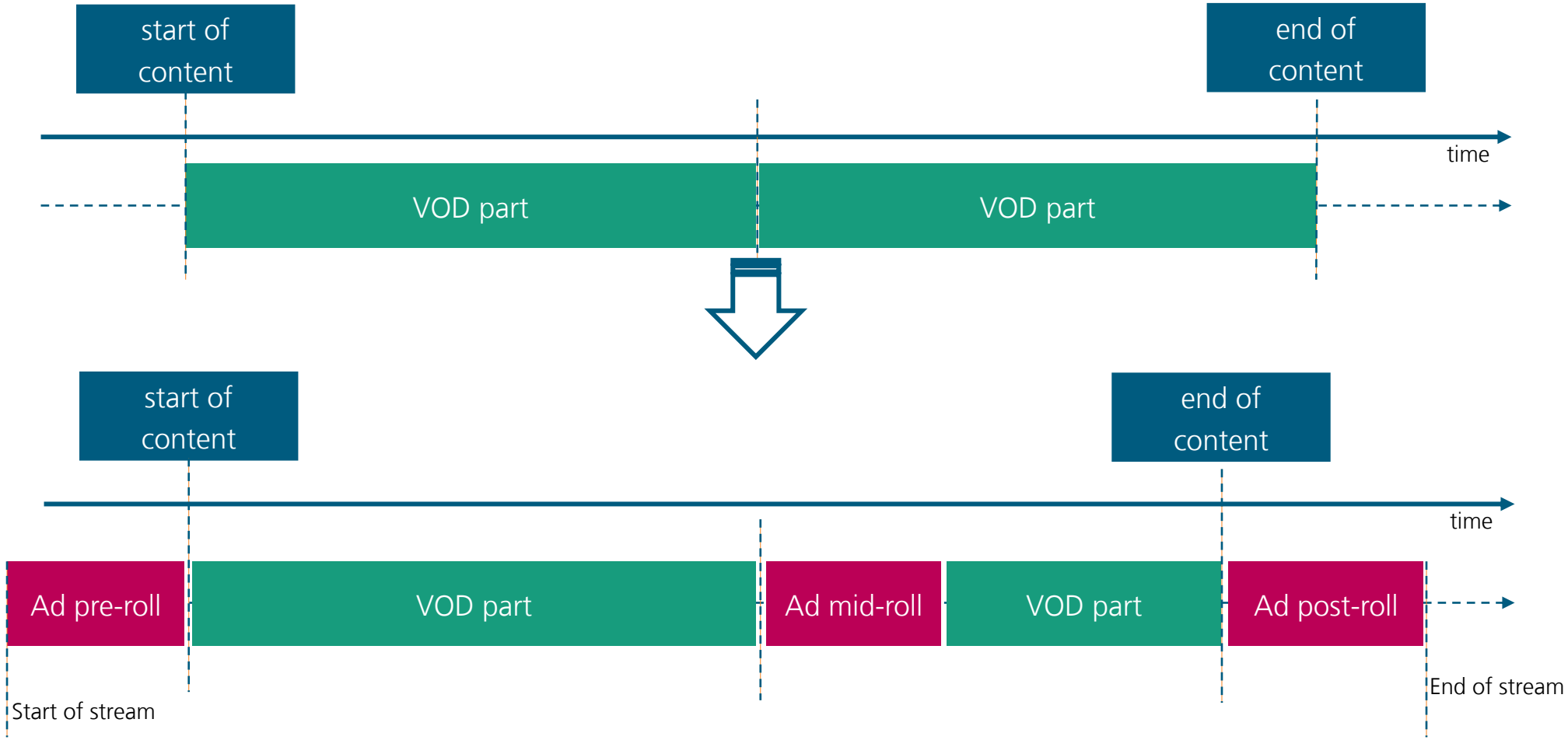
Fundamentals

Dynamic Ad Insertion

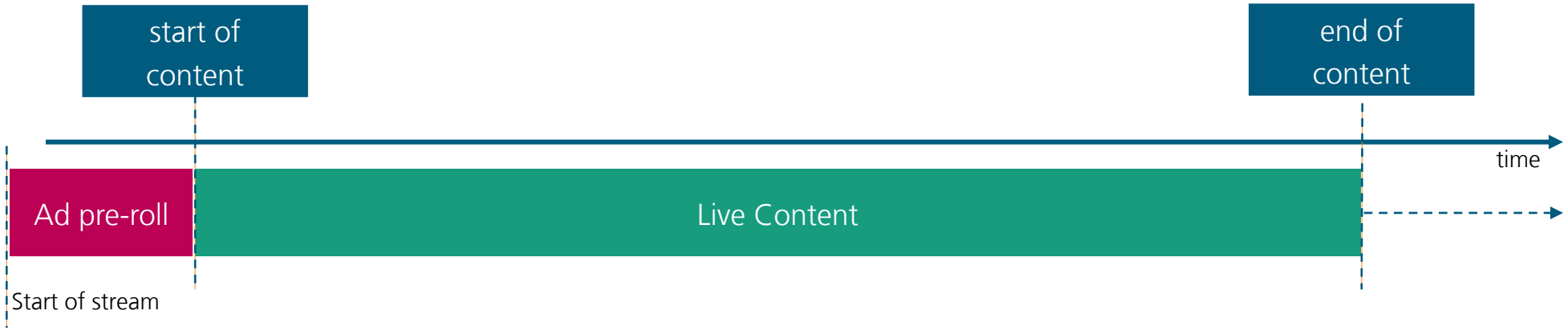
- provide targeted ads to individual users our audiences
- monetize content
- increase yield through higher CPM



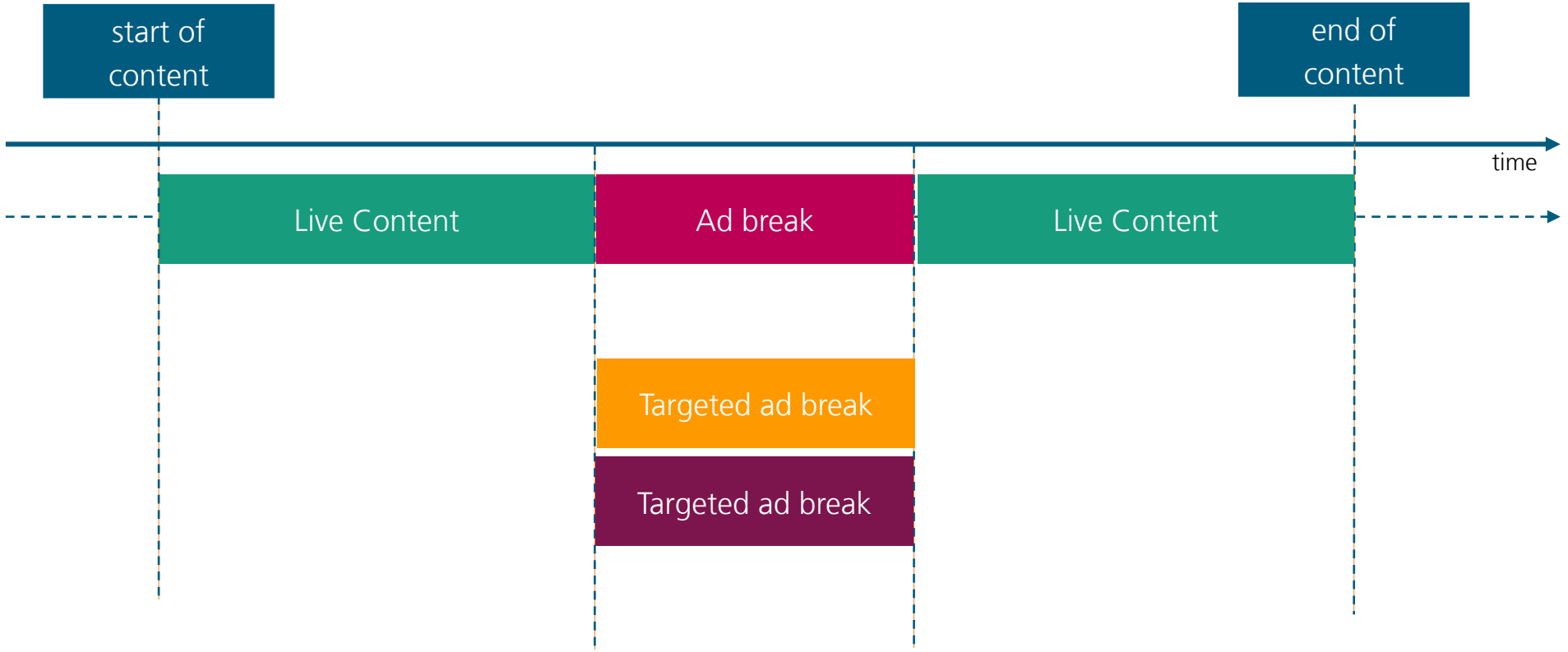
DAI Use Cases - AVOD



DAI Use Cases – Live Pre-Roll



DAI Use Cases – Live Ad Replacement



Ad Formats

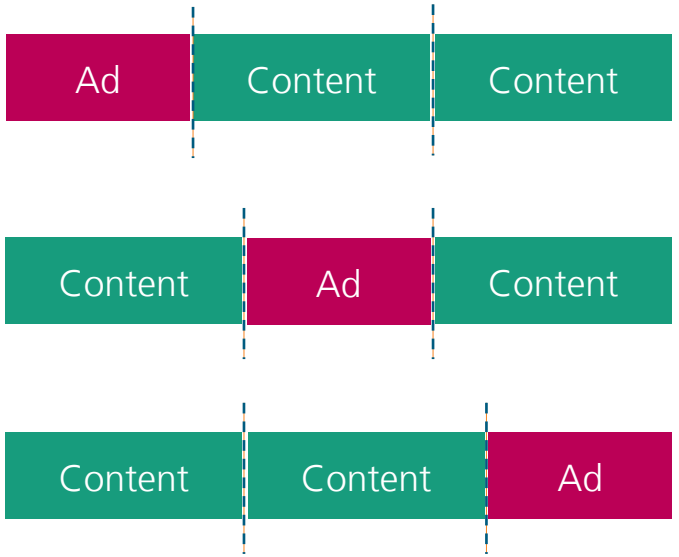
Ad Formats – Classification

- **Video Ads**
  - In-Stream
  - Out-Stream
- **Addressable TV**
  - Banner Ads
    - L-Shape Banner
    - Fullscreen
    - SwitchIn / SwitchIn Spot
    - Click to Video

Quelle: iStock.com/kckate16

Ad Formats – Video Ads

- **In-Stream**
  - placed within a publisher's online video, CTV, and app content
  - Pre-Roll
    - Stream immediately before the original content
  - Mid-Roll
    - Stream in the middle or any given position of the original content
    - Similar experience as conventional commercial break on TV
  - Post-Roll
    - Stream immediately after the original content
  - Bumper Ad
    - Short version of Pre-, Mid-, Post-Roll Ads
    - Typically achieves a high view-through rate

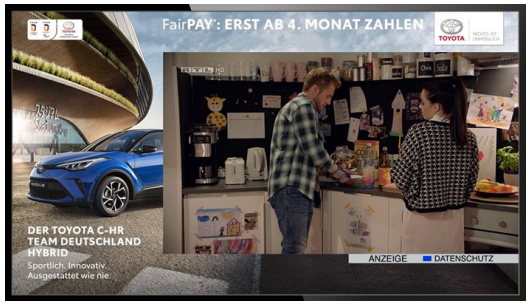


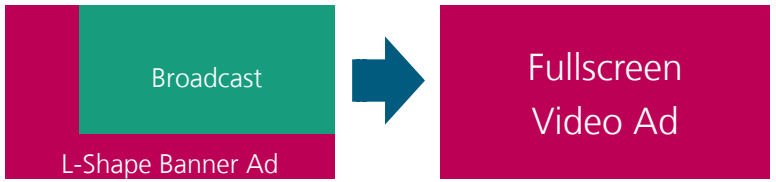
Ad Formats – Addressable TV

- Addressable TV ads are mainly display ads (banner)
- Might also contain video creatives in form of „click-to-video“
- File Type: JPG, PNG, (animated GIF)
- Interaction via color buttons on remote control → „click-to-action“

Ad Formats – Adressable TV

- **TV Advertising Formats**
  - L-Shape Banner
    - On HbbTV usually in form of a„SwitchIn“ display ad
    - Visible for a short period (e.g. ~10 seconds)
    - Shrinked broadcast signal
  - Fullscreen
    - Shows a fullscreen overlay display ad (1280x720)
  - Other SwitchIn formats
    - Zoom
    - Masthead
    - Spot & Bumper (video ad accompanied by a L-Shape Banner)
  - Click to Video
    - Combines display and video (dedicated color button to launch video from a banner ad)



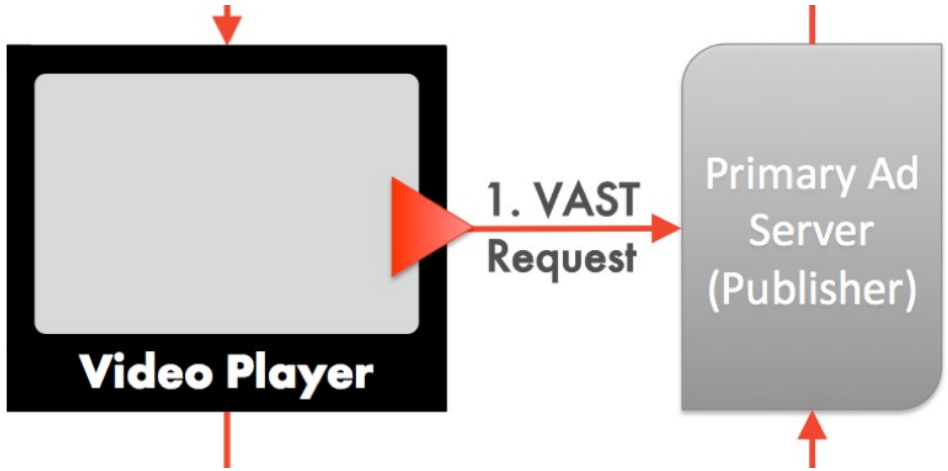


Terms, Abbreviations, Metadata & Standards

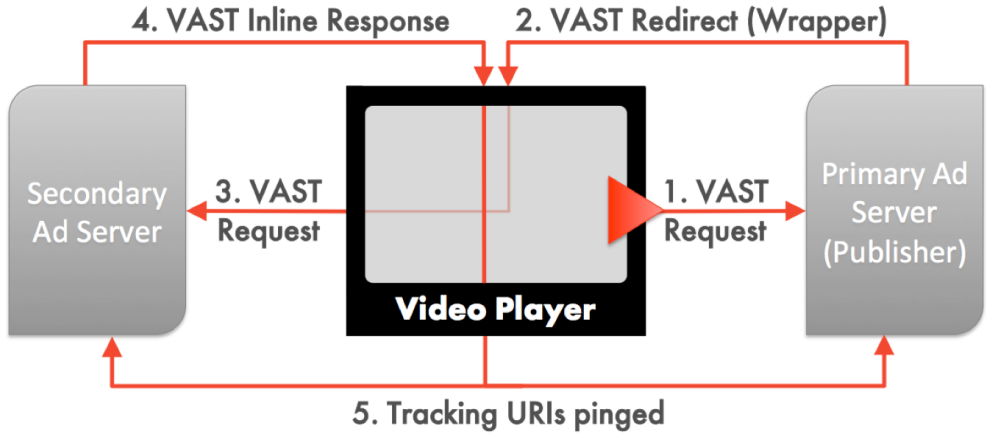
VAST - Video Ad Serving Template

- IAB standard since 2008
- template for structuring ad tags that serve video and audio ads to media players
- transfers important metadata about an ad from the ad server to a media player
- designed to standardize communication between video players and servers, facilitating traffic across all kinds of publishing platforms
- uses a XML schema
- Support for server-side ad insertion since Version 4.0
- Most recent version is VAST 4.3, released in October 2022

VAST



VAST with second ad server involved



How VAST works

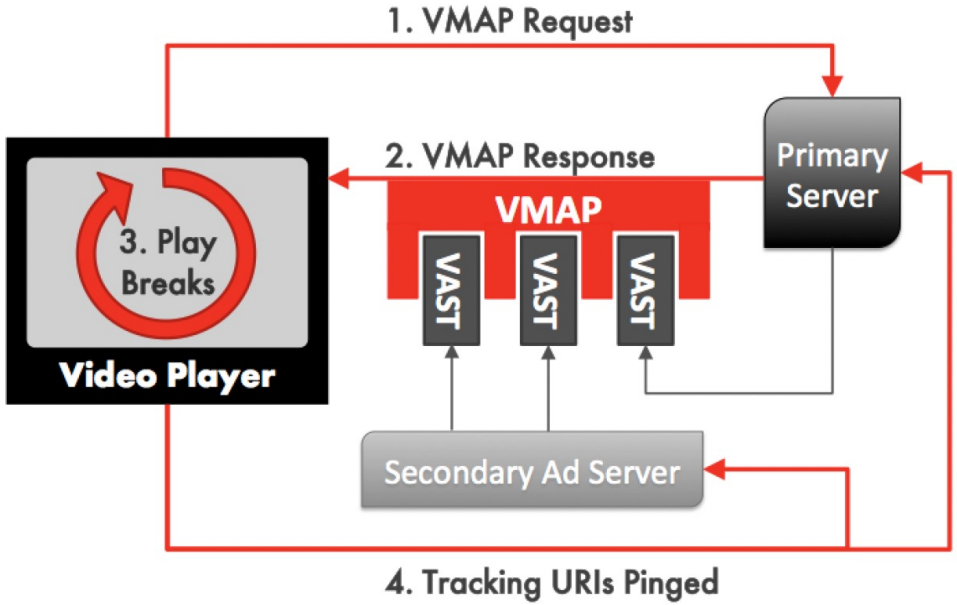
https://www.iab.com/wp-content/uploads/2015/06/VASTv3_0.pdf

VAST 4 - features

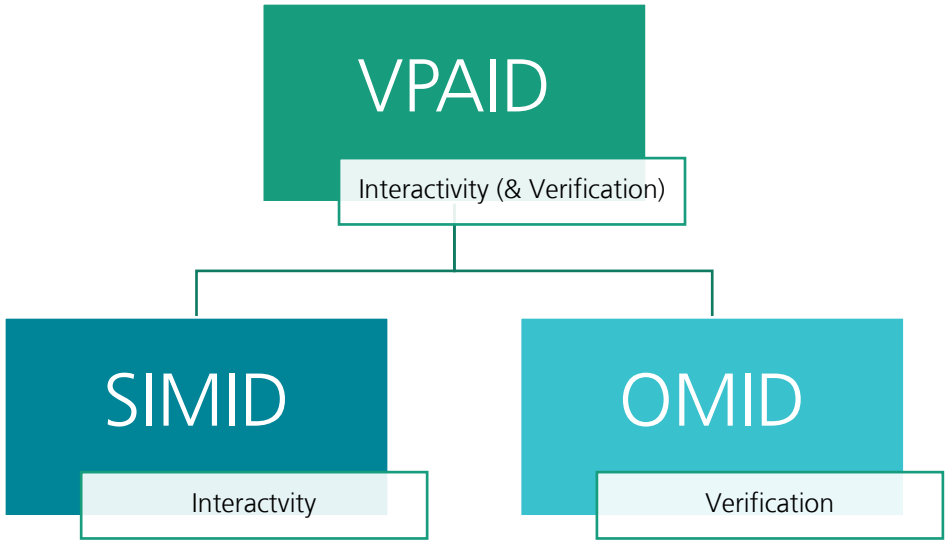
- **Separate Video File and Interactive File** to ensure more successfull display of ads across platforms as interactivity APIs got more ecomplex
  - SIMID is expected to be the Interactive File standard
- **Mezzanine File support**
  - Not eligiable to be used for ad display but important for SSAI workflows to generate ads matching the source specs
- **Universal Ad ID** `<UniversalAdId>`
  - used specifically for including a creative identifier that is maintained across systems
- **Ad Verification** and Viewability Execution
  - VPAID has been used by verification venfor measurement verification instead of using it for ad interaction as VPAID was intended.
  - VAST 4 offers a designated space `<AdVerifications>` for inserting ad verification APIs to be used in OMSDK
- To avoid being mistaken as fraudulent traffic, ad stitching providers must include with their ad tracking requests the following HTTP headers:
  - `X-Device-IP` set to the IP address of the device on whose behalf this request is being made.
  - `X-Device-User-Agent` set to the User-Agent of the client on whose behalf the request is being made.
    - Optional: `X-Forwarded-For` for backwards compatibility, although this is already deprecated since VAST4.1

VMAP – Video Multi-Ad Playlist

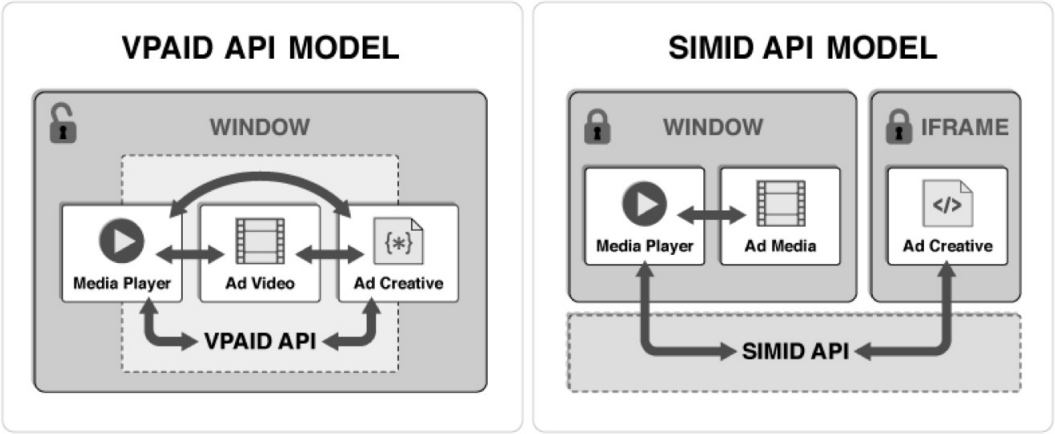
- XML template that video content owners can use to describe the structure for ad inventory insertion when they don't control the video player or the content distribution outlet
- Defines a structure for a playlist of video ads sent from an ad server to a video player
  - e.g. a set of timed ad breaks within video content (how many breaks, ad types, timing)
  - VMAP cannot provide ads directly → it wraps one ore more ad responses from an ad server defining which ads to display using VAST
- VMAP is NOT a replace for VAST – it is complementary to VAST → VAST can be served w/o VMAP



VPAID & SIMID

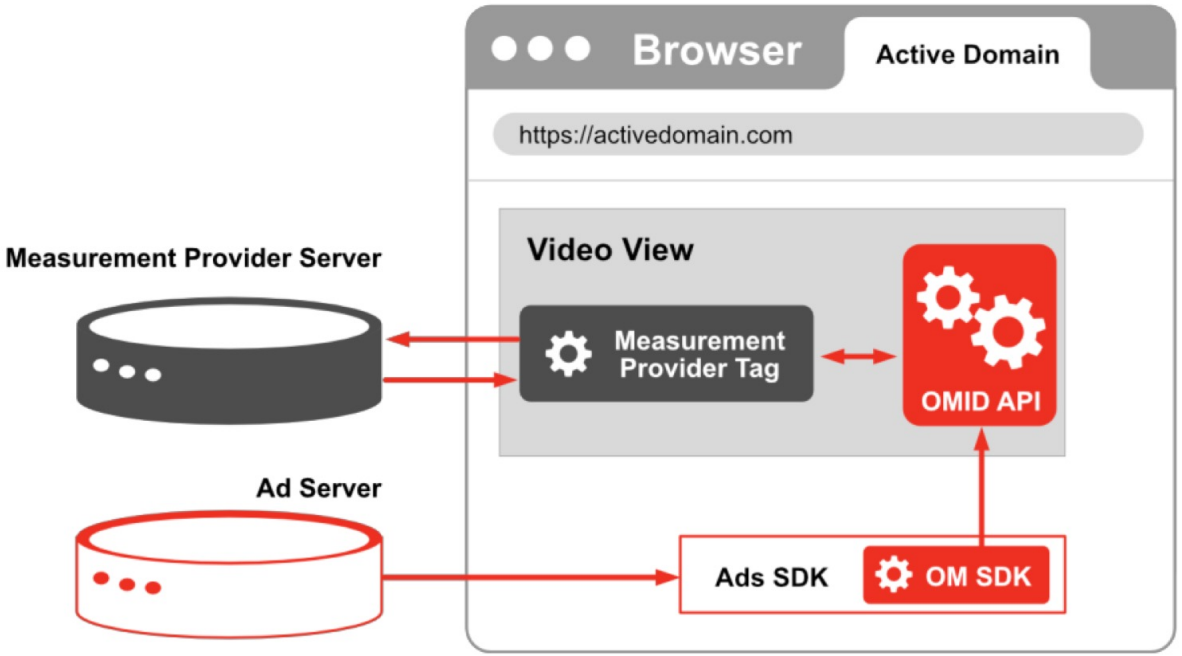


- Video Player Ad Interface Definition (VPAID)
  - specifies the protocol between the ad and the media player required to enable ad interactivity and other advanced video advertising functionality
  - allows for dynamic two-way communications, enabling the user, the video ad, and the video player to interact, so a rich interactive in-stream ad experience can be delivered
  - addresses how publishers discover various metadata assets related to an ad campaign
  - →VAID is DEPRICATED since VAST 4.2 and replaced by OMID (Verification) and SIMID (Interactivity)
- Secure Interactive Media Interface Definition (SIMID)
  - addresses how the publisher's media player should communicate and interface with a rich interactive layer and vice versa
  - clear separation of the interactive layer from the media asset



Open Measurement

- Open Measurement is an IAB standard that allows publishers to use third-party viewability providers to verify impressions and click measurements
- used to track mobile in-app advertizing → measure viewability
- IAB certifies companies for correctly using the Open Measurement Software Development Kit (OM SDK)
- The OM SDK itself is an implementation of the Open Measurement Interface Definition (OMID)
- Measurement vendors include e.g. AppNexus, Comscore, DoubleVerify, Integral Ad Science (IAS), Moat and others



https://iabtechlab.com/wp-content/uploads/2021/01/OM-Web-Video-Getting-Started-Webinar-Deck.pdf

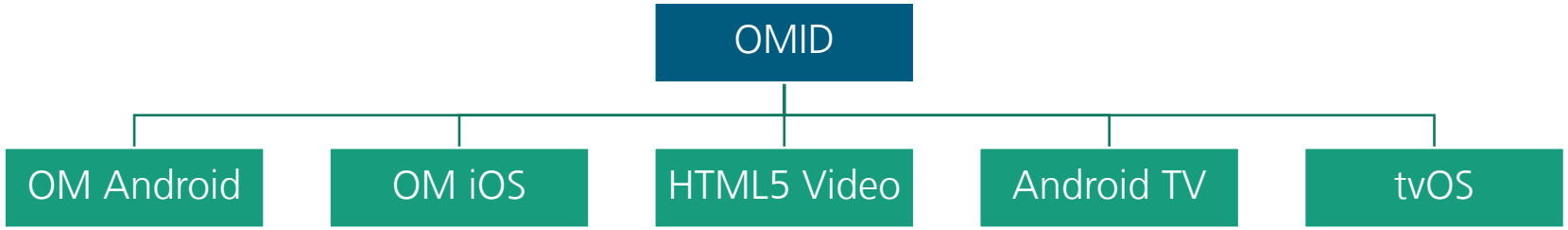
Open Measurement for CTV

- OM SDK has also been released Version 1.4 for CTV in 08-2022 (AndroidTV / tvOS / Amazon Fire)
  - identifying CTV Traffic: Know the environment in which a native app is running
  - Last Activity: Signal that an event occurred indicating someone is “still watching”
  - Display Connection Status: Understand when the TV display is off, but applications may still be running
  - Video Pod Measurement: Enable measurement of video ad pods with gapless playback for “javascript” session types
    - gapless pods of video ads were difficult to measure using the native OMID AdSession APIs because finishing one OMID ad session and starting another requires several roundtrips between the webview and native layer, which causes latency
- While some platforms may be able to adapt OM SDK for Web Video to work in their OS, others may need to support the development of a custom SDK.
- Currently not supported on

x Roku
x Samsung - Tizen
x LG-WebOS
x Vizio - Smartcast OS
x Web Cast Apps
x HTML5 Apps for CTV

Open Measurement Interface Definition (OMID)

- OMID defines an API that any third-party verification provider can use to effectively measure ads served to any mobile SDK that implements that API.
  - receives all signals collected by sites and apps via the OM SDK
  - Measurement service providers place their scripts in the ad creative in order to transmit those signals to their own servers via the OMID
  - OMID enables the communication between the OM SDK in the publisher's player and the service provider's measurement script
- Benefits:
  - Whith the use of OMID, there is no need for Video Player-Ad Interface Definition (VPAID) wrappers anymore
  - ad playback is more stable with fewer latencies, resulting in a better user experience
  - a more reliable measurement



Types of Dynamic Ad Insertion

CSAI – Client Side Ad Insertion

CSAI (Client-Side Ad Insertion) is the method of delivering ads to clients (desktop, mobile, CTVs, gaming consoles, etc.) where the client (video player) requests the ad server for an ad when it reaches ad-markers in the stream or in the manifest (HLS/DASH).

- After receiving the client's ad-request, the ad server returns an ad based on data collected from the client and other information (campaigns, preferences, etc.).
- The video player then pauses the video, plays the ad (or group of ads), and then resumes video playback (if any).
- It is also the responsibility of the client to report on the ad metrics (playback, quartiles, interactives, etc.)

CSAI – Client Side Ad Insertion

When the user presses play,

(1) The player downloads the manifest from the CDN and begins the video playback.

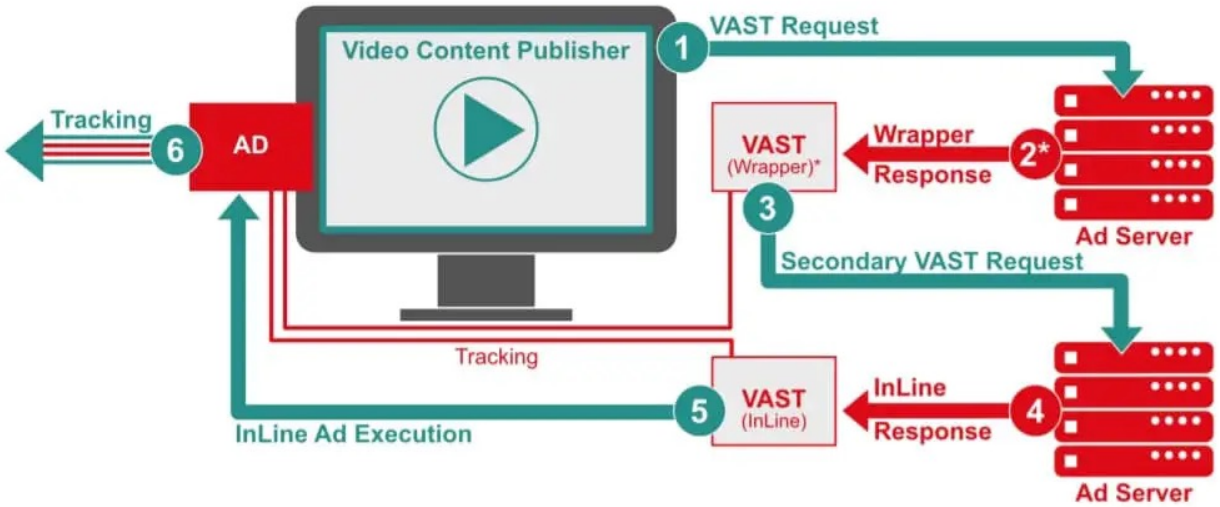
(2) In the case of a pre-roll ad, the ad begins to playback before the video.

(3) When the video player reaches an ad-marker, it pauses the video playback and makes an API call to the ad server requesting an ad for playback.

(4) Typically, the ad server will respond with a VAST XML (or VMAP, VPAID, etc.) containing information about the ad, the ad's media, and tracking information, etc. To learn more about VAST tags, click here, and click here for the VPAID tutorial).

(5) The video player uses the information received from the ad server and plays back the ad. After the ad playback, the video player continues playing the video stream.

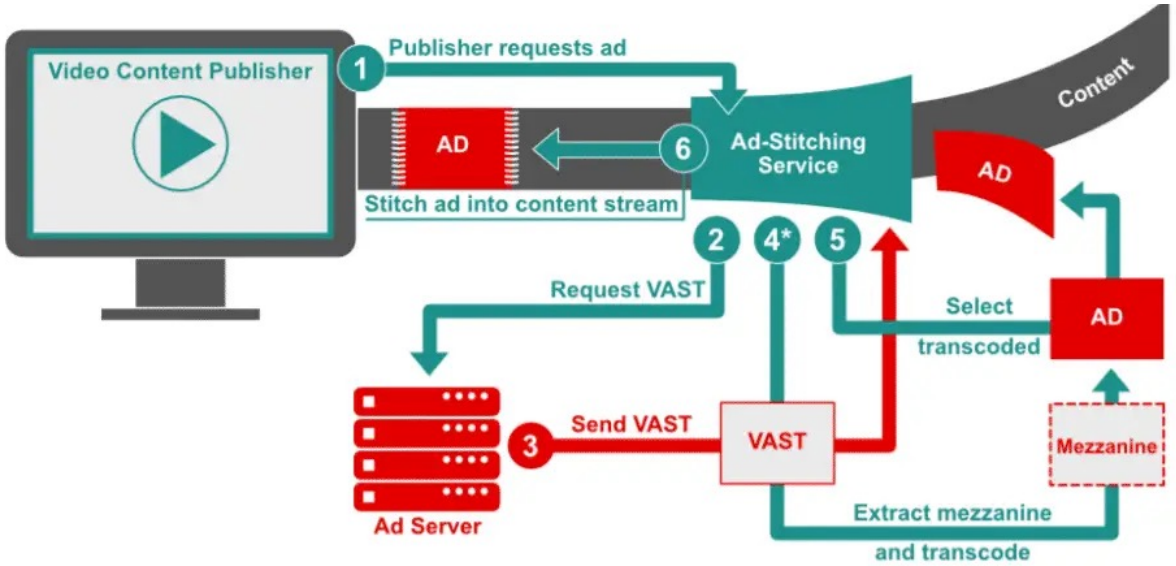
(6) Streaming platforms usually embed their video players with SDKs provided by 1st or 3rd party ad verification services. These SDKs track the position of the ad playback, completion rates, errors, etc., and report back to servers periodically with this information.



https://iabtechlab.com/wp-content/uploads/2019/06/VAST_4.2_final_june26.pdf

https://ottverse.com/csai-vs-ssai-client-side-server-side-ad-insertion/

SSAI



When the user presses “play”:

(1) the video player will get the HLS or DASH manifest from the SSAI service.

(2) When the player asks for the manifest, it usually provides some information about itself (i.e., the viewer’s location, subscription-levels, etc.) to enable personalized ad-delivery.

1. Players generally provide the “**X-Device-User-Agent**” and “**X-Device-IP**” HTTP headers, as per the IAB VAST guidelines.
2. Additional data can be provided if the player is set up to do so by the SSAI vendor.

(3) SSAI contacts the Ad Decisioning server, whose job is to supply the ad for insertion.

(4) The Ad Decisioning Server uses various bits of information such as the data sent from the player at the start of the playback session, ad campaigns, etc., to get an ad out of the inventory and pass it on to the Ad Insertion Server.

(5) The Ad Insertion Server transcodes the ad to match the video’s bitrate ladder, packages the ad, and produces a manifest for the ad, and stitches the ad’s manifest into the video’s manifest.

(6) When the player gets this refreshed manifest, it seamlessly delivers the ad as if it were delivering any other video!

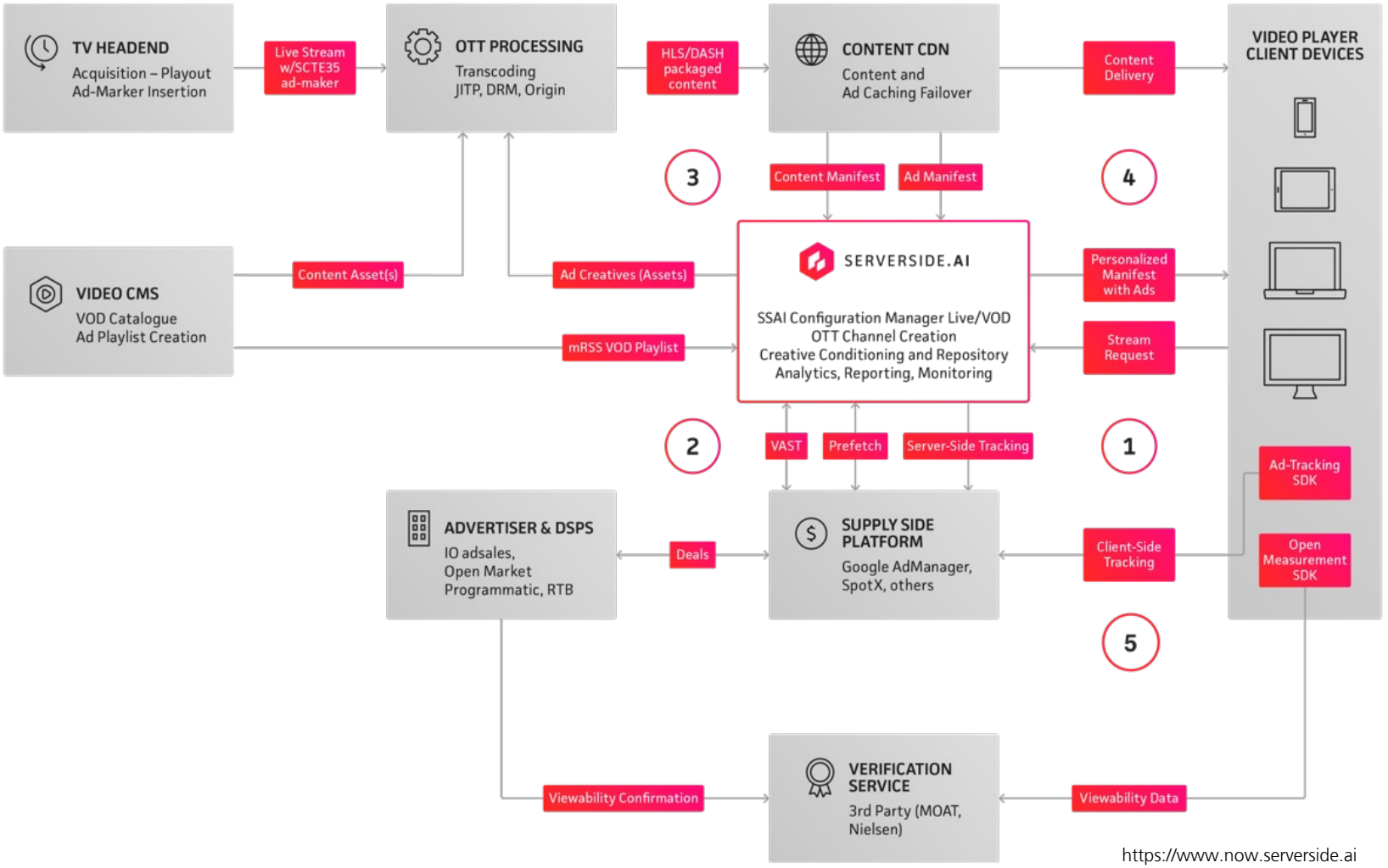
(7) After the ad completes, the video player continues playing the content without any interruptions or breaks.

(8) Apart from this, the player can be modified to report on the ad playback to ad-tracking services. i.e., client-side reporting for server-side ad insertion.

https://iabtechlab.com/wp-content/uploads/2019/06/VAST_4.2_final_june26.pdf

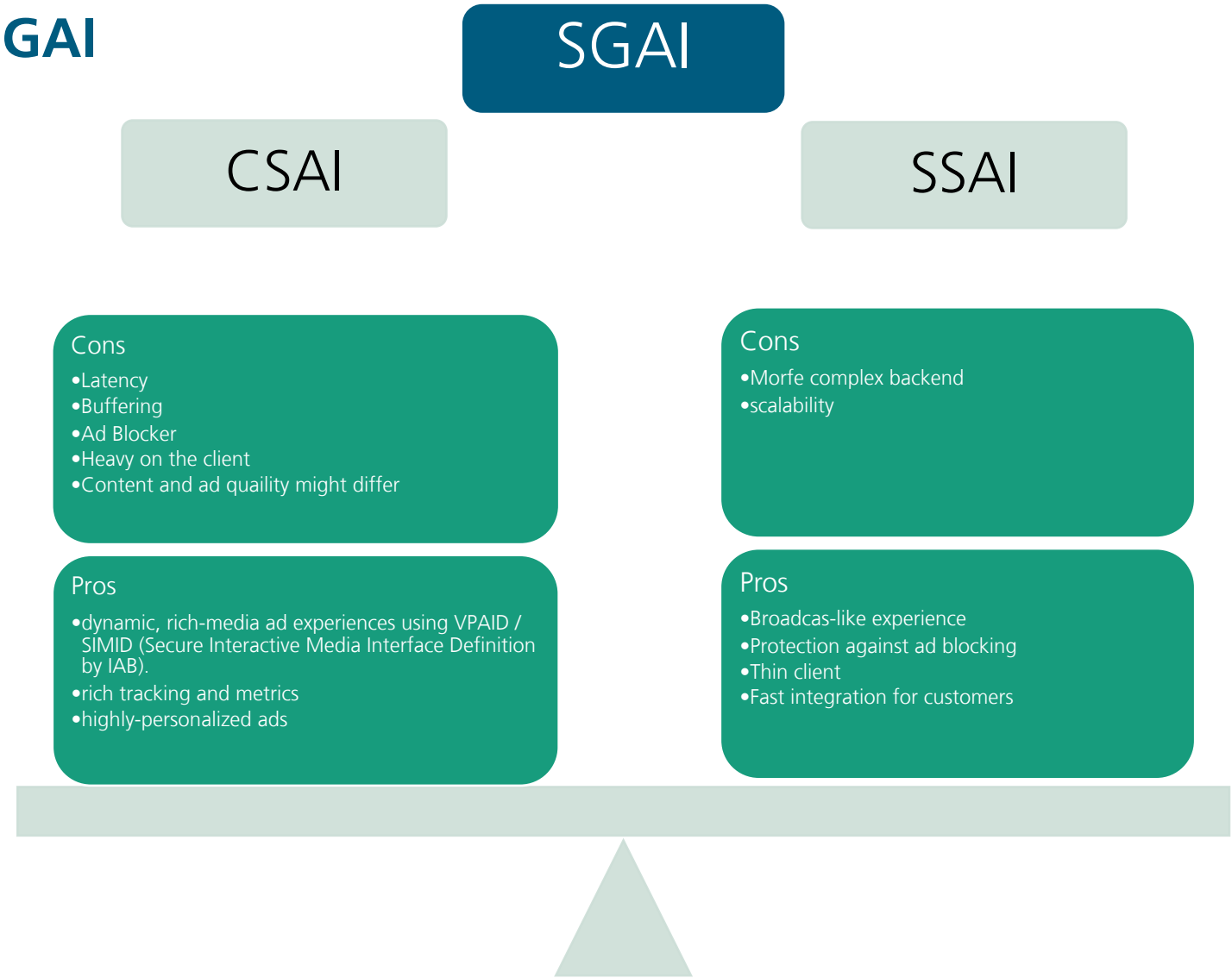
https://ottverse.com/csai-vs-ssai-client-side-server-side-ad-insertion/

SSAI System Overview



https://www.now.serverside.ai

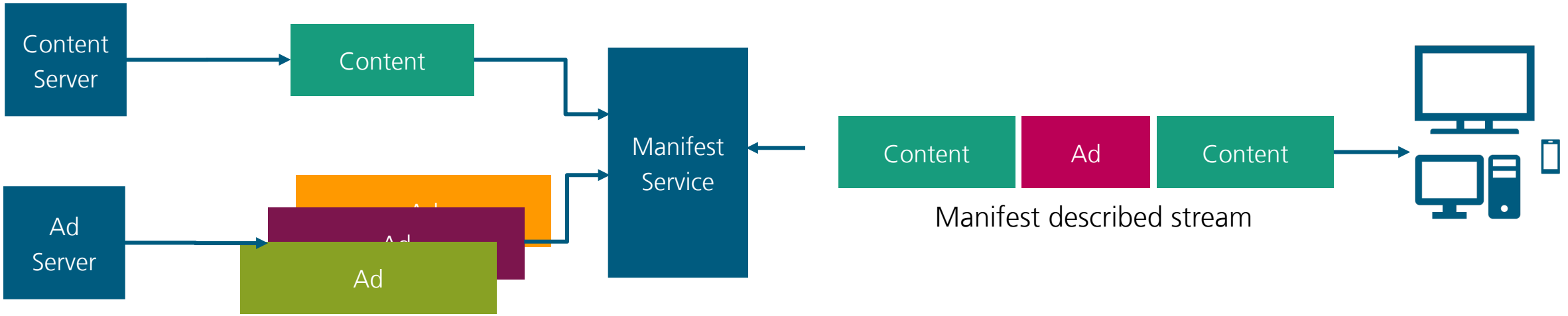
CSAI vs SSAI vs SGAI



Server Guided Ad Insertion

Server Side AD Insertion

- systems must scale with number of clients
- Level of targeting impacts cacheability
- wasted ad inventory

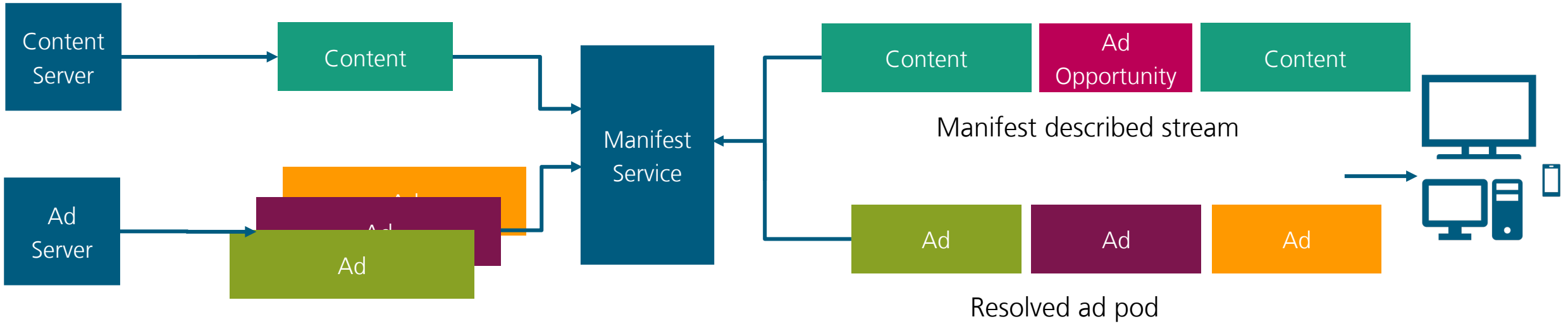


Server Guided Ad Insertion

- SGAI reflects the pros of both CSAI and SSAI and attempts to limit the cons
- the server outlines insertion opportunities into a common manifest for the main content, with instructions for the client on how to reach out to the ad-server
- On the client application, when the player reaches near an insertion point, it attempts to resolve the ad for playback based on the instructions.
- Client app reaches out to the ad-server and receives the ad manifest to insert, and queues up the segments just like it does for the main content
- this is highly scalable, cost-effective, and reduces complexity in client decisioning logic
- It also allows for seamless transition between main content and ads, and can also be applied to multiple kinds of auxiliary content beyond just ads

Server Guided Ad Insertion

- Scalability
- Isolate video streaming systems (ads and main content)
- Seamless ad experience
- keep targeting granularities



DASH-IF Interoperability Points; Part 5: Ad Insertion in DASH

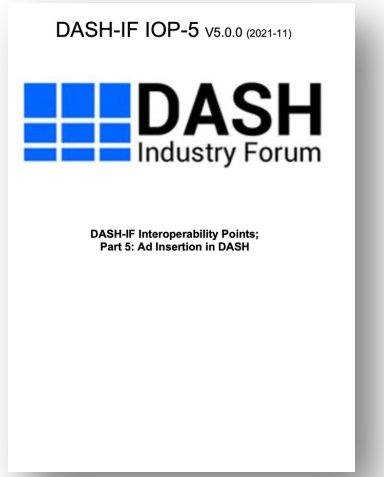
Server Guided Ad Insertion

VOD

- Remote Periods via DASH XLINK

Live

- Patch Manifests



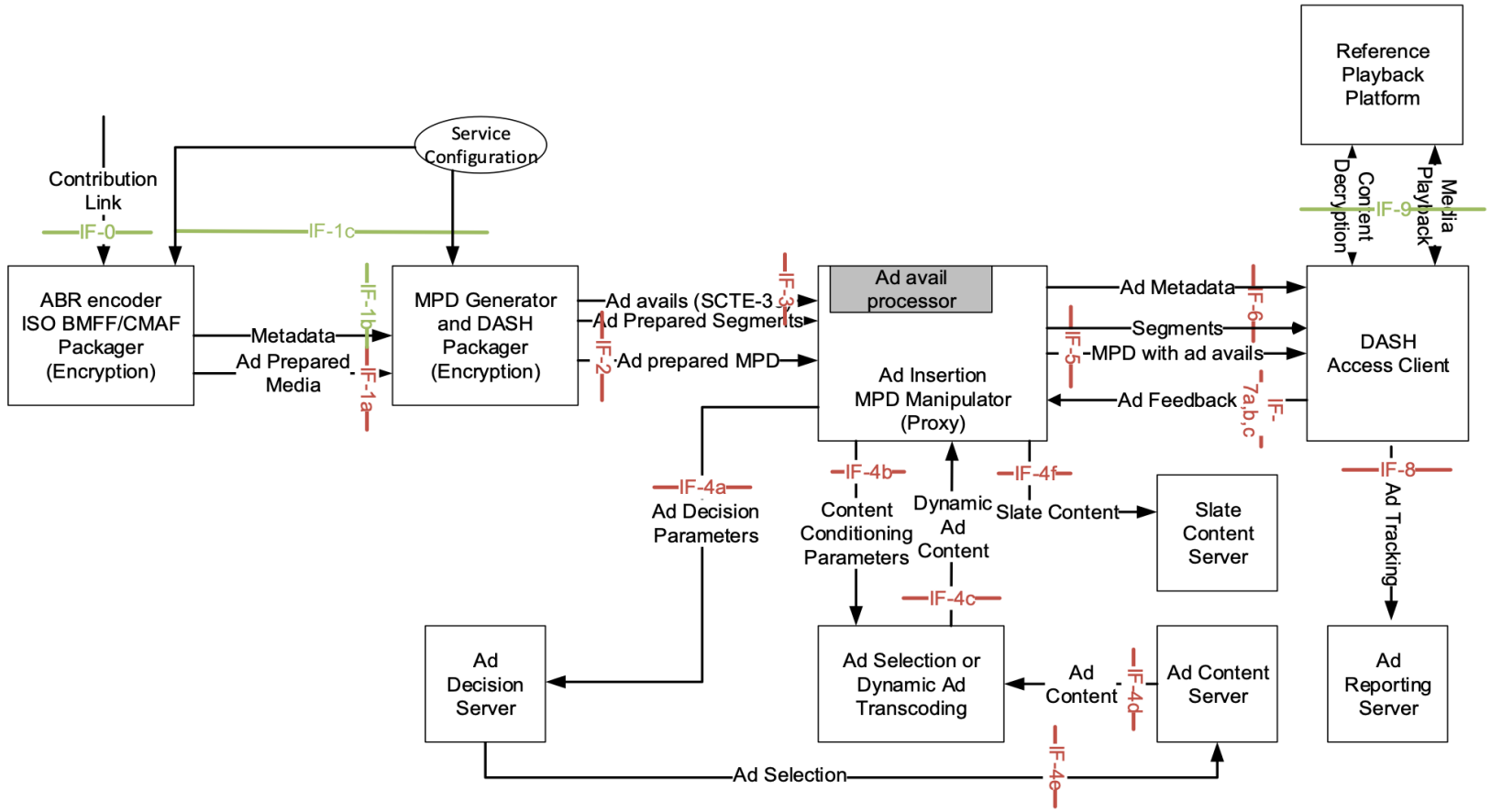
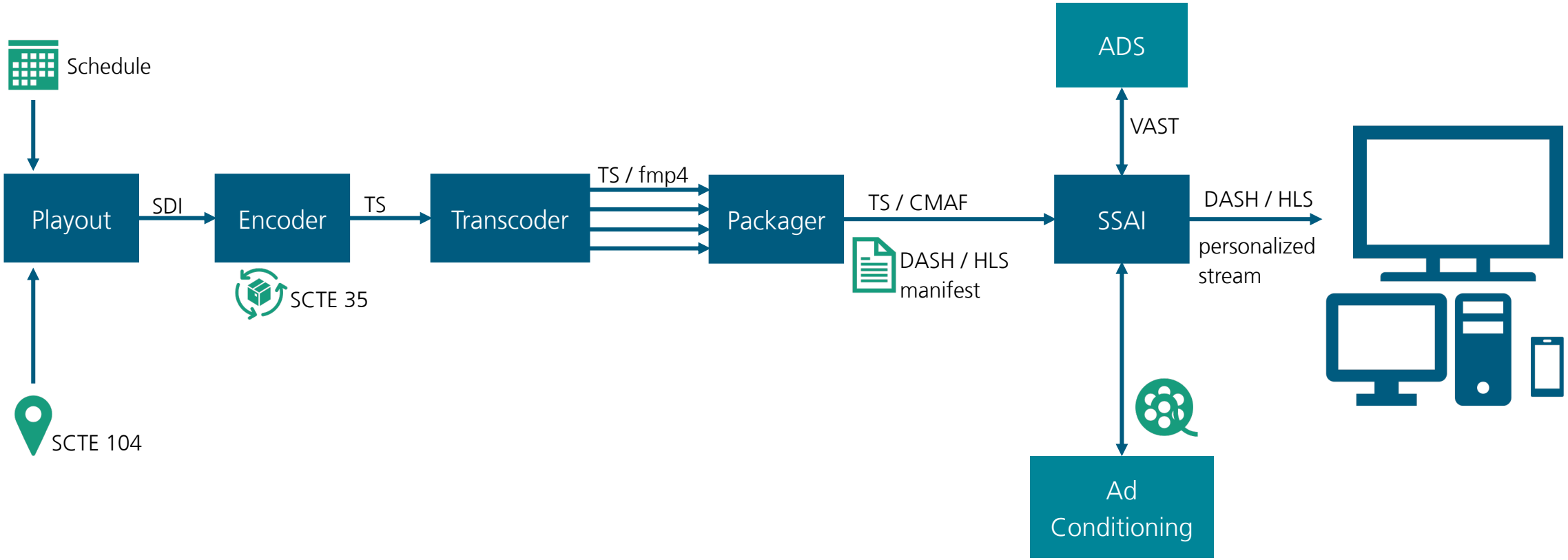


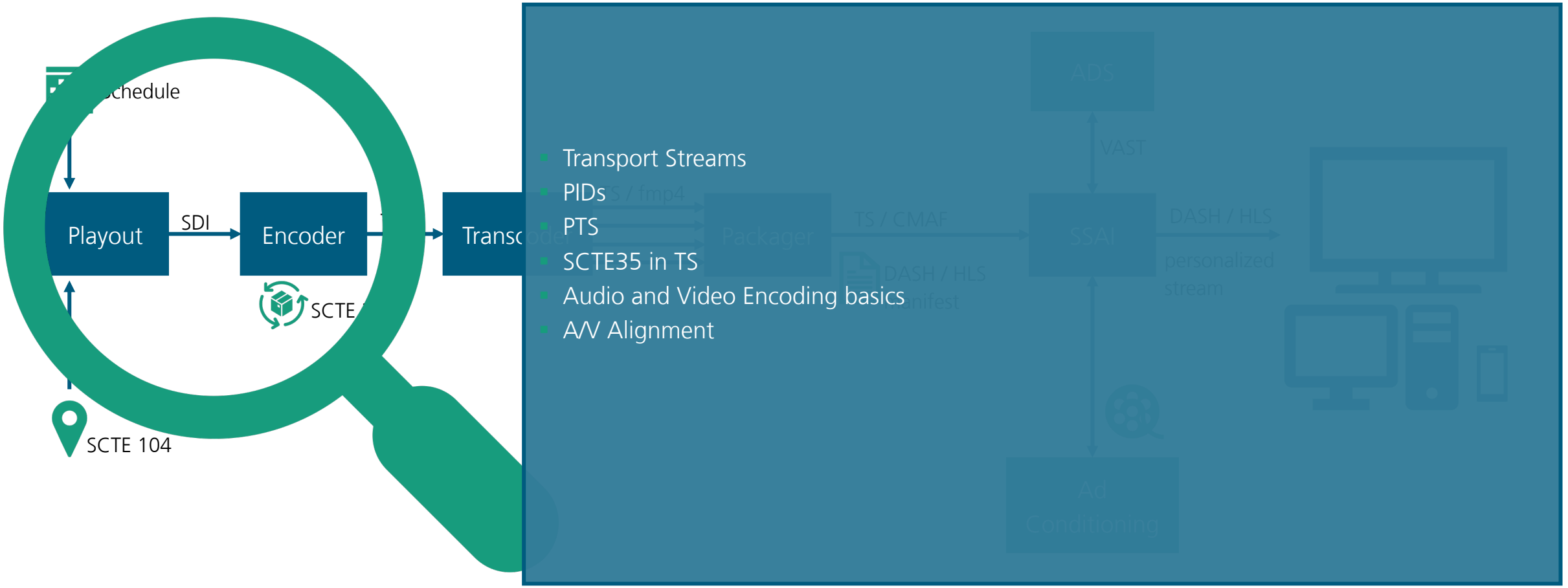
Figure 1 DASH-IF Ad Insertion Architecture

Content and signaling lifecycle for ad-enabled OTT Streaming

Content and signaling lifecycle – live linear

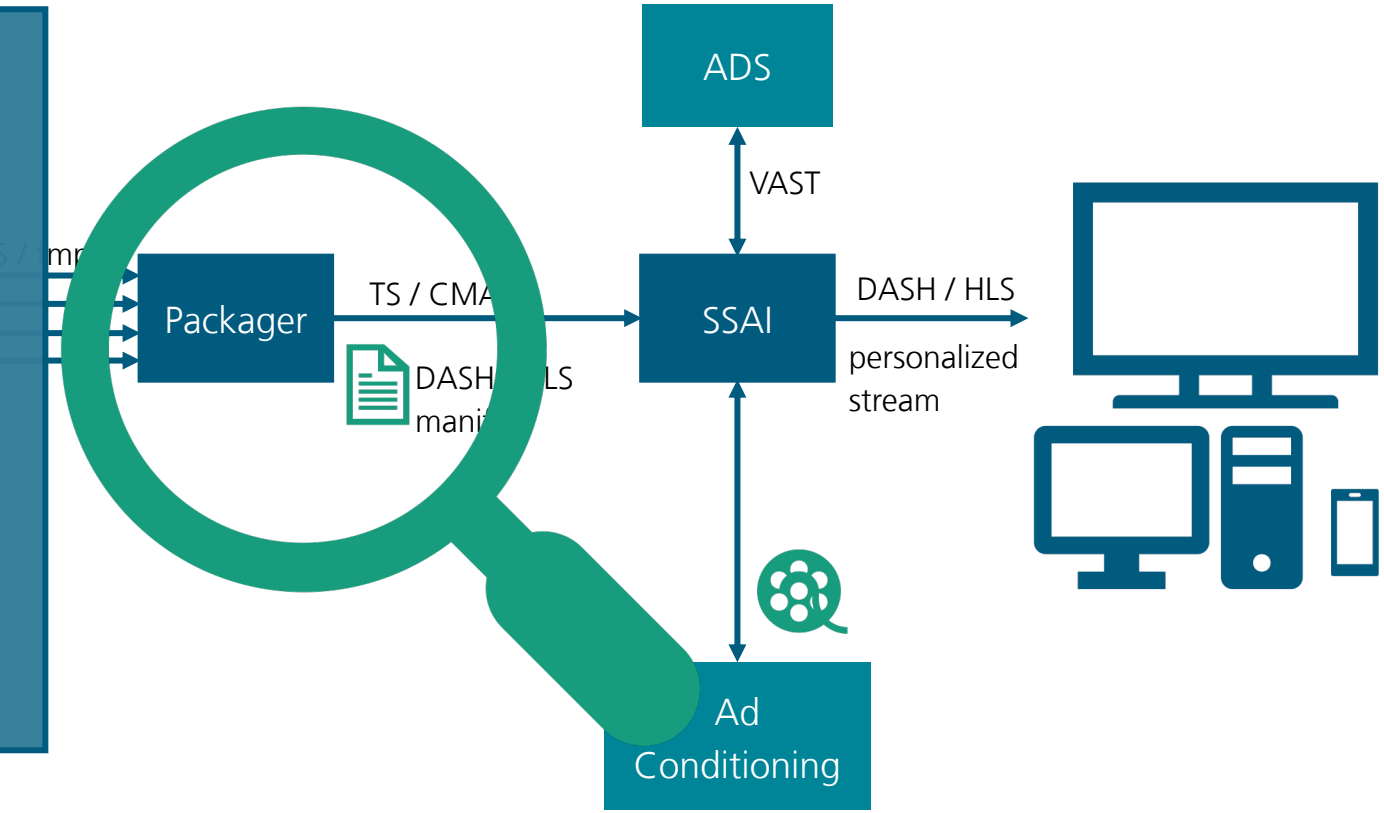


Content and signaling lifecycle – live linear

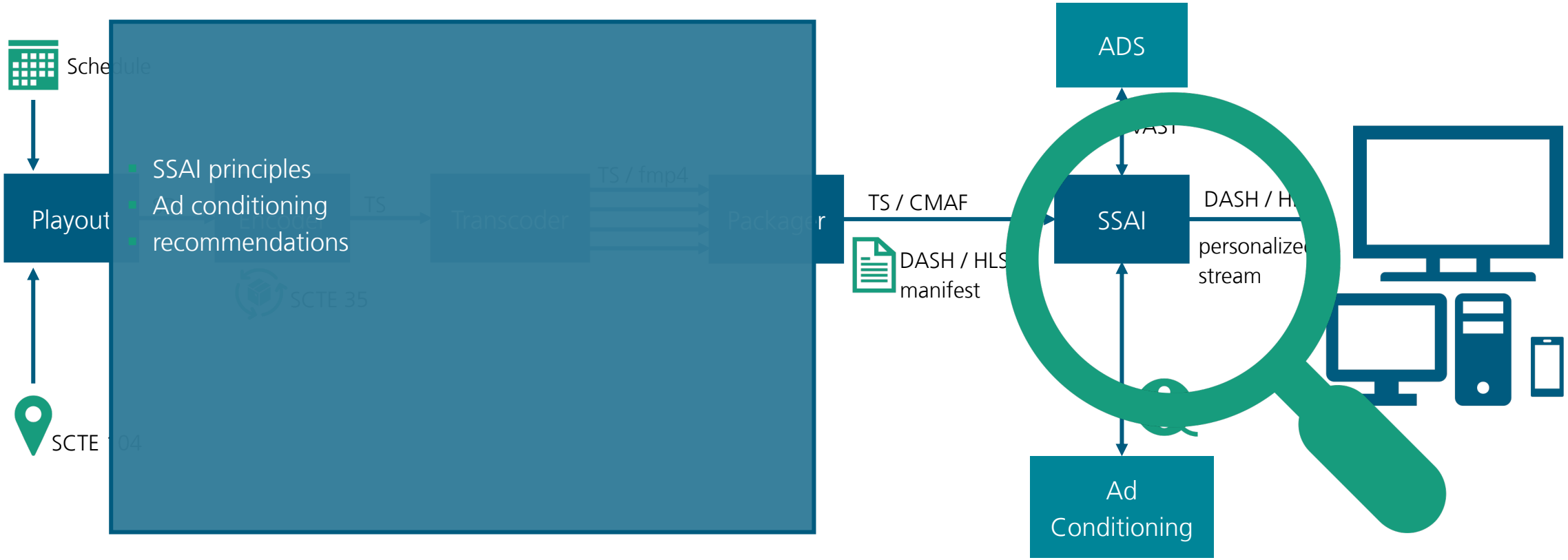


Content and signaling lifecycle – live linear

- Manifest Decoration to allow Ad Insertion
- SCTE35 styles and signaling in HLS & DASH
  - #EXT-X-CUE-OUT
  - #EXT-OATCLS-SCTE35
  - #EXT-X-DATERANGE
  - #EXT-SCTE35:CUE-OUT
  - SCTE binary messages
  - Multi- vs single period DASH with ad signaling
  - Event streams
- IDR Frames & Segment Boundaries

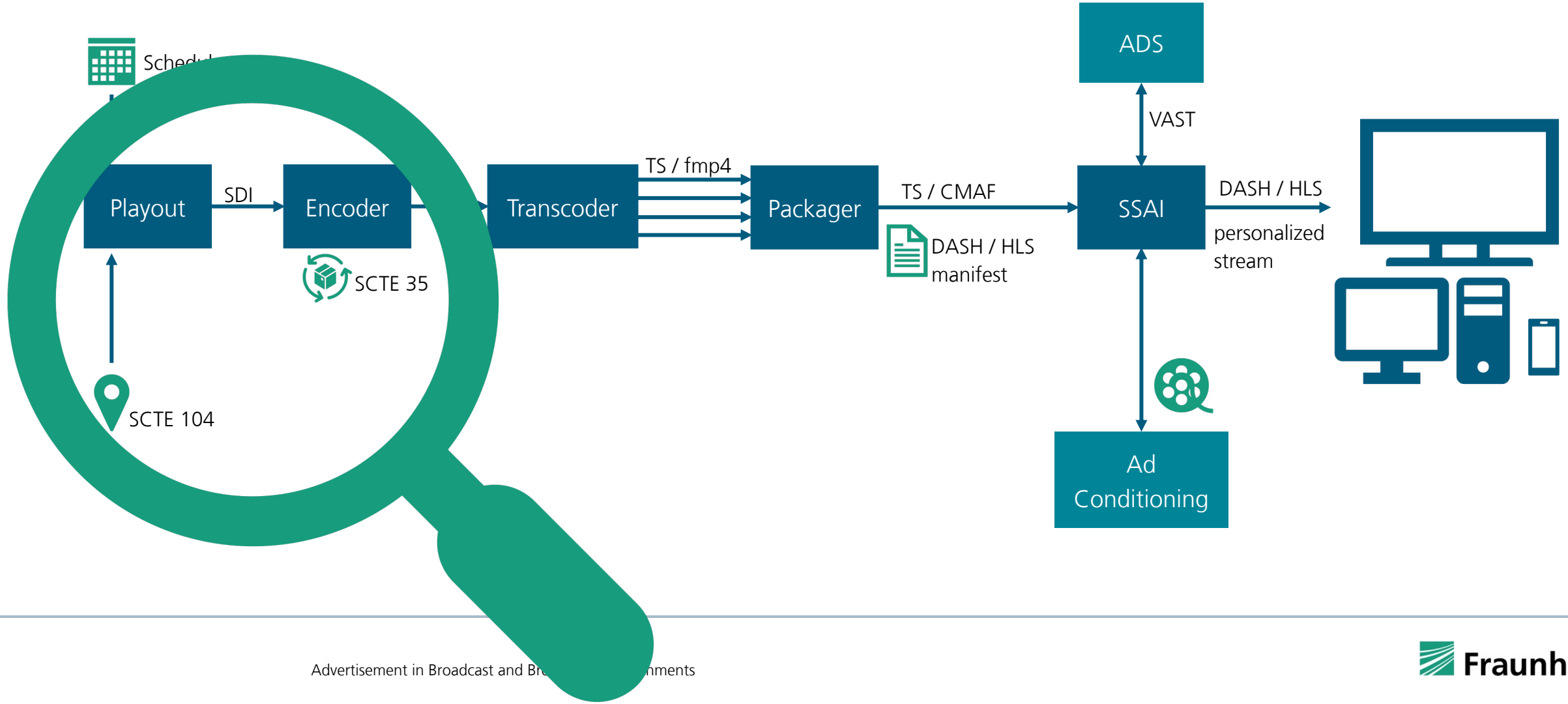


Content and signaling lifecycle – live linear



SCTE in Broadcast and Broadband Environments

Content and signaling lifecycle – live linear



What is SCTE

- Society of Cable Telecommunications Engineers
- ANSI-accredited platform developing technical specifications supporting cable telecommunications
- part of CableLabs since 2020
- SCTE standards
  - Energy management
  - Data communications
  - Equipment & cabling
  - Network operations
  - DIGITAL VIDEO
    - ✓ IP video transport
    - ✓ Streaming
    - ✓ Ad Insertion
    - ✓ Audio levels & synchronization

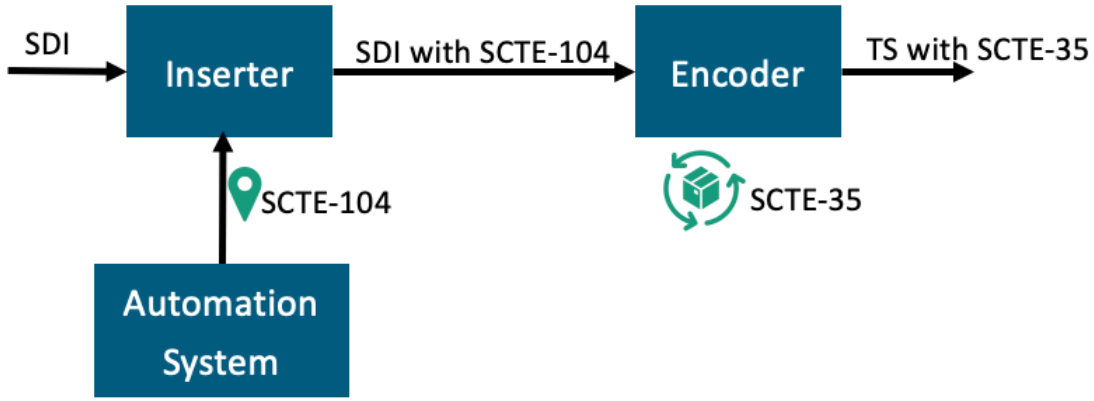
SCTE-104

Advertisement in Broadcast and Broadband Environments

Fraunhofer FOKUS

SCTE-104 messaging

- widely adopted throughout the broadcast industry
- standard method for including specific program automation signals and markers in video
- SCTE-104 is used with baseband signals (e.g.SDI)
- Typical messages cover:
  - program parts
  - commercial breaks



- SCTE 104 markers are inserted by an automation or playout system prior to sending the baseband video feed to the inserter.
- Once the feed reaches the encoder, those markers get translated into compressed stream markers to be transmitted with the content (e.g. a MPEG-TS). The compressed markers are what SCTE-35 defines.
- SCTE 104 messages are sent between systems via TCP/IP

SCTE-35

Advertisement in Broadcast and Broadband Environments

Fraunhofer FOKUS

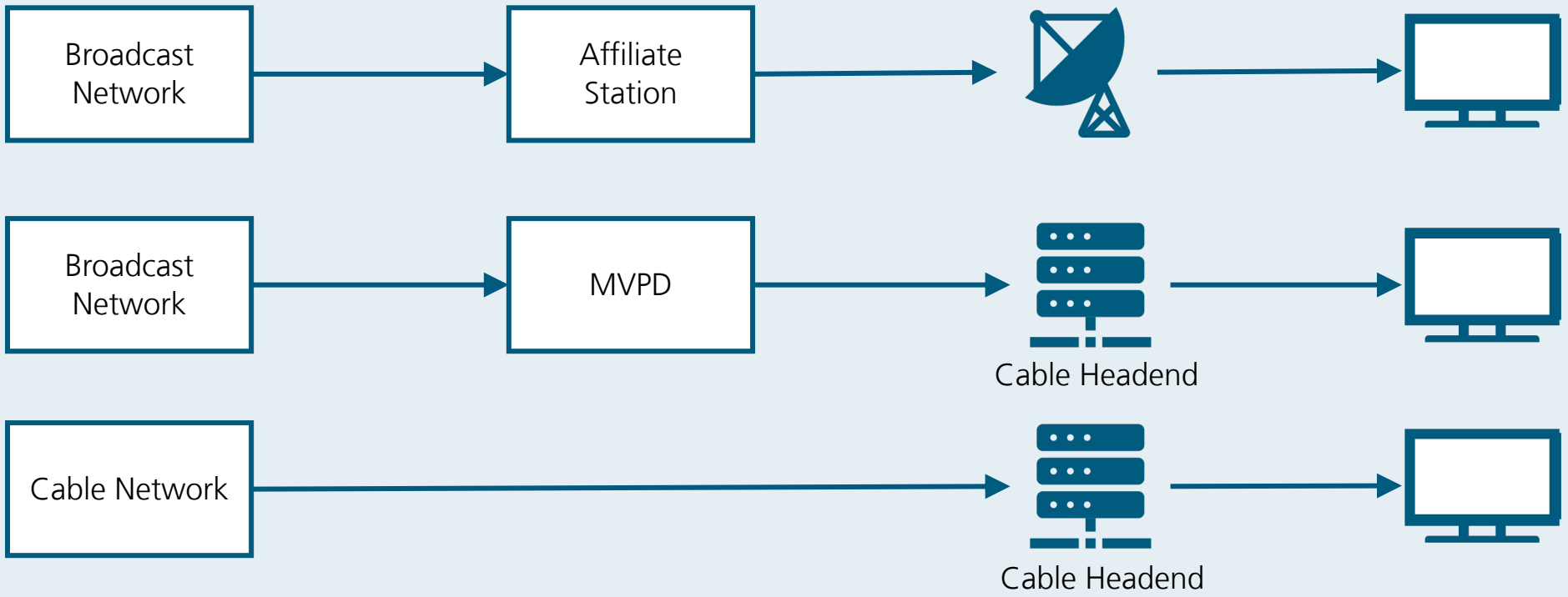
SCTE-35

- SCTE standards specification is called „Digital Program Insertion Cueing Message for Cable“
  - https://www.scte.org/standards-development/library/standards-catalog/scte-35-2019/
  - Covers In-Band Event Signaling for Live Video
- Standard for signaling events, splice points and content segment boundaries for
  - Advertising
  - Program and distribution control (e.g., blackouts)
  - EPG synchronization
  - Content Identification (ad breaks, advertising content, programming content like chapters or specific programs)
- related SCTE specifications
  - SCTE-67: Recommended Practice for Digital Program Insertion for Cable
  - SCTE-224 2021: Event Scheduling and Notification Interface (ESNI)

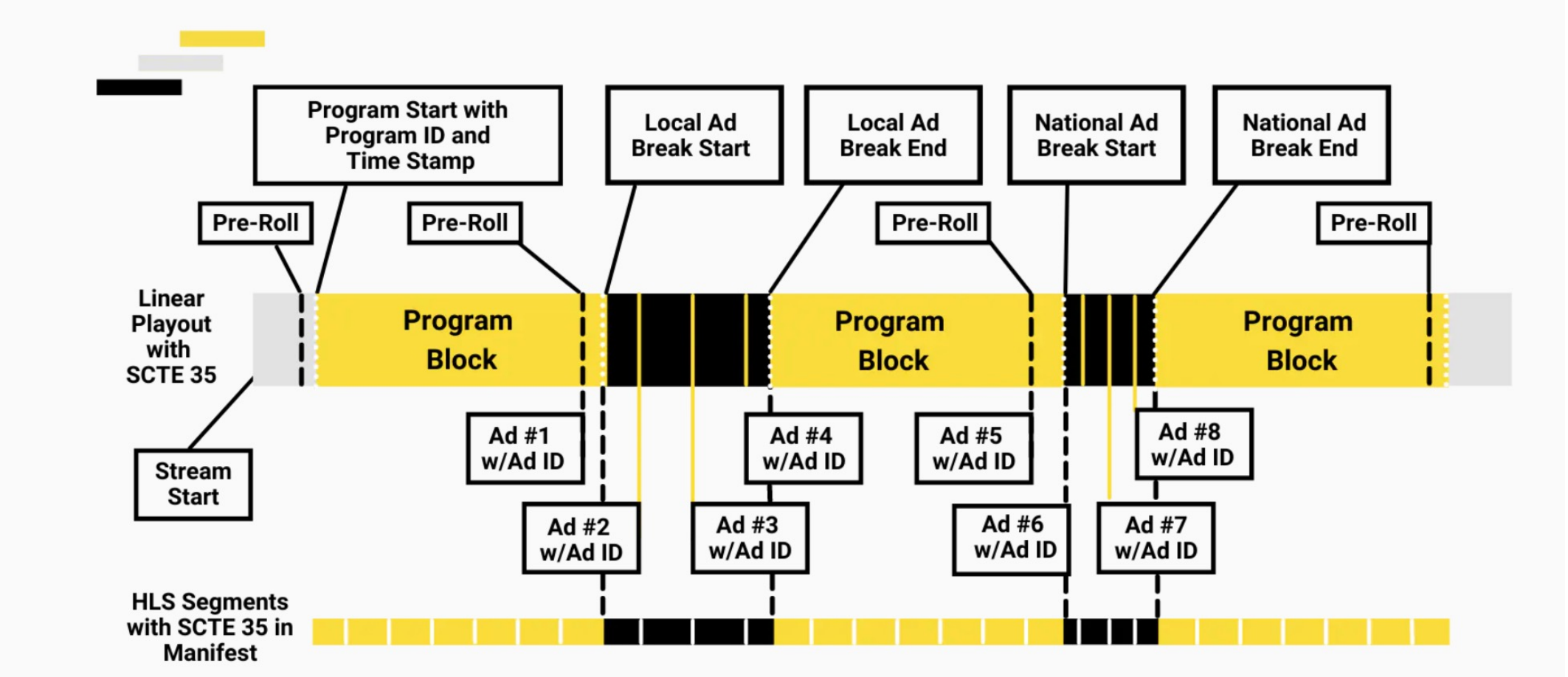
Ad Insertion – the original use case for cue messages

- Cue messaging for ad insertion is widely adopted in the U.S. since many years
- tiered distribution model in the broadcast and cable industry where:
  - Network broadcasters sell national ad breaks (ad time), and
  - MVPDs (Multichannel Video Programming Distributors) like Dish, Comcast, AT&T ore local stations sell local ad breaks
  - Based on the signaled placement opportunities (avails) a local station or MVPD splice in local ads
- 16 minutes of ads per hour consists of
  - 12 min national ads (provider)
  - 4 min local ads (distributor)
- distribution scheme is reflected in the cue messaging scheme
- OTT extends this model by new monetization options for CatchUp, VOD, DVR, etc.

TV distribution model in the US



SCTE-35 embedded programmed HLS stream

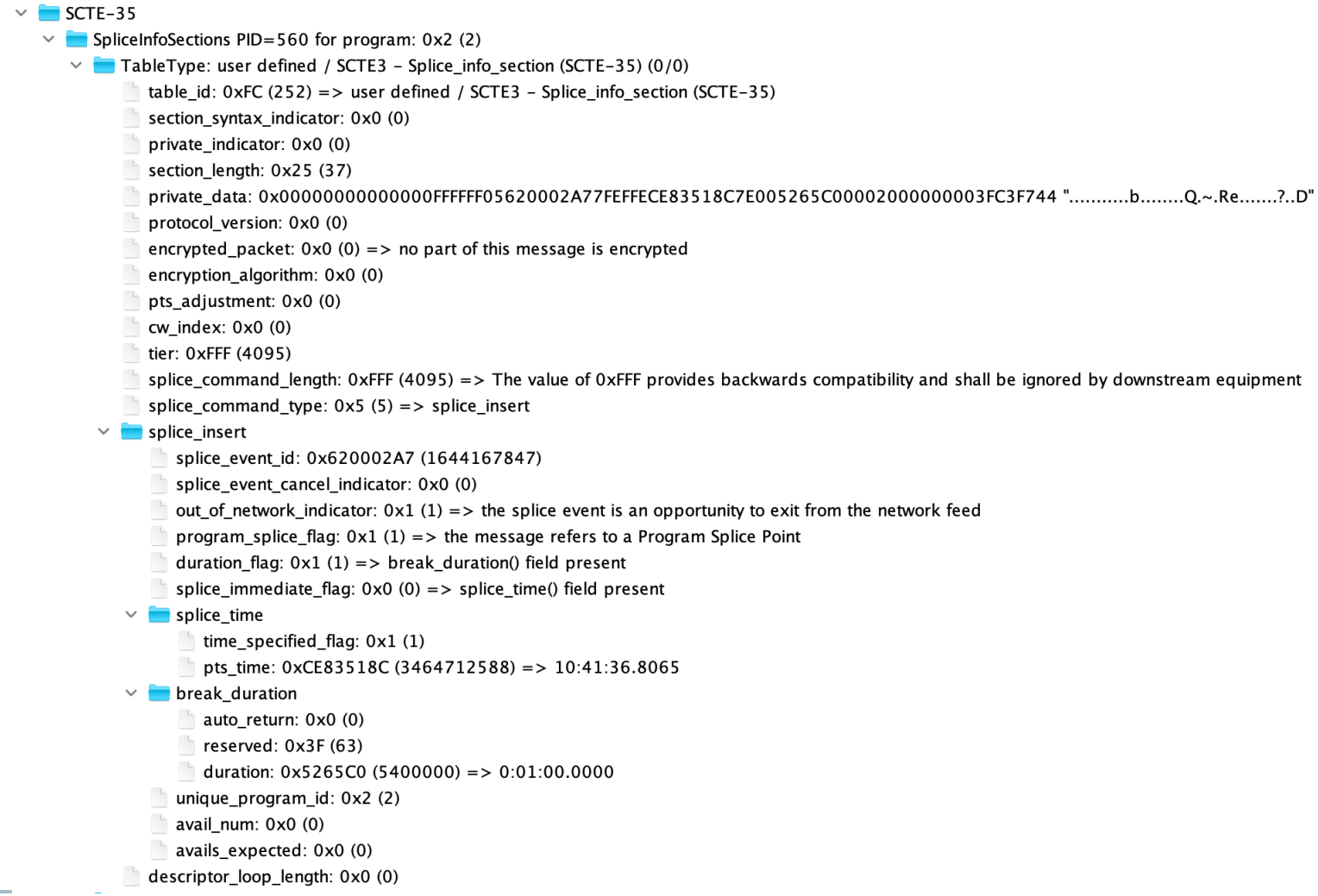


SCTE-35 Messages

- Binary
- Original message format
- Carried in MPEG-2 TS
  - dedicated PID (typically PID 500)
  - 30-50 bytes per message
- Not human readable
- Non-trivial to parse and decode
- Converted to Base64 or hex strings when signaled in DASH or HLS

- XML
  - XML schema added in 2013 to the standard
  - human readable
  - Uncommon, due to file size

SCTE in Transport Streams



SCTE-35 Binary Messages

```
#EXTM3U
...
#EXT-X-DATERANGE:ID="splice-6FFFFFF0",START-DATE="2014-03-05T11:
15:00Z",PLANNED-DURATION=59.993,SCTE35-OUT=0xFC002F0000000000FF0
00014056FFFFFF000E011622DCAFF000052636200000000000A0008029896F50
000008700000000

... Media Segment declarations for 60s worth of media

#EXT-X-DATERANGE:ID="splice-6FFFFFF0",DURATION=59.993,SCTE35-IN=
0xFC002A0000000000FF00000F056FFFFFF000401162802E6100000000000A00
08029896F50000008700000000
```

```
#EXT-OATCLS-SCTE35:/DA0AAAAAAAAAAABQb+ADAQ6QAeAhxDVUVJQAAAO3/PAAEUrEoICAAAAAAg+2UBNAAANvrtoQ==
#EXT-X-ASSET:CAID=0x0000000020FB6501
#EXT-X-CUE-OUT:30.000
.
#EXT-X-CUE-OUT-CONT:ElapsedTime=5.939,Duration=30.000,SCTE35=/DA0AAAA+…AAg+2UBNAAANvrtoQ==
.
#EXT-X-CUE-IN
```

https://www.middleman.tv/scte35-parser?base64=%252FDBlAAAAAAAAAP%252FwBQb%252FxIxlfwBPAiNDVUVJAAAAAn%252BDDBRNRFNORjMxMDEyODYwMjAwMDEwMREAAAIoQ1VFSQAAAAN%252FwwAAqhGADBRNRFNORjMxMjA1ODMwMTAwMjMwMRAAAFxbVI8%253D&format=json


```
{
  "table_id": 252,
  "section_syntax_indicator": false,
  "private_indicator": false,
  "section_length": 52,
  "protocol_version": 0,
  "encrypted_packet": false,
  "encryption_algorithm": 0,
  "pts_adjustment": 0,
  "cw_index": 0,
  "tier": "0x0",
  "splice_command_length": 5,
  "splice_command_type": 6,
  "splice_command": {
    "splice_time": {
      "time_specified_flag": true,
      "pts_time": {
        "pts_time": 3150057,
        "wall_clock_seconds_ext": 35.000633333333333,
        "wall_clock_time_ext": "00:00:35:00100"
      }
    }
  },
  "splice_descriptor_loop_length": 30,
  "splice_descriptors": [
    {
      "splice_descriptor_tag": 2,
      "descriptor_length": 28,
      "identifier": 1129661769,
      "private_bytes": "QAAA03/PAAEUrEoICAAAAAg+2UBNAAA",
      "segmentation_event_id": 1073741883,
      "segmentation_event_cancel_indicator": false,
      "program_segmentation_flag": true,
      "segmentation_duration_flag": true,
      "delivery_not_restricted_flag": false,
      "web_delivery_allowed_flag": false,
      "no_regional_blackout_flag": true,
      "archive_allowed_flag": true,
      "device_restrictions": 3,
      "component_count": 0,
      "elementary_pid_streams": null,
      "segmentation_duration": {
        "pts_time": 18132042,
        "wall_clock_seconds_ext": 201.46713333333332,
        "wall_clock_time_ext": "00:03:21:46700"
      },
      "segmentation_upid_type": 8,
      "segmentation_upid_name_ext": "TI: AiringID (former Turner ID), used to indicate a specific airin
g of a Program that is unique within a Network.",
      "segmentation_upid_length": 8,
      "segmentation_upid": "0x0000000020FB6501",
      "segmentation_type_id": 52,
      "segmentation_message_ext": "Provider Placement Opportunity Start",
      "segment_num": 0,
      "segments_expected": 0,
      "sub_segment_num": 0,
      "sub_segments_expected": 0
    }
  ],
  "alignment_stuffing": null,
  "E_CRC_32": null,
  "CRC_32": "0x36FAEDA1",
  "CRC_32_computed_ext": "0x36FAEDA1",
  "scte35_parser_ext": {
    "parse_status": "SCTE-35 cue parsing completed with 0 errors.",
    "error_messages": [],
    "table_id": 252,
    "splice_command_type": 6
  }
}
```

SCTE-35 Commands and Descriptors

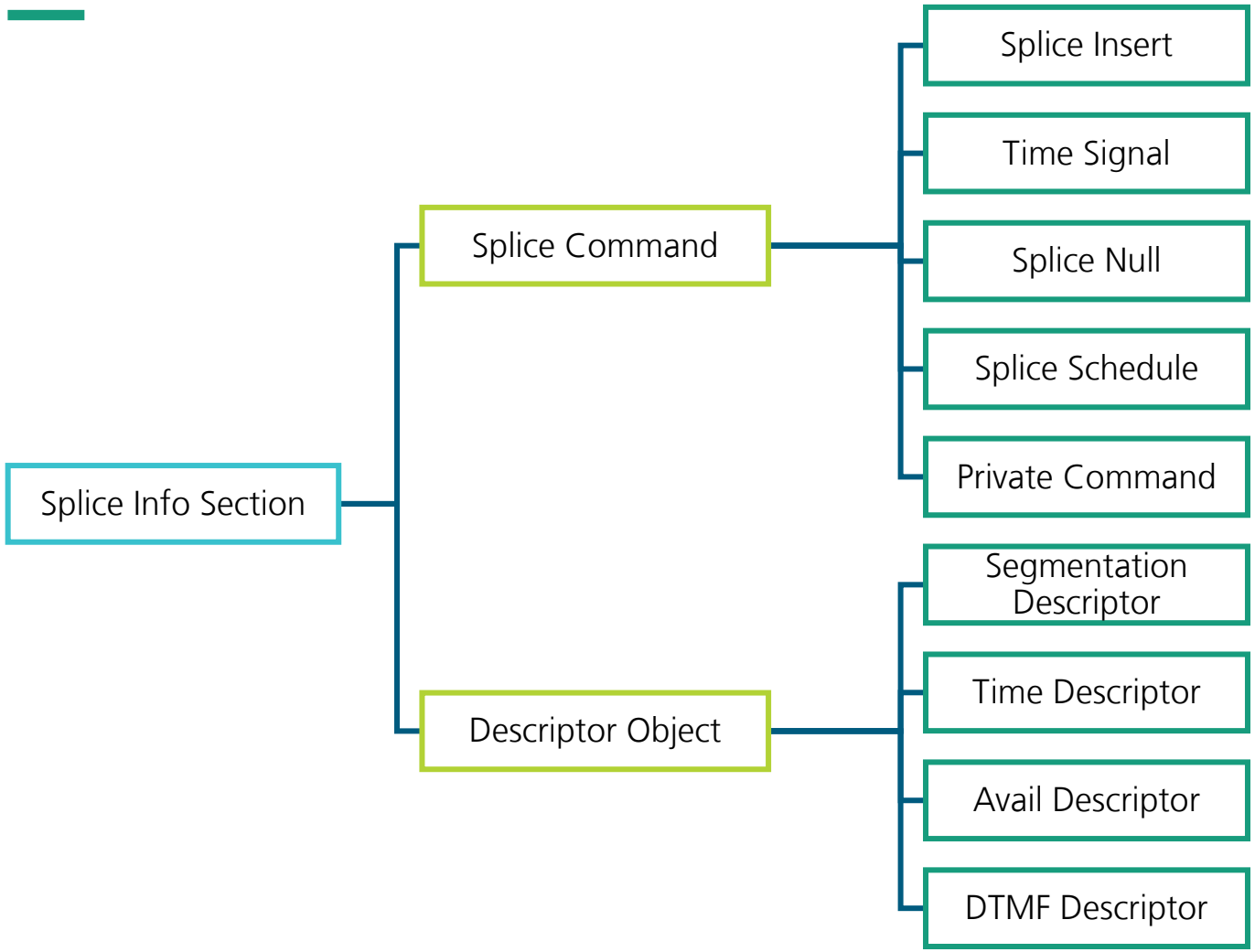


Table 5 - splice_info_section()

Syntax	Bits	Mnemonic	Encrypted
splice_info_section() {			
table_id	8	uimsbf	
section_syntax_indicator	1	bslbf	
private_indicator	1	bslbf	
sap_type	2	bslbf	
section_length	12	uimsbf	
protocol_version	8	uimsbf	
encrypted_packet	1	bslbf	
encryption_algorithm	6	uimsbf	
pts_adjustment	33	uimsbf	
cw_index	8	uimsbf	
tier	12	bslbf	
splice_command_length	12	uimsbf	
splice_command_type	8	uimsbf	E
if(splice_command_type == 0x00)			
splice_null()			E
if(splice_command_type == 0x04)			
splice_schedule()			E
if(splice_command_type == 0x05)			
splice_insert()			E
if(splice_command_type == 0x06)			
time_signal()			E
if(splice_command_type == 0x07)			
bandwidth_reservation()			E

ANSI/SCTE 35 2020

Syntax	Bits	Mnemonic	Encrypted
if(splice_command_type == 0xff)			
private_command()			E
descriptor_loop_length	16	uimsbf	E
for(i=0; i<N1; i++)			
splice_descriptor()			E
for(i=0; i<N2; i++)			
alignment_stuffing	8	bslbf	E
if(encrypted_packet)			
E_CRC_32	32	rpchof	E
CRC_32	32	rpchof	
}			

SCTE-35 Splice_insert

- Historically used to signal local ad placement opportunities
- Usually only includes splice timestamp and planned duration (optionally)
- Limited in syntax and use cases
- widely used in OTT
- derived simplified signaling styles without binary message

CUE-OUT-Style: elemental

```
#EXT-X-CUE-OUT:30.000
.
#EXT-X-CUE-OUT-CONT: 8.308/30
.
#EXT-X-CUE-OUT-CONT: 20.391/30
.
#EXT-X-CUE-IN
```

ANSI/SCTE 35 2020

Table 10 - splice_insert()

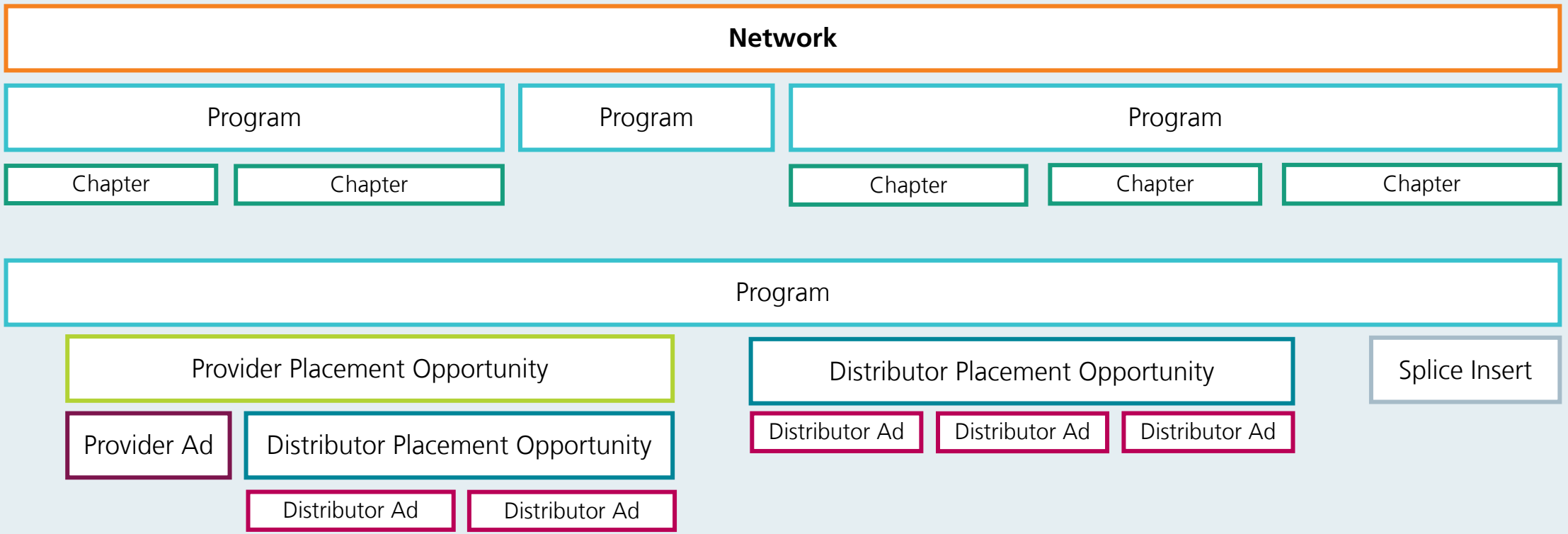
Syntax	Bits	Mnemonic
splice_insert() {		
splice_event_id	32	uimsbf
splice_event_cancel_indicator	1	bslbf
reserved	7	bslbf
if(splice_event_cancel_indicator == '0') {		
out_of_network_indicator	1	bslbf
program_splice_flag	1	bslbf
duration_flag	1	bslbf
splice_immediate_flag	1	bslbf
reserved	4	bslbf
if((program_splice_flag == '1') && (splice_immediate_flag == '0'))		
splice_time()		
if(program_splice_flag == '0') {		
component_count	8	uimsbf
for(i=0;i<component_count;i++) {		
component_tag	8	uimsbf
if(splice_immediate_flag == '0')		
splice_time()		
}		
}		
if(duration_flag == '1')		
break_duration()		
unique_program_id	16	uimsbf
avail_num	8	uimsbf
avails_expected	8	uimsbf
}		
}		

SCTE-35 Time_signal

- Generic command for signaling timestamp-synchronized data
- Useful only with segmentation_descriptor object
- More enhanced signaling schemes
- Applicable for all use cases that involve more than just local avail signaling
- Uses the hirarchical syntax for provider & distributor


```
{
  "table_id": 252,
  "section_syntax_indicator": false,
  "private_indicator": false,
  "section_length": 75,
  "protocol_version": 0,
  "encrypted_packet": false,
  "encryption_algorithm": 0,
  "pts_adjustment": 0,
  "cw_index": 255,
  "tier": "0xFFF",
  "splice_command_length": 5,
  "splice_command_type": 6,
  "splice_command": {
    "splice_time": {
      "time_specified_flag": true,
      "pts_time": {
        "pts_time": 0,
        "wall_clock_seconds_ext": 0.0,
        "wall_clock_time_ext": "00:00:00:00000"
      }
    }
  },
  "splice_descriptor_loop_length": 53,
  "splice_descriptors": [
    {
      "splice_descriptor_tag": 2,
      "descriptor_length": 51,
      "identifier": 1129661769,
      "private_bytes": "YgAFin//AABSZcAJH1NJR05BTDpEUjIxWjA3WlQ4YThhc25pdVVoZWlBPT00AAA=",
      "segmentation_event_id": 1644168586,
      "segmentation_event_cancel_indicator": false,
      "program_segmentation_flag": true,
      "segmentation_duration_flag": true,
      "delivery_not_restricted_flag": true,
      "web_delivery_allowed_flag": false,
      "no_regional_blackout_flag": false,
      "archive_allowed_flag": false,
      "device_restrictions": 0,
      "component_count": 0,
      "elementary_pid_streams": null,
      "segmentation_duration": {
        "pts_time": 5400000,
        "wall_clock_seconds_ext": 60.0,
        "wall_clock_time_ext": "00:01:00:00000"
      },
      "segmentation_upid_type": 9,
      "segmentation_upid_name_ext": "ADI: CableLabs metadata identifier with abbreviated syntax: [element]:[identifier]."
      "segmentation_upid_length": 31,
      "segmentation_upid": "SIGNAL:DR21Z07ZT8a8asniuUheiA==",
      "segmentation_type_id": 52,
      "segmentation_message_ext": "Provider Placement Opportunity Start",
      "segment_num": 0,
      "segments_expected": 0,
      "sub_segment_num": 0,
      "sub_segments_expected": 0
    }
  ],
  "alignment_stuffing": null,
  "E_CRC_32": null,
  "CRC_32": "0xF3DC6757",
  "CRC_32_computed_ext": "0xF3DC6757",
  "scte35_parser_ext": {
    "parse_status": "SCTE-35 cue parsing completed with 0 errors.",
    "error_messages": [],
    "table_id": 252,
    "splice_command_type": 6,
    "mdat": null
  }
}
```

Segment Hirarchy



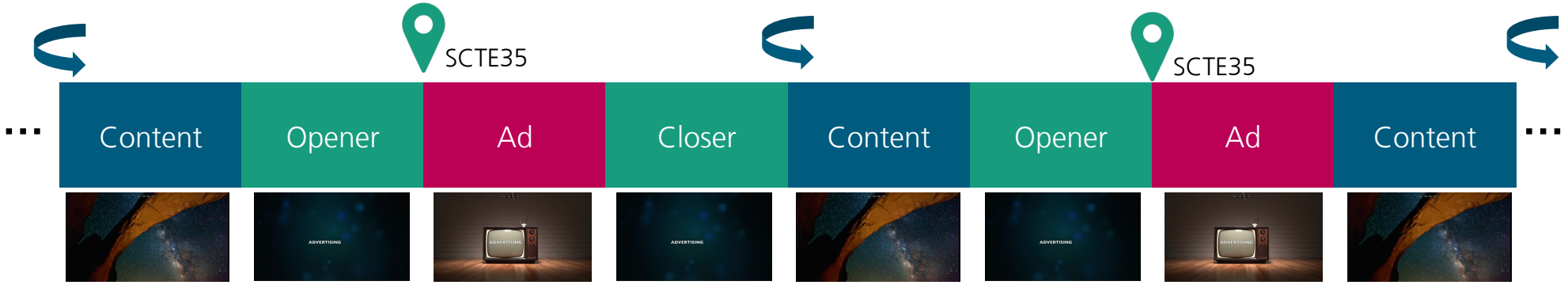
In practice the hirarchical structure is used in combination with time_signal carried in SCTE-35 binary messages

Hands On – SCTE 35 Analysis in Transport Streams (Encoder & Transcoder level)

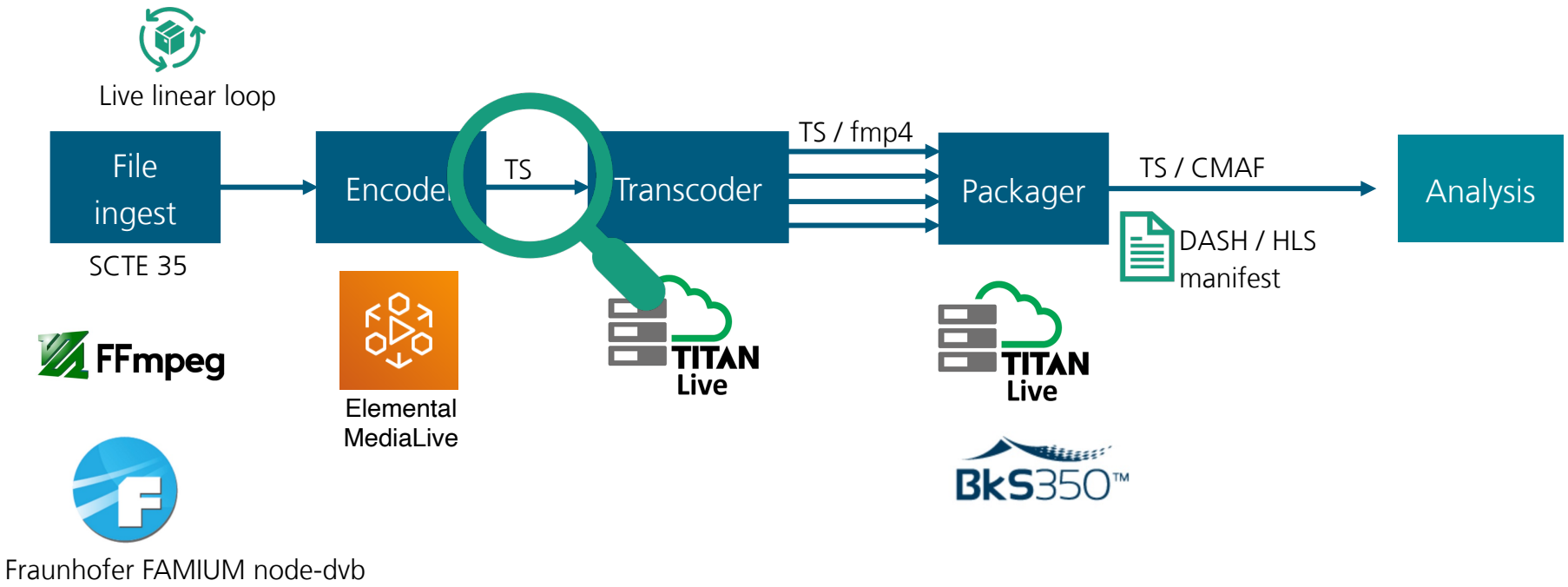
Hands on

Sample content:

- TS file composed of Content, Opener, Ad, Closer
- We inserted SCTE35 Splice_Insert with a Splice_Time matching the 1st Frame of the Ad + duration
- We loop this file to create a live linear stream on the encoder
- Subsequent steps are transcoding and packaging

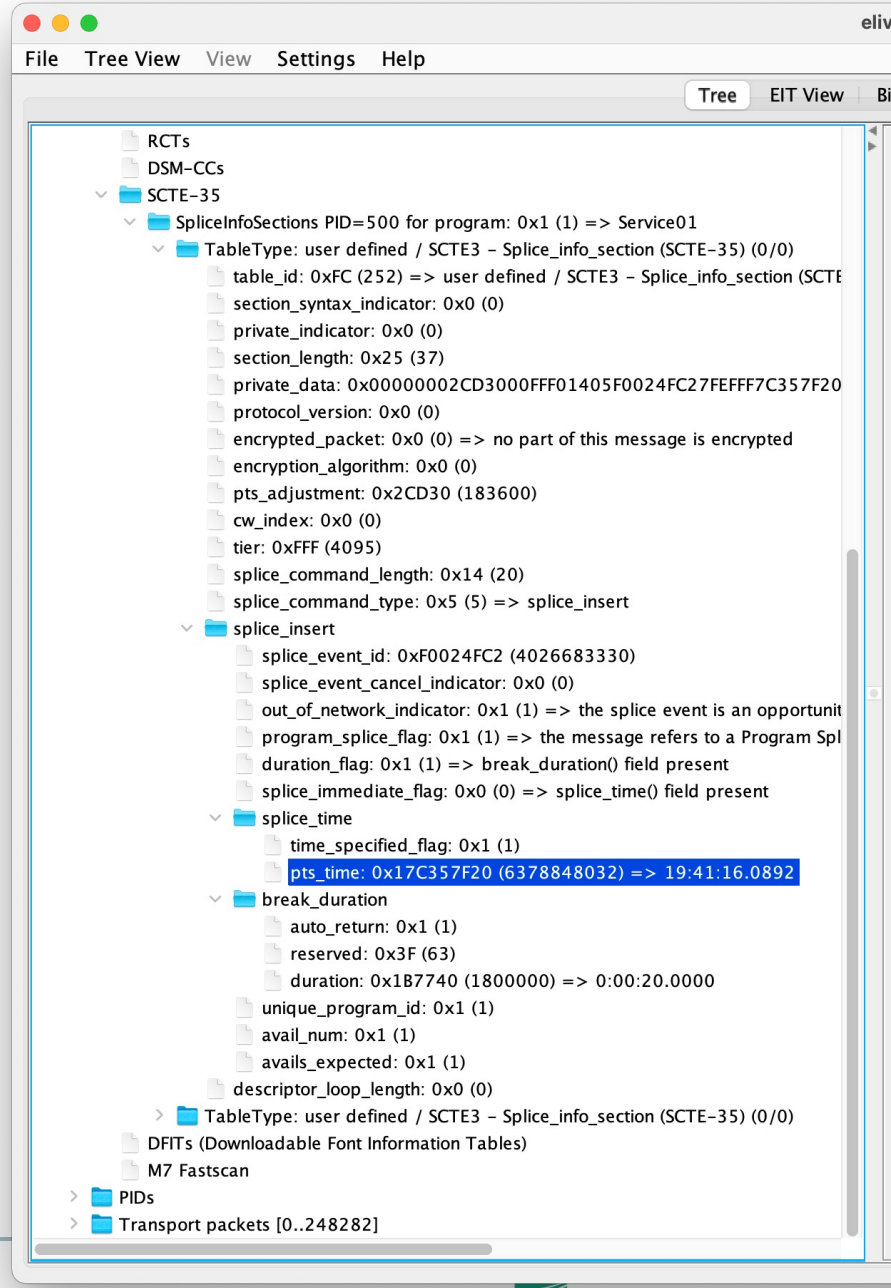


Hands On


Live linear loop
File ingest
SCTE 35
Encoder
TS
Transcoder
TS / fmp4
Packager
TS / CMAF
Analysis
DASH / HLS manifest
FFmpeg
Elemental MediaLive
TITAN Live
TITAN Live
BkS350™
Fraunhofer FAMIUM node-dvb

Encoder Output

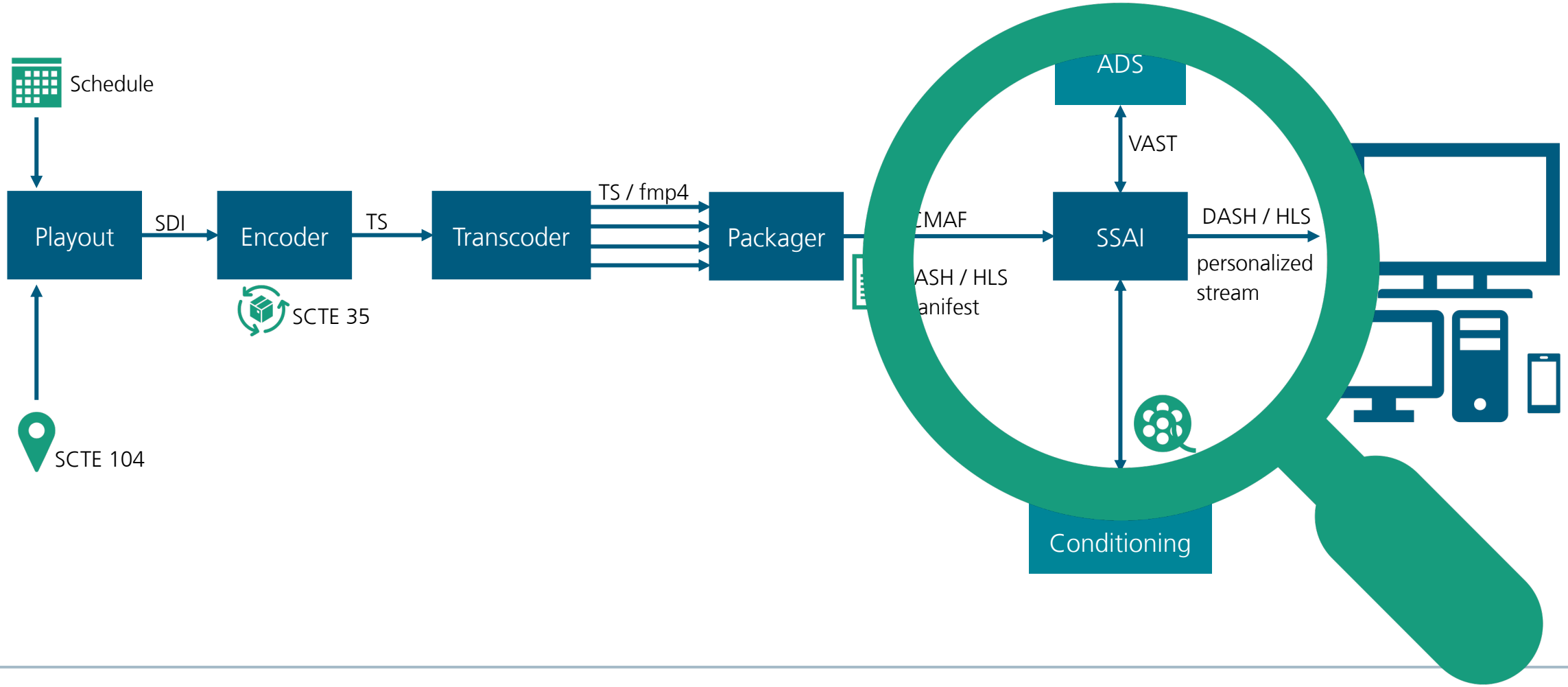
- SCTE-35 Splice Insert, Splice Time
  PTS+ADJUST (6378848032 + 183600)/90000 → 70878,129
- ffplay -vf "drawtext=text='%{pts} %{pict_type}':fontsize=(h/10):box=1:x=(w-tw)/2:y=h-(2*lh)" elive-output.ts



Fraunhofer FOKUS

Dynamic Ad Insertion - recommendations to allow a seamless SSAI experience

Content and signaling lifecycle – live linear



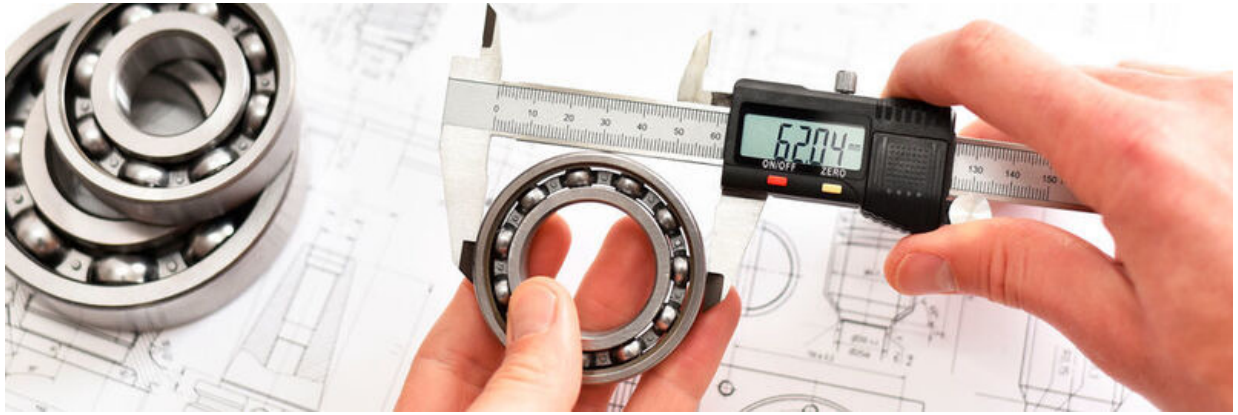
Fraunhofer FOKUS

SSAI Ingest

- **Continuously check and validate your sources**

- Be aware that manifest manipulation based SSAI workflows require to have an appropriate encoded and conditioned source to work properly

  - Check correct timing and signaling of SCTE35 CUE marker
  - Check segment boundaries and correct encoding / IDR-Frame decoration



SCTE-35 signaling in HLS

- Multiple different approaches to signal messages via playlists
- #EXT-X-CUE-OUT
  - No binary messages carried
  - Translates to a simple splice_insert
- #EXT-OATCLS-SCTE35
  - Carries binary message as Base64 string
- #EXT-X-SCTE35
  - Preferred by SCTE
  - Carries binary message as Base64 string
- #EXT-X-DATERANGE
  - Preferred by Apple & Native AVPlayer API support
  - Carries binary message as HEX string

```
#EXT-X-CUE-OUT:30.000
.
#EXT-X-CUE-OUT-CONT: 8.308/30
.
#EXT-X-CUE-OUT-CONT: 20.391/30
.
#EXT-X-CUE-IN
```

```
#EXT-OATCLS-SCTE35:/DA0AAAAAAAAAAABQb+ADAQ6QAeAhxDVUVJQAAAO3/
PAAEUrEoICAAAAAg+2UBNAAANvrtoQ==
#EXT-X-ASSET:CAID=0x0000000020FB6501
#EXT-X-CUE-OUT:30.000
.
#EXT-X-CUE-OUT-CONT:ElapsedTime=5.939,Duration=30.000,SCTE35=/DA0AAAA+…AAg+2UBNAAANvrtoQ==
.
#EXT-X-CUE-IN
```

```
#EXT-SCTE35:CUE-OUT=YES,ID="22",DURATION=22.000,CUE="/DAlAAETZlHBAP/
wFAUAAAAWf+//UmtLsH4AHjZgAAAAAAAAsf7cfA=="
.
#EXT-SCTE35:CUE-IN=YES,ID="22",DURATION=22.000,CUE="/DAlAAETZlHBAP/
wFAUAAAAWf+//UmtLsH4AHjZgAAAAAAAAsf7cfA=="
```

```
#EXT-X-DATERANGE:ID="20",START-DATE="2020-06-03T14:56:00Z",PLANNED-DURATION=19,SCTE35-OUT=0x
FC302000000000000000FFF00F05000000147FFFFE001A17B0C0000000000061DFD67D
#EXT-X-CUE-OUT:19
#EXT-X-PROGRAM-DATE-TIME:2020-06-03T14:56:00Z
.
#EXT-X-CUE-IN
```

SCTE-35 signaling in DASH

- 2 approaches to signal SCTE35 message in DASH
  - Described in SCTE67 „Recommended Practice for Digital Program Insertion for Cable“

  - Associating DASH Periods with SCTE 35
    - align periods with ad break cue points (SCTE-35 segments)
    - entire DASH period is treated as ad break and replaced

  - Use of SCTE 35 with DASH Events
    - Signal events in DASH MPD
      (or media segments as Event Message Box „emsg“)

```
<EventStream schemeIdUri="urn:scte:scte35:2013a:bin" value="scte35_track_001_000"
timescale="50000">
      <Event presentationTime="77934885650300" duration="1500000" id="31218">/DAlAAAAAAAAAP/
      wFAUAAHnyf+/+Ih1YrP4AKTLgAAAAAAAAXos7AQ==</Event>
</EventStream>
```

- MPD constraints for SCTE use cases are standardized in SCTE-214 Part1 „ANSI/SCTE 214-4 2018 - MPEG DASH for IP-Based Cable Services Part 4: SCTE Common Intermediate

Audio / Video Alignment

Goal:

- Package DASH / HLS in a way, that audio and video segments have the exact same length -> "they are aligned"

Background:

- Audio is encoded in audio frames that calculate the frame duration as Samples_per_Frame/Sampling_Frequency.
- The Samples_per_Frame value depends on the audio codec in use.
  - As an example, AAC LC uses 1024 samples per frame, while AAC HE uses 2048 samples per frame.
- The video frame duration is only based on the video frame rate and is calculated as 1/video_frame_rate.
- A segment can only contain an integer value of frames, meaning frames can't be split across two segments.
- Therefore, if audio is to be in a separate stream from the video, then trying to find a segment duration that is equal in duration for both the audio and video segments can be tricky.

Audio and Video Alignment

Alignment of audio and video segments

- Integer framerates
- Smooth switch between content & ads
- VOD: round up to next segment boundary
- LIVE: Encoder must create IDR frame at ad marker position (close GOP)
  - Use segmentTimeline in DASH
- Helps to prevent timeline gaps
- Keeps DASH manifest small

24fps, AAC 48kHz		
Segment duration (seconds)	Video frames	Audio samples
8.00	192	375

25fps, AAC 48kHz		
Segment duration (seconds)	Video frames	Audio samples
1.92	48	90
3.84	96	180
6.40	160	300

30fps, AAC 48kHz		
Segment duration (seconds)	Video frames	Audio samples
1.60	48	75
4.80	144	225
6.40	192	300

https://anton.lindstrom.io/gop-size-calculator/

Ad Conditioning

- Ad Conditioning should match the content source
  - renditions / variants → Bitrate & Resolution & Codecs
  - I-Frame only playlists
  - Will prevent reinitialization of the media source in MSE players
  - Adjustment of audio level > normalization
- Prepare Ad Slates to handle gaps in signaled ad break duration and available ads
- Prefetch ads if your workflows allows for it

SCTE-224

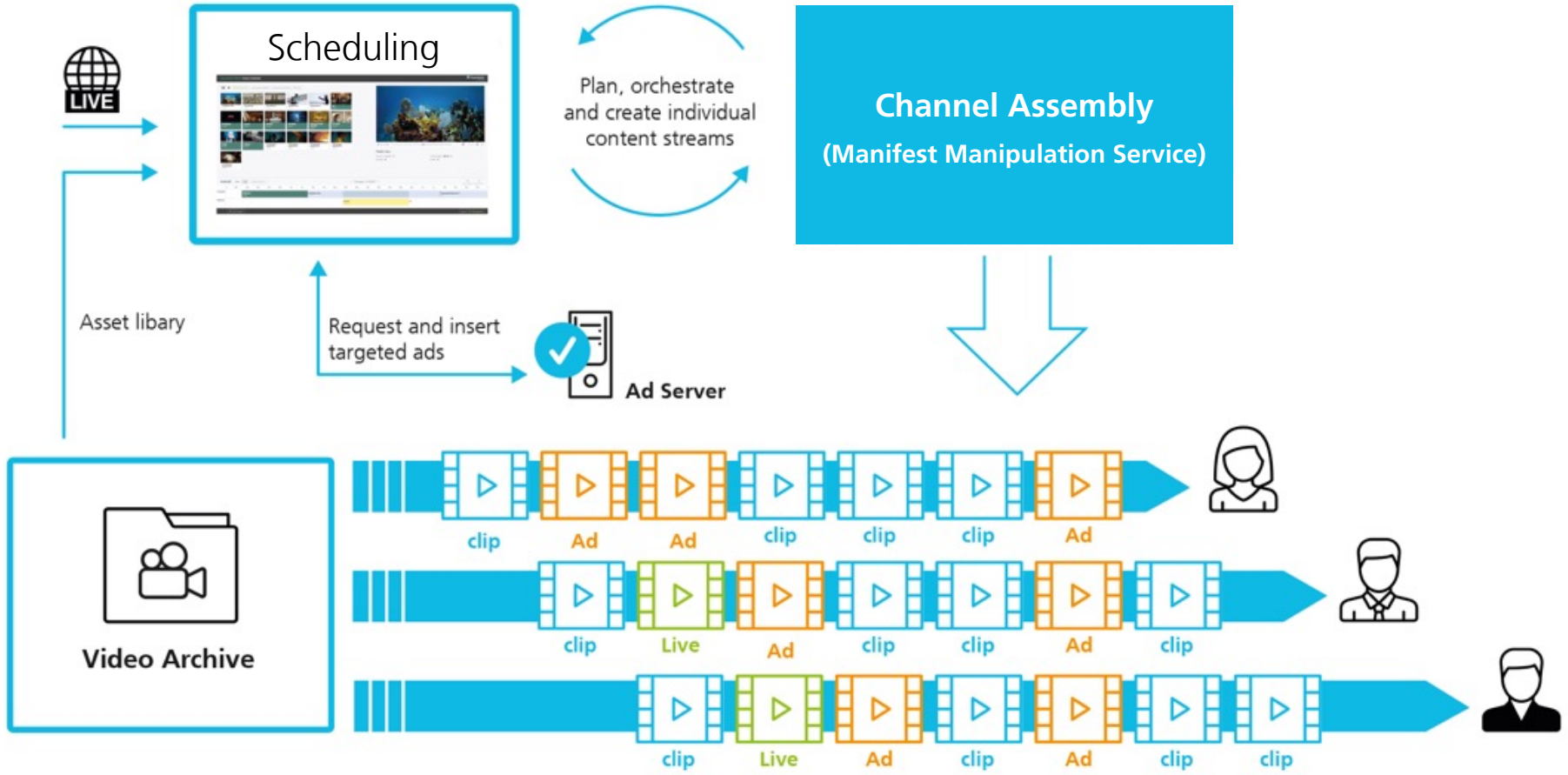
Advertisement in Broadcast and Broadband Environments

Fraunhofer FOKUS

VoD-to-Live Channel Origination Workflow (Channel Assembly)

Creating individualized and ad-enabled linear streams through manifest stitching

→ **Technical foundation for FAST – "free ad-supported streaming TV"**



SCTE-224 Basics

Defines the Event Scheduling and Notification Interface (ESNI) → out of band signaling (web interface)

- includes linear schedule timing, information and metadata (e.g. blackouts, restrictions, capture rights, Startover etc.)
- enables machine to machine control of playout to viewers

Audience

- Definition of user groups
- Described by a set of attributes

Schedule

- Set of program elements to provide upcoming schedule or signaled events
- Identified by unique program ID
- „Media Point“ defines a point in time thats associated to a specific action

Policies

- Add multiple viewing policies to an audience

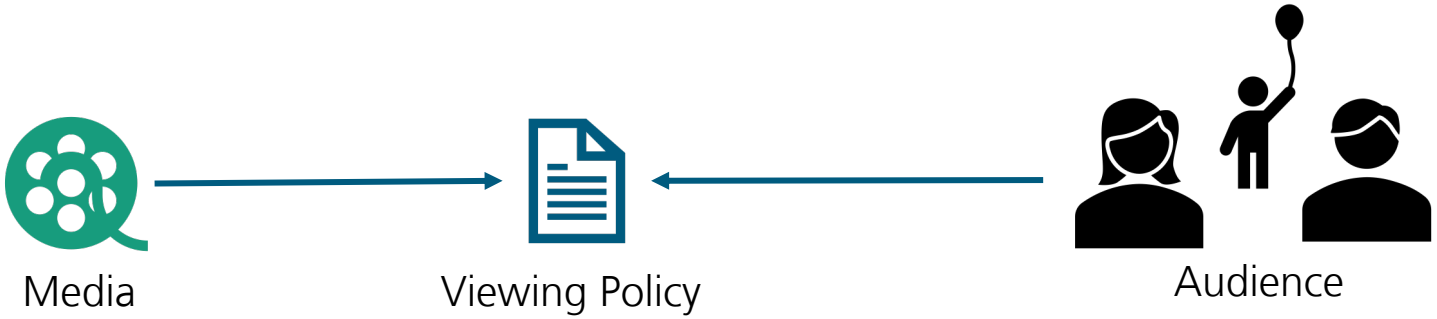
SCTE-224

- SCTE-224 is a control protocol that facilitates the transmission of event and policy information
- Communicates upcoming schedule or signal-based events and corresponding policy
- Allows a programmatic interface to replace existing content distribution controls traditionally performed via manual control in IRD's by providers.
- SCTE-224 defines rich metadata that transmits accurate, multi-level, schedule information which helps to schedule complex ad breaks providing detailed, frame-accurate, local ad insertion.
- SCTE-224 is the glue that makes SCTE104/35 work for more sophisticated applications as it carries the out-of-band metadata between programmers and local stations / broadcasters

SCTE-224 – How does it work?

SCTE-224 supports 5 basic message types:

- Media – a container for all events in the schedule (the channel)
- Media Point – specific „points“ within the schedule allowing for a dedicated action
- Policy – object to manage viewing policies
- ViewingPolicy – connects events and audiences
- Audience – defines a group of viewers



SCTE-224 – Message types

Media Point

- Id
- Description
- Source
- Alternate Ids
- Policies (apply / remove)
- Expires
- Effectiveness
- Metadata
  - StartTime
  - EndTime
- Match time
  - Offset
- Match signal
  - Segmentation type
- Episode ID
- Episode description
- Startover
- …

Audience

- Country (ISO3166)
- State
- Postal Code
- DMA
- vIRD
- Zip
- Authenticated
- Network
- OS
- DRM property
- Player feature
- Fast forward
- Programmer branding
- Default
  - Web blackout
  - Placement opportunity
  - …
- Lat Long
- Device Features
  - Dedicated display
  - Local Storage
  - Mobile
  - 3D
  - …
- …

- Exclusion
  - Match = NONE
    - Audience: Zip 80202, 080203, 80204
- Inclusion
  - Match = ANYX
    - Audience: Zip 80202, 080203, 80204

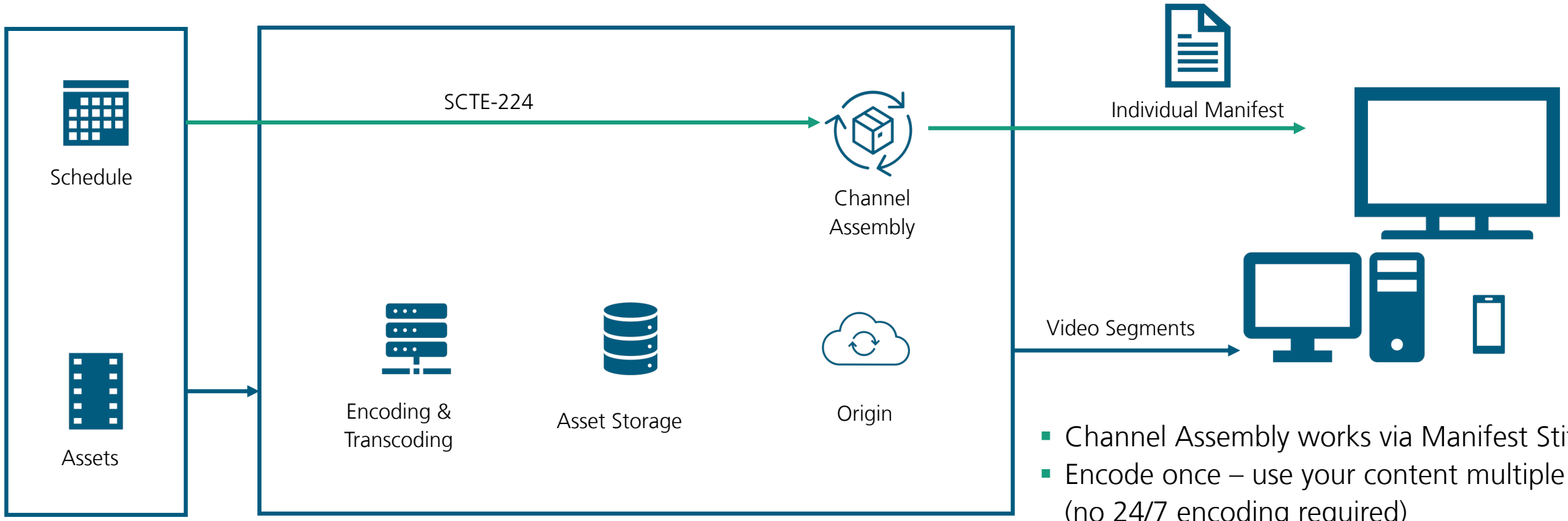
Action

- DRM
- Max resolution
- Resume
- Rewind
- HDMI blocked
- Download blocked
- Mirror blocked
- Capture
- Preroll slate
- Preroll DAI
- Midroll DAI
- Postroll DAI
- Subscriber view limit
- Max concurrent clients
- …

SCTE-224 – Use Cases

- Blackout (Zip / device / Region)
- Resident Media Points (e.g. default ad for a show if something goes wrong)
- Alternate Content (Switch to alternate channel/ feed or VOD asset playout)
- Geographical enforcement (regional content, travel rights, country boundary enforcement)
- EPG (restrictions, regionalization, long-running game)
- Advertising
  - local vs national
  - Ad descriptions
  - Ad inclusion / exclusion
  - C3 / C7 restrictions
  - Targeted ads
- Live to VOD / Vod to Live (FAST) / Playout Controls

SCTE-224 based Channel Assembly



- Channel Assembly works via Manifest Stitching
- Encode once – use your content multiple times (no 24/7 encoding required)

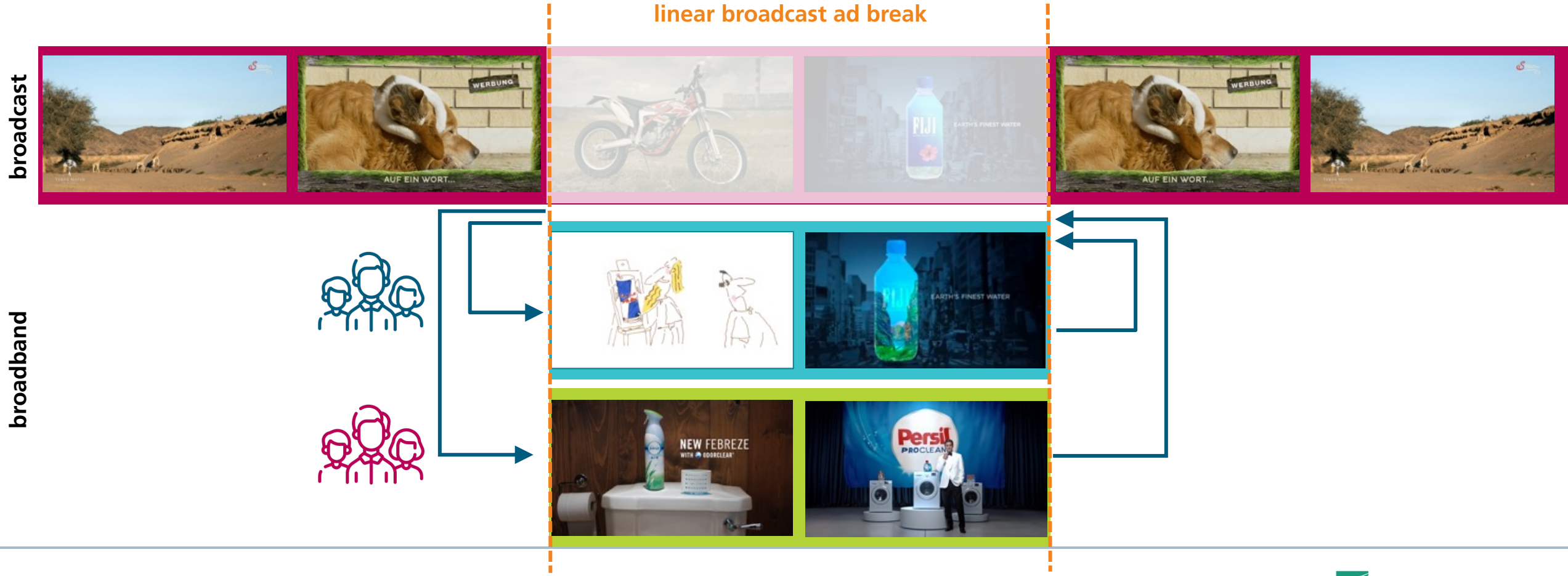
SSAI for Dynamic Ad Substitution in HbbTV 2.0.x

Advertisement in Broadcast and Broadband Environments

Fraunhofer FOKUS

Dynamic Ad Substitution in Hbbtv 2.0

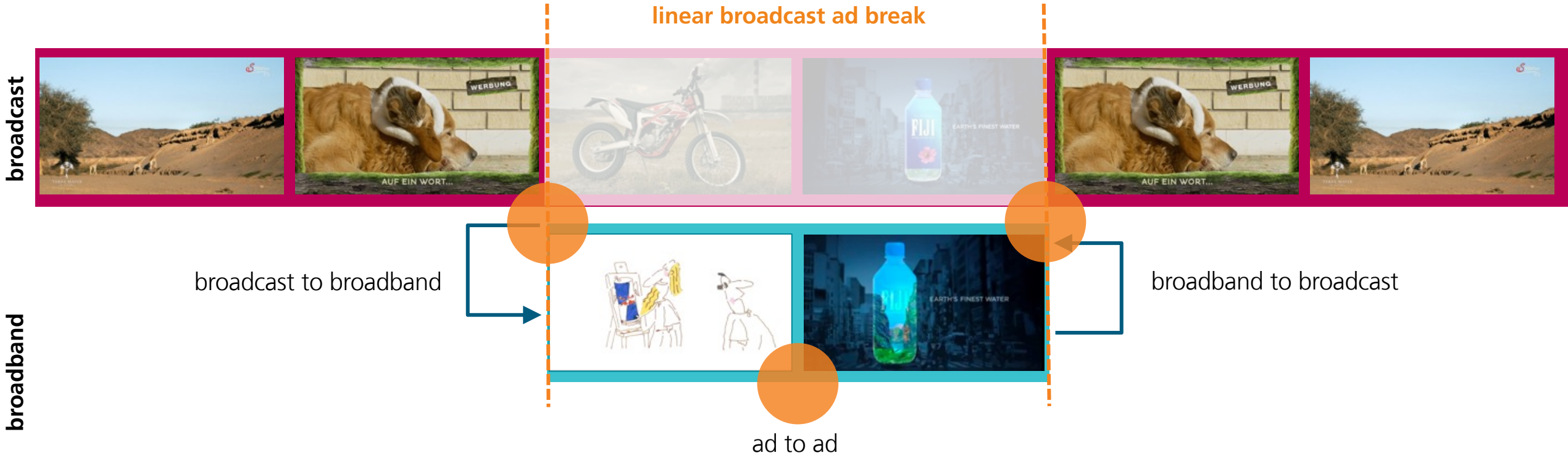
spot replacement



Fraunhofer FOKUS

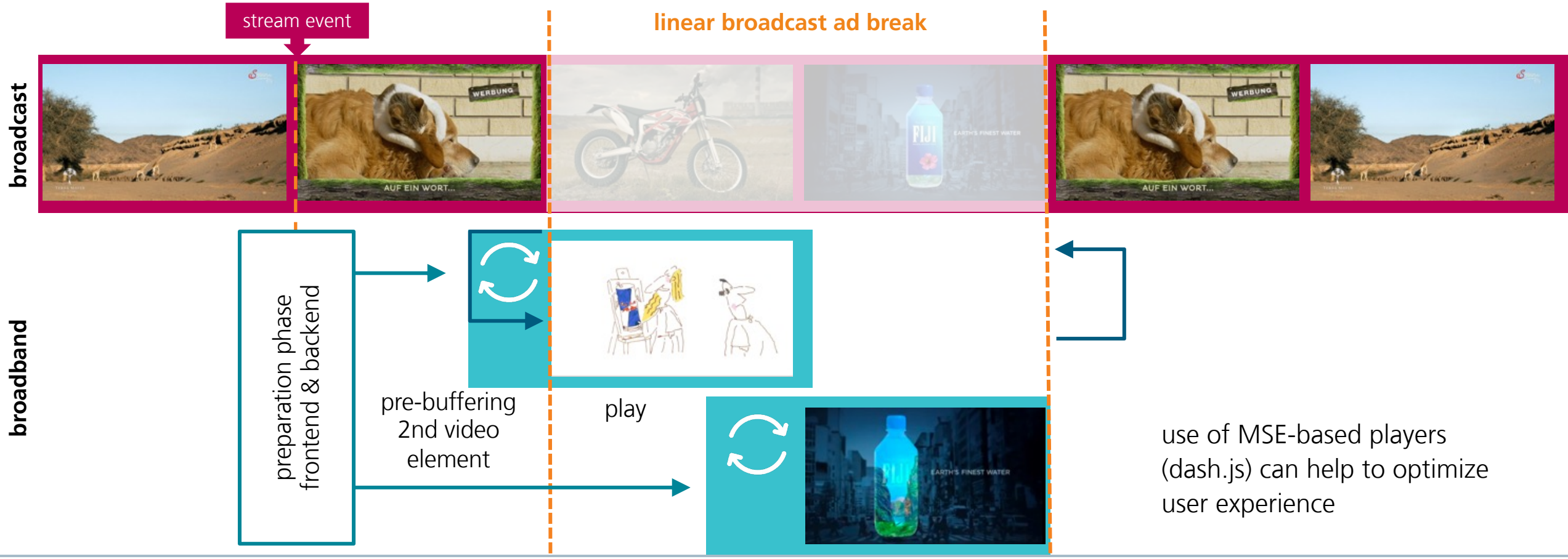
Dynamic Ad Substitution in Hbbtv 2.0

areas that need attention in spot replacement



Dynamic Ad Substitution in Hbbtv 2.0

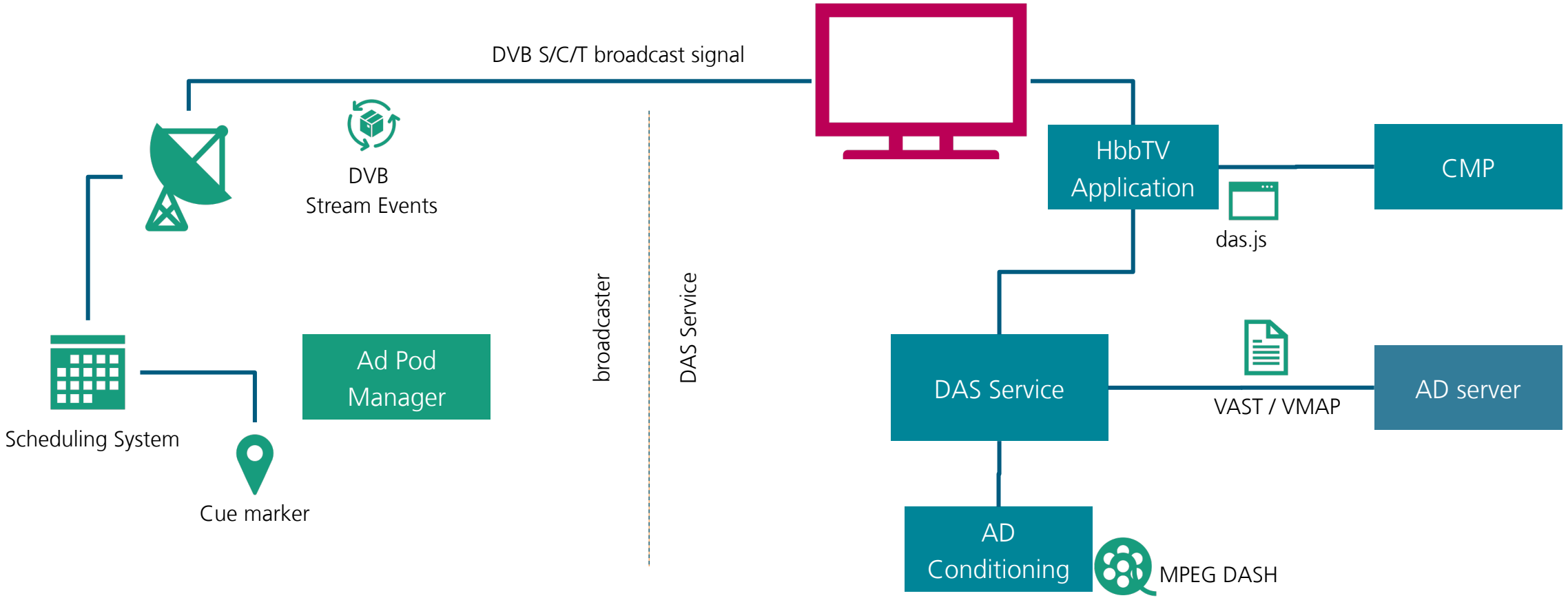
HbbTV 2.0 Solution



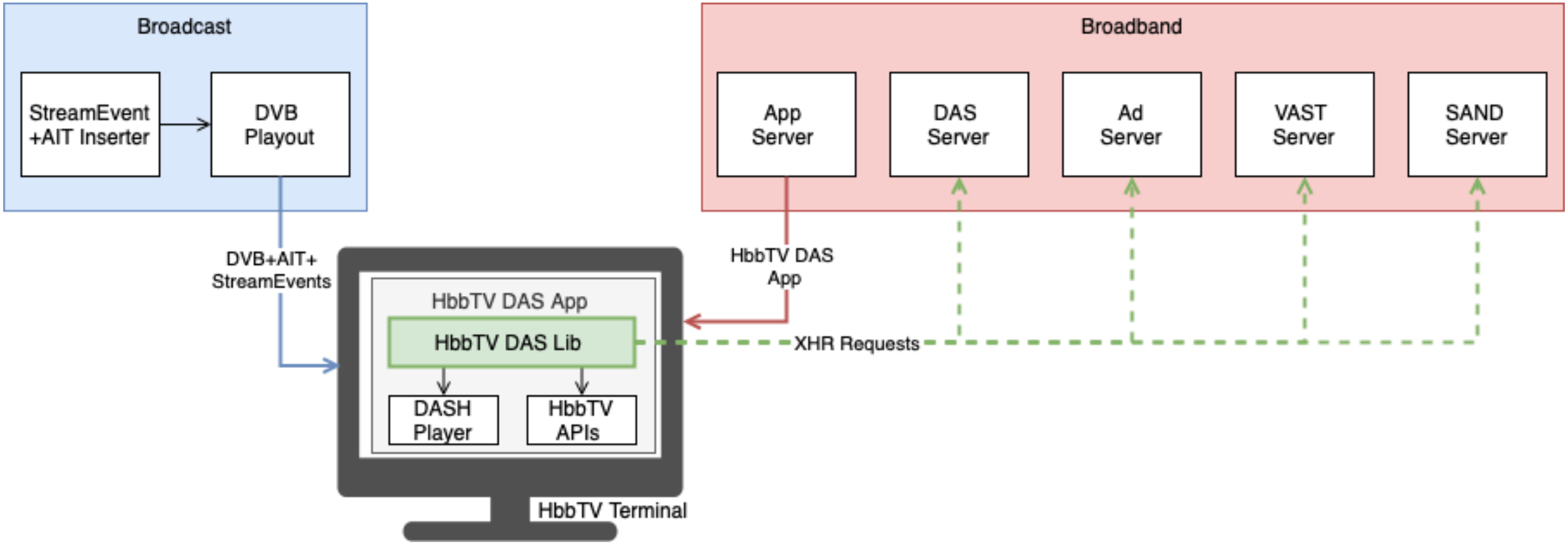
use of MSE-based players (dash.js) can help to optimize user experience

Fraunhofer FOKUS

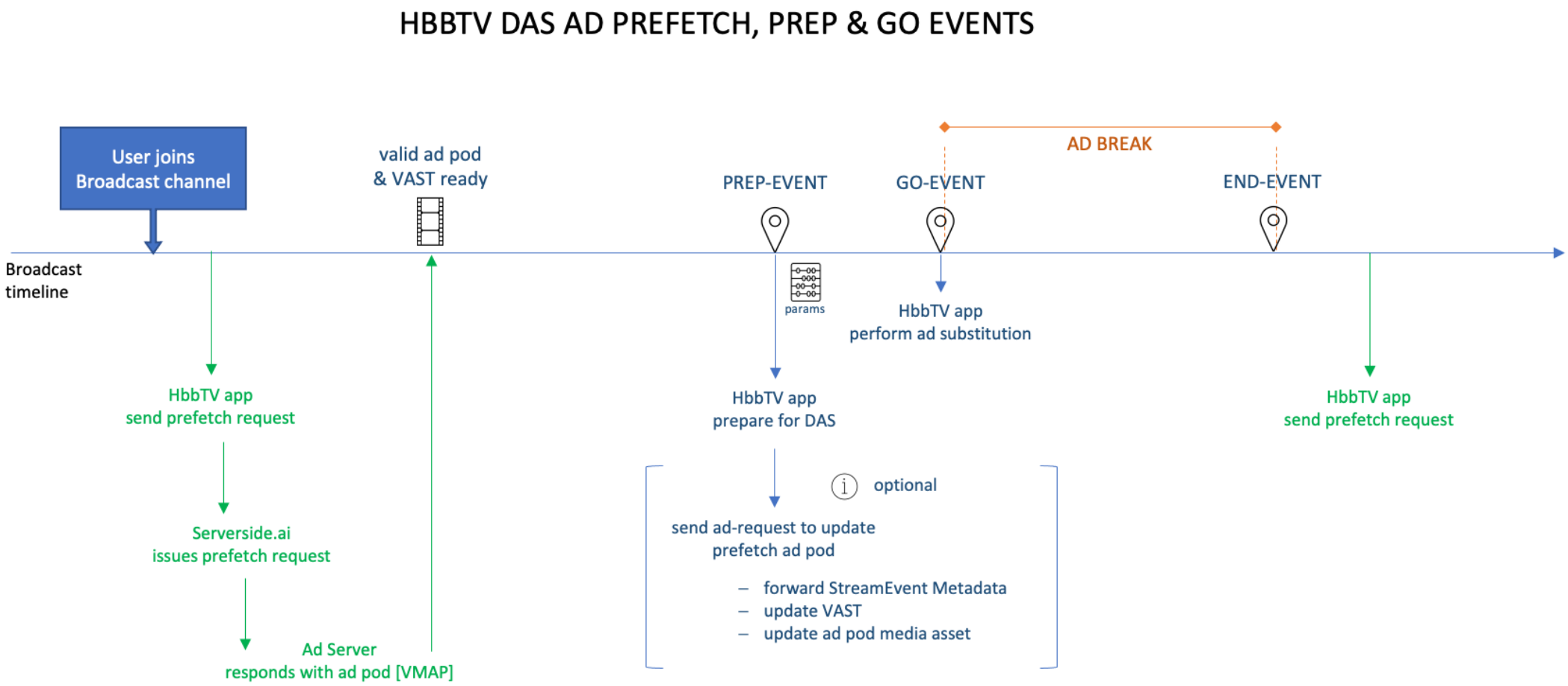
Dynamic Ad Substitution System Overview



Dynamic Ad Substitution



HbbTV DAS break signaling using DVB stream events



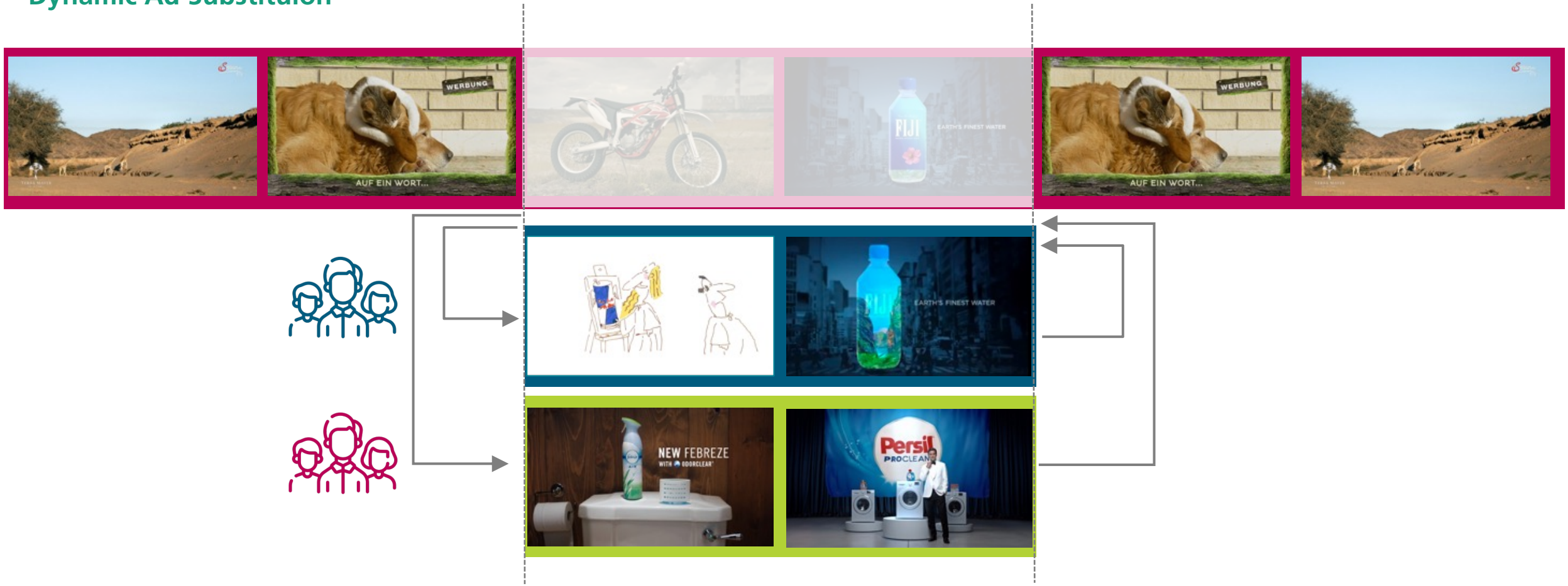
HbbTV-TA

Advertisement in Broadcast and Broadband Environments

Fraunhofer FOKUS

HbbTV-TA

Dynamic Ad Substituion



Fraunhofer FOKUS

HbbTV-TA

- Specification is about ad substitution via broadband
- Ad substitution opportunities will be announced by existing HbbTV technologies
  - DSM-CC stream events perhaps in combination with the TEMI timeline for better accuracy
  - SCTE 35/104 messages can be translated to stream events
- Business logic will re-use web advertising solutions
  - A mixture of JavaScript code in the HbbTV app and servers in the cloud
  - Ads will be delivered either via existing DASH / 'progressive download' or (NEW) Media Source Extensions in HbbTV
- New 'fast media switch' API' defined to enable optimized switching from broadcast to broadband & back
- 2 performance profiles defined for switching
  - Profile 1 enables replacing ads at the end of an ad break or entire break
    - implementations where broadcast and ads re-use the same video and audio decoder
  - Profile 2 also enables replacing ads in the middle of an ad break
    - implementations that use different video & audio decoders for ads & for broadcast and can keep decoding the broadcast while playing the ads

Announce a placement opportunity

- Existing HbbTV mechanisms can be used
  - DSM-CC stream events from HbbTV 1.5
  - Access to broadcast timelines from HbbTV 2

    PTS – simple but may be modified in broadcast distribution network

    TEMI – work needed for broadcasters to add this but more likely to pass through broadcast distribution network intact
- Stream events on their own not frame accurate
- Announcements far enough in advance can also be sent over the web
- Direct reception of industry standard SCTE-35 messages not yet included – In theory SCTE-35 message payload can be re-packaged into stream events

Thanks for your attention!

Robert Seeliger
DAI Lead & Senior Project Manager
Future Applications and Media
robert.seeliger@fokus.fraunhofer.de
Fraunhofer FOKUS
Berlin, Germany