# ADTV
# (Advanced Digitial TV)
# Draft V1.3

Dan Lovy
Keith Lehman
Steve Lee
Brian Dolan

Change Log

| Date | Action | Author |
|---|---|---|
| 8/12/15 | Creation  Draft v 1.0 | Dan Lovy |
| 8/20/15 | Update – small fixes, internal release | Dan Lovy |
| 9/8/15 | Draft 1.2 Edits – Response to discussions | Dan Lovy, Keith Lehman, Steve Lee, Brian Dolan |
| 7/10/16 | Draft 1.3 Addition of data components and name change | Dan Lovy |
| | | |
| | | |
| | | |

# TABLE OF CONTENTS

# 1.0 Introduction

Television audiences increasingly engage content on multiple screens and incorporate social media into their viewing experience, and "appointment television" has given way to cord-cutting and on-demand viewing. These seismic shifts are forcing content producers and advertisers to redefine storytelling and audience engagement. Cutting-edge television producers now strive to tell multi-season stories with direct tie-ins to games, apps, and social media. The lines are blurring between episodes, webisodes, games, ads, websites, and apps.

The media distribution world is shifting to a digital, one-to-one distribution model for content. This is creating opportunities to exploit the ability to create unique and personal content and add two way, and multi-way communication between viewers and content.

The problem is, there is no standard, defined way to create any of it; it's all being done one-off, poorly, or not at all. It is urgent that the industry define ways in which content producers can create these experiences and build a compatible set of supporting tools, technology, and infrastructure.

ADTV (Digital Communication and Media Protocol) is a set of signals and mechanisms designed to standardize the delivery of this new broadcast model. As this represents a convergence of two worlds, the interactive, internet delivered experiences and the large scale media distribution infrastructure, ADTV borrows from both categories of standards bodies. There are elements found in IAB (Internet Advertising Bureau) and the SCTE (Society of Cable Television Engineers).

ADTV is the work of SIMC. SIMC is the Social and Interactive Media Consortium. SIMC is an interdisciplinary group of companies that represent a range of stake holders. Members include advertisers, content producers, content publishers, infrastructure providers and tool companies.

The protocols and component communications are designed to be freely available and open source. There will be a validation suite as well as a repository for ADTV Logic Modules (described later).

# 2.0 Goals and Requirements

The primary goal is to create a standard set of mechanisms, protocols and signals (both inbound and outbound) that enable scalable interactive for IP (internet protocol) video broadcasts.

## 2.1 Content Experience

The following are basic content and experience goals [NOTE – this will become more refined as we begin working with content and media companies].

- ✓ The creation of arbitrary sized groups. A group can range from 1 to thousands (millions?), each seeing the same stream basic historical state.
- ✓ The group experience can change (1 or more) based on input from user(s).
- ✓ A content stream can enter a wait for input state.
- ✓ Streams can respond to an asynchronous input signal.
- ✓ Advertising can be managed in standard video stream insertion techniques but support interactive, 'long form' advertising.
- ✓ Seamless integration on multiple screens (phone, tablet, TV).
- ✓ Input model that supports a wide variety of devices.

- ✓ There needs to be an individual maintenance of state. This should be during stream transmission and state from past transmissions.
- ✓ Signals can drive disparate events across devices and mediums. Example, a signal from a video can trigger events on a tablet.
- ✓ Support for rich story telling techniques. This might include connection to gaming system.
- ✓ System has to be highly extensible and provide for future content experimentation and invention.
- ✓ Content creators need to be assured that their content will run on any infrastructure but still have complete control over where it is allowed to run. Content must have the ability to be optimized for different operating environments.
- ✓ Content creators should be able to embed data payloads for delivery to advertisers and vice versa, with full control of the payload's relative temporal position in the content stream.

## 2.2 Embedded Signals

This protocol is based on signals, both input and embedded. Key features of the embedded signal are:

- ✓ Easy to embed in the broadcast stream. There are a number of standards that currently embed signals in digital video streams. This signal should be as easy to add as currently implemented signals.
- ✓ The signal should have access to user or group state information. The generation and behavior of a signal can vary based on internal state information.
- ✓ It should be able to trigger both internal events and be broadcast to trigger external events.
- ✓ It can put the stream into various states. For example, loop a stream while waiting for an input.
- ✓ It can cause a switch to a different stream in a smooth, frame accurate way.

## 2.3 Input/Output Signals

A key set of goals is the ability to take input directly from users to create compelling interactive experiences. The protocol needs to manage the following:

- ✓ Asynchronous inputs from users from a wide range of devices. Inputs will be normalized to manage if the input comes from a smart phone or a television remote control.
- ✓ Easy to map inputs to broadcast streams.
- ✓ As with embedded signals, input/output signals need to have access to state and trigger context correct responses. For example, certain asynchronous inputs may be ignored until the content is in a state that can manage a response.
- ✓ Input can come from sources like Facebook or Twitter and be selectively overlayed into content streams. The system may reach out to a web site, pull information in and one or a group of users see that information.

## 2.4 Interactivity

The goal is not to provide a precise set of interactive experience but rather a set of capabilities that provide a creative set of tools and a pallet to build compelling experiences.

Key to this is the ability match signals, both outbound and inbound to manage the switching of streams to 1 or many users. State information must be matched to content logic to create experiences.

## 2.5 Data

ADTV will be collecting information for a number of purposes. As with any modern digital system, care will be taken in how this data is collected and used. Appropriate privacy and opt out policies will be designed, communicated and executed. Data will be used to drive advertising decisions, content decisions and help inform what content works and does not work.

# 3.0 Content Use Cases

Here are a few content use cases to illustrate some of the system goals. Clearly, do not judge these on the merits of their entertainment value but as feature demonstrations. Later in the document there will be a retelling from a technical view point.

## 3.1 Game Of Thrones

HBO has decided to create a multi-user, group based, interactive, multi-threaded story based on the Game Of Thrones universe.

Viewers form groups of 6. Each viewer is one particular character. Groups can be randomly assigned or allowed to select themselves. The story(s) are then told from each characters point of view.

At different points in each character's story line the viewer is asked to make a simple character choice, form an alliance with another character or betray someone. After that decision, everyone's story line could be effected. These choices are offered either directly on screen or through another device.

Yes, this would require a great increase in footage, but it could encourage a great deal of engagement and multi viewings as viewers play out different scenarios and try different characters. Ten linear episodes would have to translate into ten times the footage but result in twenty times the individual viewership. There would be a viewing data feedback loop to manage this.

The creator has a complex universe to manage but can create a different style of emotional engagement. Selection of the choice points and intertwining story lines is a new and different story telling pallet.

It should be noted that much of the advertising could also be group based and interactive (see 3.2).

## 3.2 Interactive Ad

Through standard targeting and retargeting methods, it is determined that you are looking to buy a car. Instead of a 30 second spot, you are interrupted for 10 seconds and asked if you would like to test drive one of several cars. The network has previously gone to all the automakers and asked if they would like to participate in this campaign.

The viewer can either decline, request that they are given the choice at the end of their show or jump right in.

The experience can start right at a specific model, such as Ford Fusion or start with a brand message first, aren't Fords great.

The viewer can have any number of experiences. Look at different colors, have a mock test drive, engage in a managed Q&A session. Maps of dealerships could be sent. Coupons for $100 sent if you come in for a test drive. The goal is a much richer engagement than a 30 second spot.

All the while data is being collected that can inform future advertisements.

### 3.3 E-Commerce

An advertiser wants to make it easy and efficient for viewers to purchase items seen in a show. The content producer can 'mark up' the video with signals for when a particular item is on screen. This signal triggers an e-commerce event that shows up on a viewer's tablet and can generate a retarget event so the item shows up on-line elsewhere.

Users can be incented to keep a certain web page open or app running with any number of promotions. For example every hour one item featured will be given away to a viewer with the winner being broadcast to all viewers. If the viewer sees something interesting, they can pause the video to take a deeper dive and maybe even place an order. Alternatively they could trigger a stream switch to a full commercial that they could watch right then or later.

### 3.4 Data Driven Advertising

ADTV can enable some classic digital marketing capabilities. Signals can also be data. This can be used to drive re-targeting and DCO (dynamic content optimization).

[ NOTE – Add deeper description of re-targeting and DCO. Include an example ]

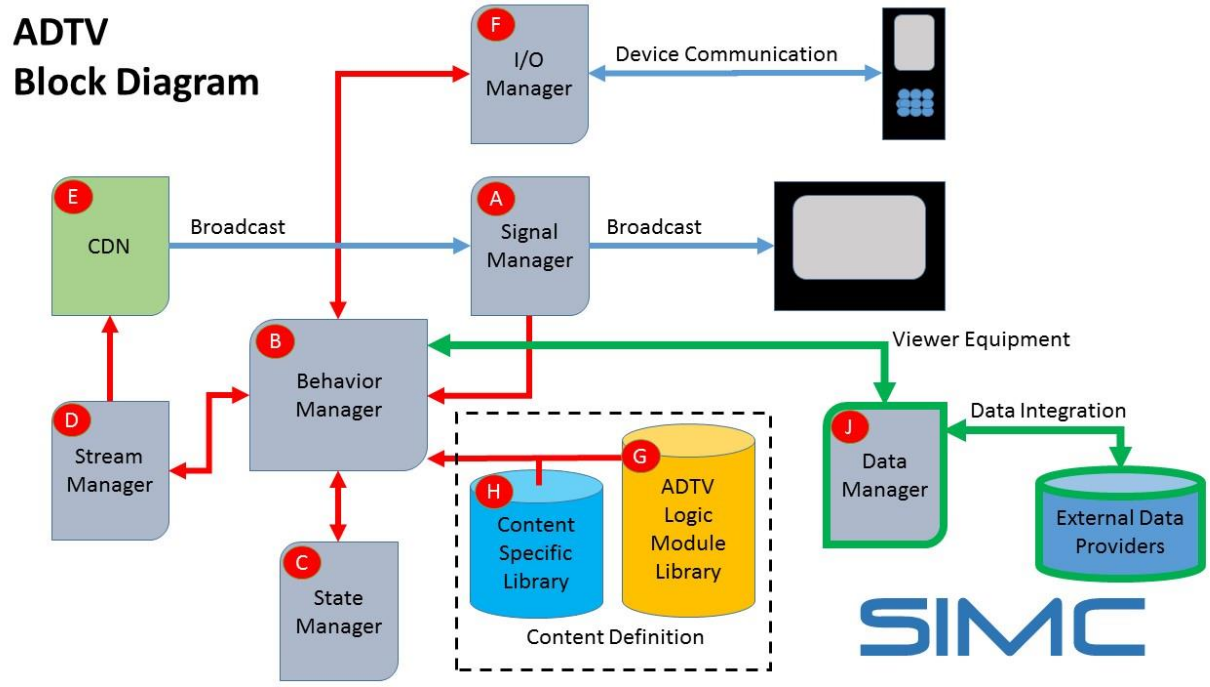
## 3.4 Dynamic Product Placement

A content producer is shooting a scene that takes place in a kitchen. They approach appliance manufacturers to ask if they would like to purchase targeted product placement. LG, Electrolux, and Bosch all agree, but Kenmore declines. The producer shoots the scene three ways - once for each sponsor. During the broadcast the participating sponsors bid for individual viewers, and each viewer sees the version that "their" sponsor paid for.

# 4.0 Theory Of Operation

This is a high level, functional block diagram of ADTV.

## 4.1 Diagram




## 4.2 High Level Description

These components and their interactions will be described in detail later in this document. There will also be examples of the signal and information flows. ADTV is made up of 5 major components called managers.

Digital content IP video streams originate from (E) CDN providers in the usual fashion. These streams could be from a typical internet based CDN provider or originate as an IP stream from an IP based MVPD (multichannel video programming distributor i.e. FiOS). Embedded in this stream are ADTV signals. Much like SCTE 35 (I'll reference docs later) digital cue tones, they are information markers that have been inserted into the stream. Invisible to the viewers, they are detected and responded to by hardware/software that is part of the transmission of the stream.

The (A) Signal Manager 'listens' to the stream to detect any ADTV signals. Its sole purpose is to detect those signals and transmit that information to the (B) Behavior Manager. This (A) Signal Manager has been designed this way to minimize integration with IP services and MVPD plants.

The (B) Behavior Manager takes ADTV signals and makes decisions that drive the user experience. It does this by interpreting signal data and by referencing information in the (C) State Manager has all the information change or modify what the viewer is seeing. Based on its logic it can completely switch the stream or add overlay information to a single viewer or group of viewers. The (B) Behavior Manager can also use information that it gets, either synchronously or asynchronously, from the (F) I/O Manager.

Content is driven by two sources. Programed logic comes from a central library called (G) ADTV Logic Module Library, a central repository of program code that drives the experiences and a (H) Content Specific Library where information specific data is stored. For example, the content maybe driven by a

state machine specified in the (G) ADTV Logic Library, and the actual state transition tables and references to different video encodings would be stored in the (H) Content Specific Library.

The (B) Behavior Manager can also update the (C) State Manager and add information that changes the internal state of a viewer or group. This is, for example, how a multi-threaded, interactive story would be managed. The (B) Behavior Manager can also, through communications with the (F) I/O Manager, trigger events on a viewer's device. For example, something appearing on a viewer's smart phone could be made to appear based on what is being streamed into the viewers TV.

Once (B) Behavior Manager has made its decision, it communicates with the (D) Stream Manager. The (D) Stream Manager can instruct the CDN or MVPD plant to switch to a different stream, again, either for a single viewer or group. It can also add a graphic overlay to the video stream. The (D) Stream Manager is the other major touchpoint to the streaming infrastructure. The (H) Content Specific Library may have different encoded video sources to optimize across infrastructure plants.

The (F) I/O Manager handles communication with viewer devices. It is unaware of context or any state information. It simply routes input and output signals to and from viewer devices. It also helps in the matching of devices to streams. If part of the content scenario involves triggering separate events on other devices, the (F) I/O manager handles that outbound communication and is driven by the (B) Behavior Manager.

[ NOTE – need to add example sequence for (J) Data Manager. This comes into play for any experiences driven by external data like re-targeting and DCO ]

It should be noted that there is no requirement that any of the managers be created by the same provider. There is a precise communications specification that ensures interoperability between components.

# 5.0 Behavior Manager

A key component and ADTV is the Behavior Manager. It drives what the viewer experiences and is the most intricate and complex of the managers. It must take in the various inputs, combine them with historical and state information about viewers and drive the user's experience. The quality of the experience will depend on how well a creator can experience themselves through the mechanisms we provide here.

The Behavior Manager is in the middle of coordinating input (from the I/O Manager), incorporating user and group states (from the State Manager) and causing experiences to be reflected on viewer's devices (by signaling the Stream Manager and I/O manager). It executes the combinational logic around that coordination.

## 5.1 Representation of Programmatic Logic

Programming logic is represented in NODE.JS [NOTE – we should discuss!]. This was chosen for its speed and scalability. There a standard set of interfaces between the Behavior Manager and the other Managers in the system.

[NOTE – READ CAREFULLY, THE FOLLOWING IS IMPORTANT]

To support flexibility and expansion that can keep pace with content invention, ADTV features a plug in capability where code, called ADTV Logic Modules can be accessed from a global library service. This is done to insure interoperability across infrastructures. Content running on an IP service (NetFlix, Amazon) can also run, unaltered on an MVPD infrastructure.

All content will be associated with a ADTV Logic Module. The programming will reside in a library that is maintained by SIMC. This library is free to use. **It is a requirement** that all code that drives action in the Behavior Manager be distributed from this central library. This is how ADTV provides interoperability and decouples content creation from technical deployment.

Logic Modules can be meta-systems onto themselves. For example, a studio designs an interactive story model that uses a directed graph style state machine. The story is a complex web of interacting viewer states (see Game Of Thrones example). To effectively author and manage this complexity, the story(s) are fully represented by states and state transitions. An authoring tool can be created and an internal representation for both states and their transitions can also be created. Coupled with the State Manager and interpretive engine could be created that uses this representation as a program execution engine. This single meta-system can drive different content, run on the same logic module and differ only in state transition codes and actions. With this Logic Module residing the SIMC library all players would be able to execute the content.

This abstraction and separation of code and data would allow content owners to still control access and distribution. While the Logic Modules are freely available, the data that defines the behavior of the content is separately managed and rights to that content will be controlled by the owners.

Content is a combination of code libraries and data that drive those libraries.

## 5.2 Communication With Other Managers

ADTV provides a standard set of methods for communicating with other managers. The Behavior Manager can query the State Manager's data base through a consistent interface. Messages from the I/O Manager come in as asynchronous events. The Behavior Manager also sends asynchronous messages to the Stream Manager.

Messages in the system will be context sensitive. Data payloads will be specified and driven by code from the ADTV Logic Module with some of the data coming from the Content Specific Library. Some messages will be universal and available to all applications. Some will be defined and usable only by the specific content instance.

Commands will usually come from the Signal Manager, however, commands can also be embedded as data. For example, a state machine engine could execute commands based on internal state transition rules. See later examples.

## 5.3 Communication With Viewers

The Behavior Manager gets its input asynchronously from the I/O Manager. From the viewer experience perspective it may seem like a synchronous activity but internally all communication for the I/O manager is asynchronous. The I/O manager has no access to state or content information, that is all managed by the Behavior Manager.

For example, program logic wants to wait for a user input, a seemingly synchronous activity. There is a signal in the stream to indicate this. The Behavior Manager instructs the Stream Manager to loop a stream for 30 seconds (maybe even add a count down graphic overlay). It also signals the State manager that the viewer is in a 'waiting for input' state. So now the viewer is seeing a looping clip. When the response, say a button press on a remote control comes in, that event is sent to the Behavior Manager. The Behavior Manager can check that the viewer is in a 'wait' state, then based on logic from the Logic Module and Content Specific Data, it can trigger the appropriate event (switch to a different video stream for example) and put the viewer into a different state. If a signal comes in from the I/O manager and the viewer is not in a 'wait' state there could be a default action or it is simply ignored. These decisions are completely in the content designers hands..

### 5.4 Timed communications

Events may also be sent to viewers asynchronously. One experience model may involve messages that occur outside a standard viewing session. A viewer may receive a text message that an event has occurred. Some of the plot may have advanced or the experience can automatically generate events in the future. The Behavior Manager is able to continue an experience in the background and communicate with users as needed.

### 5.5 Communication with Content Specific Library

The ADTV Logic Modules specify programmatic logic, the actual information that would drive the behavior is in the Content Specific Library. This information may take the form of pure data or commands. The command structure and syntax would match that of the Signal Manager. This allows events to be triggered from within the stream or programmatically in data (See Content Specific Library section).

***Additionally, there is a mapping from symbolic, logical video stream references to physical streams. The Content Specific Library would have this information. It is an open question as to whether the Behavior Manager or the Stream Manager resolves these references – TBD.***

## 6.0 Signal Manager

The Signal Manager 'listens' to the stream to detect any ADTV signals. Its sole purpose is to detect those signals and transmit that information to the Behavior Manager. It has been designed this way to minimize integration with IP services and MVPD plants.

### 6.1 Signal

[NOTE – The details here are still under discussion. The notion that the signal is based on SCTE 35 is a suggestion being investigated]

The basic ADTV signal is an SCTE 35 digital cue tone with additional data occupying the unused data fields in the SCTE 35 specification. This allows ADTV to make use of existing SCTE infrastructure and existing tools. It also makes advertising integration more straight forward as it becomes incremental to existing technology.

These signals are embedded in the video stream using authoring and work tools designed to support this protocol.

Examples of data embedded in the signal (a full data spec will flesh this out. This list is to give a sense of requirements). [NOTE – data will have a very assembly language like feel to them]

| | |
|---|---|
| Logic Module | Specifies the context for this signal and helps guides its interpretation. This tells the Behavior Manager which Logic Module to use in executing the interpretation for this signal. The same command or data may cause different behaviors depending on the Logic Module context. |
| User or Group ID | Specifies a particular view or group of viewers that this command affects. This brings all the appropriate state variables into scope. |
| Command | Signals trigger actions by issuing commands. Commands can have 1 or more parameters. Examples might be:<br>- Loop clip (X) waiting for input<br>- Execute a state transition for a viewer. For example a story event has occurred and that information needs to change the state of a character<br>- Overlay macro – example, overlay a twitter feed<br>- External Device Trigger – send information to a second device. Example, show a product page in an app.<br>- Logic Module Preload – Signals the Behavior Manager which Logic Modules need to be loaded. |
| Parameter(s) | Commands may have 1 or more parameters. These parameters may be absolute like wait though 6 loops for input. The can also be relative or symbolic. For example, an input video loop may be a symbolic reference that might translate to different actual streams based on device type or URLs based on infrastructure (Netfilx vs. FiOS) |

Examples (for illustration only)

Commands are of the form

Logic Module Command Parameter(s)

```
IO_CNTRL      LOOP_FOR_INPUT(ID)    6,stop_watch,7      ; Loop the stop watch
                                                        ; stream 6 times
                                                        ; Waiting for input
                                                        ; current clip is 7

STORY_TELLER  SWCH_CHAR(ID)         fred, angry         ; Character fred is
                                                        ; switched to angry.
                                                        ; this may or may not trigger a
                                                        ; video for fred
COMMERCE      SHOW_OFFER9(ID)       dress,1765          ;Have device show dress
                                                        ; SKU# 1765
```

See the execution examples for a fuller set of samples.

## 7.0 Stream Manager

When the Behavior Manager makes a decision to change what a viewer or group of viewers is seeing, it calls the Stream Manager. The Stream manager directly communicates with the IP video streaming systems at the broadcasters. It needs IP address of the stream, and fully resolved stream source [NOTE – may also need an insertion time code and possibly device ID and device type].

The Stream Manager is also responsible for coordinating the frame accurate splicing. This might require an additional signal to identify frame accurate splice points (like SCTE 35).

Stream sources may have a symbolic reference. A logical stream, such as a looping clock, may resolve into one of many physical streams optimized for device or infrastructure. ***The current design is to have the Behavior Manager handle the look up and resolution to physical streams. It may make more sense to have the Stream Manager take care of the look up and resolution as it is closer to the actual infrastructure, better isolates the Behavior Manager from actual physical streams and there maybe more subtleties to address. TBD***

### 7.1 Overlay Manager

Content experiences may require the overlay of text onto video streams. The content design could require the display of a question, showing user text messages, or feeds from other sources. The Overlay Manager (as part of the Stream Manager [NOTE – may break this component out]) will manage communications with the streaming broadcast to insert these overlays onto the out going signal.

There is a full graphic and text specification subsystem to describe the visual aspects of the overlay. It describes things like; frame dimension, background color, border style, text color, text style, font, on screen duration.

The actual text and graphic content would come from Behavior Manager with the actual data sourced in the Content Specific Library. Overlays can target individual viewers or groups of viewers. There may also be an on off state that hides overlays. The Behavior Manager will manage the sending or blocking of overlays.

## 8.0 I/O Manager

The I/O Manager is responsible for communications with non-streaming devices that may be part of the viewer's experience. Sample events could include;

- Triggering the display of a particular web page.

- Communicate directly with an app running on a tablet or smart phone.

- Gathering input from an app, web page or device remote control.

- One of these inputs could be to pause or replay a portion of a stream.

### 8.1 Input

As discussed in the Behavior Manager description, the inputs from viewers are assumed to be asynchronous. The I/O manager is not aware of any state information, it simply passes events to the Behavior Manager as they occur. The Behavior Manager will respond or ignore them based on state and rules.

There will be a method for identifying individuals (see Device Management). Input from a web page may be a call to a service that has the data and a unique identifier as part of the message. The same would be true for an app.

Gathering input from remote control device would require working with each vendor or system to generate the signals. The I/O manager would normalize this input before sending it on to the Behavior

Manager. Additionally if this input is coupled with a visual, say a YES/NO menu, the I/O manager is responsible for generating the on screen behavior and normalizing the response. Some use cases would include;

- The device is a browser running on a phone/tablet. The I/O Manager can use standard web technology to manage that experience.

- The visual is displayed in the stream, but the input is sourced from another device. This might be the scenario with a group screen (DOOH example). The device would need to have been paired with the broadcast stream and the proper web or app signals taken in.

- The visual is displayed in the stream and any UX feedback and selection is made with a hardware remote control (PRESS A for YES). This requires a tight integration with the platform and hardware vendor and will require an integration on a case by case basis.

There will be an abstraction that would allow the Behavior Manager to issue the request for input that would allow the I/O Manager to create the correct experience for the viewer. This might include the Behavior Manager also communication with the Stream Manager to generate the correct overlay [NOTE – this is a bit fuzzy and would need some more thinking]

## 8.2 Output

Content can communicate with viewers on other devices. In the simple case, standard dynamic web and app technologies can be used. The I/O Manager would be given an IP address (or possibly another form of ID) and would communicate with the device.

As with the Overlay Manager, there is a meta format for displaying information.

For an example of how the input and output functions of the I/O manager see the e-commerce example.

[NOTE – there is currently no mechanism for communicating directly with the Stream Manager for output tasks. This may need to be rethought as the I/O manager may need to insert overlays into stream as part of its output normalization]

Since the Behavior Manager can trigger out of session messages (text messages for example), the I/O Manager is able to send these. A stream doesn't need to be currently running for the I/O manager to engage with the viewer.

## 8.3 Device Management

[NOTE – device management has been made part of the I/O manager. This may need to change architecturally]

For a fuller discussion see section on IP and Device Management. The I/O manager is responsible for device communications and must naturally be very precise about how that communications reaches the correct user at the right time and in the right format. There is also a goal to have the content triggered and managed by the Behavior Manager.

This is done by storing and maintaining this matching information in the State Manager. When the Behavior Manager issues a display command to the I/O manager, it will also include in the command information about the device dependent form of the output.

One function the I/O manager can perform is matching devices to individual streams. In group display settings (DOOH) where many people are viewing a single public screen, simply adding a unique code overlay to an output stream, viewers could simply enter that number in their device and create a coupling between the stream and the device. This helps get around some of the complexities of device identification through IP means.

# 9.0 State Manager

Managing the viewer's experience will require the management of states. There are two classifications of viewer states, device and content.

## 9.1 Device State

ADTV functions in a very heterogeneous environment. State that matches viewer devices with streams across hardware platforms and is optimized for experience is key. This information is stored on a user basis and experience basis. The viewer may change devices and settings over time. For example, most viewing could occur at home on FiOS but occasionally will occur on a tablet serviced by NetFlix. Many of the mechanics for I/O would be different but the Behavior Manager needs to be isolated for these issues.

This device state information is stored in the State Manager then passed along passively to the I/O manager by the Behavior Manager. For example, the content is requiring that a text overlay be shown. The Behavior Manager issues the command to the Stream Manager and the device state is simply passed along.

***[NOTE – It may be the case that the Stream Manager and the I/O manager need access to the device state and in turn to the Content Specific Library to actually simplify things – to be discussed TBD]***

Another example would be if the user has requested that they see not on-screen text feeds. This information would be stored as a device state and the I/O manager can act accordingly.

An example discussion with the I/O manager, the notion that viewers can self pair devices with streams can also be managed by storing the pairing data as part of the device state. Additionally as increasingly sophisticated auto pairing technologies are devised, the device state can store that information as well.

## 9.2 Content

Content may also require the management of state. The Behavior Manager uses the State Manager store content specific state information. ADTV Logic Modules have an abstraction layer that can allow code to reference and modify content state information.

This state information could persist only with in a session (and advertisement) or span multiple session (episodic show).

## 9.3 Groups

Content may have the notion of groups. These groups are collections of viewers that maybe interacting around content. There is a notion of group states and can be managed and accessed by ADTV Logic Module code just as individual state information is managed.

The process and UX for creating groups, either sign up or random assignment, is managed through ADTV Logic Module code.

# 10.0 Embedded Signal

It has been suggested that we use the SCTE 35 standard as our basis for the embedded signal. This section will be expanded as that decision is reached.

# 11.0 Input Signal

User inputs are handled by the I/O manager. The current design under consideration is simply web services with the I/O manager simply making calls with the correct parameters. This section will be expanded as this solution is researched.

# 12.0 IP Address And Device Management

One of the key complexities ADTV addresses is messaging across disparate devices, platforms and user experiences. This is complex issue given the state of devices, in-home networks and mobile IP networks. One simple solution might be the viewer simply entering a code number that matches a number overlayed onto a video stream.

This section will be expanded with more research.

# 13.0 ADTV Logic Modules

The Behavior Manager acts more like a switch board. Experiences are driven from a set of code bases that reside in a library called ADTV Logic Modules

## 13.1 Module Access and Organization

There is a central and freely accessible library of ADTV Logic Modules. The Content Specific Library will have a preamble that will specify Logic Modules that are used in a particular content experience. This will trigger a pre-load and caching of the code base.

There will be a published set of guidelines regarding how to write code. SIMC will also provide a validation suite that will test code in closed environment to help authors make sure that code will run across the supported devices and platforms.

Authors can choose to make their Logic Modules available for general use or restrict it to only content they produce. We anticipate there will be a mix of free to use and private Logic Modules.

There will be a standard set of base Logic Modules to handle common tasks. For example, the machinery for creating and maintaining groups resides in a common Module. The behavior will be driven by code but the look of the menus would be driven by data in the Content Specific Library for that content. Individual producers can create their own standard modules or create modification of existing standard ones.

The code base is NODE.JS [NOTE – to be discussed]. Any server infrastructure needs to be able to run NODE.JS. It is assumed that when a content is set to run, the code is copied. There is also a publish and subscribe model to manage updates to Logic Modules.

ADTV is a very data driven model. Logic Modules can, in fact, be full interpretive data engines that execute higher level constructs. For example, you can imagine experiences that are driven by state/transition diagrams. Tools could manage those graphs and produce data that is interpreted by the Logic Module.

Logic Modules can reference and call code in other Modules (provided they have the right permissions). Most calls will have a user or group identifier.

### 13.2 Command Processing

The core of a Logic Module is command processing. These commands can originate from a signal in the video stream or be generated by internal processing. Since signals have a Logic Module field, the correct Module will called by the Behavior Manager. In other cases the Logic Module must be specifically indicated.

ADTV provides a standard set of command processing utilities. A general command processor is available to route flow of control to the correct execution code. This uses an event driven processing model.

Additionally there are routines available to aid in extracting command parameters.

#### 13.2.1 Variable abstraction and Symbol Table Management

Information can be referenced within a Logic Module symbolically. For example, a video clip of a stop watch used to loop and indicate a 'wait for input' user state can be referenced by a single symbol (i.e. STOP_WATCH).

The actual clip is defined within the Content Specific Library. Additionally the Content Specific Library can specify different clips that match different display environments. The Logic Module is shielded from this complexity.

Most commands will have an individual or group ID parameters. Through the State Manager, this will bring the correct set of variables into scope. There will be a set of commands, like broadcast a message to all viewers that will not require an ID but the bulk of commands will.

The system provides a robust symbol table management set of functions that can maintain symbolic information and various scopes; global, group, and individual. This scope can persist for a session, across sessions and be accessible if background logic is being executed. The actual data store is the State Manager.

The symbol table manager allows Logic Modules to create and reference information stored in the State Manager without having to worry about scoping, scale and cross platform issues.

## 14.0 Content Specific Libraries

ADTV is a highly data driven system. Content can be viewed as a set of video streams, graphic overlays and data to drive extensible rules engines. This content is created by a specialized tool set (see Tools section) that produces this information in a form that can be executed by ADTV installations.

When contented is 'mounted' on a system, it is actually a Content Specific Library that is loaded and drives the specific instance of a viewer's experience.

## 14.1 Preamble

Each Content Specific Library will contain a preamble of basic information. This includes a list of all the ADTV Logic Modules required to execute the content. The loader (to be defined later) can then set up the runtime environment. These modules are in a central library of modules. This ensures a run anywhere capability.

## 14.2 Symbol Mapping

Part of the content creation process is the actual production of visual assets. These assets, either through automated means or human created means, will playback in a wide range of environments. To accomplish this, Content Specific Libraries include an initial symbol table that manages these assets.

For example, to take our stop watch loop example, there maybe multiple versions that have been created. Sometimes to deal with device variation, sometimes to deal with distribution issues (video needs to be available on multiple infrastructures for example). There maybe a single symbol STOP_WATCH that actually references several video assets. Down stream components can dereference and select the correct stream to broadcast. ***[NOTE – this is why it might be necessary to give access to this library to both the Stream Manager and/or the I/O Manager. Alternatively the Behavior Manager can manager routing the correct information – TBD]***

The pre-loaded symbol table will also contain data that Logic Modules my need to reference symbolically. Examples might include; error messages, internationalization data, other global information used by the content

## 14.3 Meta Systems

[NOTE – VERY IMPORTANT]

To support ease of expansion and streamlined work flow, the ADTV architecture has been designed to support user defined, data driven authoring systems.

By creating data models and tools, content producers can invent languages to express different classes of content. Since it is possible to design a ADTV Logic Module that functions as a data interpreter, a designer can create languages for creating and managing new content.

Let's use the Game of Thrones state machine as an example. We will also use it later in the content usage examples.

Suppose the user experience can be described as a large state transition table. Each viewer is one of a few characters and state is determined by interactions with other viewers and the overall story arc for each character. Transitions are all triggered by events in the broadcast or actions by other characters, in each case the state transition can be completely represented by data.

Now an authoring tool can be built that can generat this data and publish it in the form required by ADTV. It can be optimized for efficient content creation for this class of experience.

You can imagine any number of authoring and runtime environments that can be created and run on the same system.

# 15.0 Advertising

Advertising will follow very much the VPAID model where Content Specific Libraries replace FLASH. There is no difference between the architecture that supports an interactive advertisement and interactive content.

There will be a set of creation tools and an information tracking system put in place similar to the tracking beacons found in VAST.

Because of the plan to use SCTE 35 as a signal, this will make integration with standard system (SCTE 130 for example) much easier.

[NOTE – This needs more design work –TBD]

# 16.0 Data Manager

This system manages the interactions to the external data world

[NOTE – more detail to be added. This is how re-targeting and DCO will be implemented.]

# 17.0 Data

During operation there will be a number of opportunities to collect data. This will range from behavioral information to information to help optimize advertising.

[NOTE – More on this later]

# 18.0 Tools

[NOTE – I will only cover the surface of the tools. It might warrant a separate document]

Key to the ADTV architecture and strategy is the ability to create robust tools to manage the creation and deployment of content. The other goal is not to impose any particular structure on the content or on the creation process. It is a system that allows multiple tools and tool strategies to be built, supported and deployed at scale.

ADTV is a data driven system where both the data forms and the interpretive code is user definable.

## 18.1 Managing ADTV Logic Modules

Creation of a tool has to go hand in hand with the creation of, or at least understanding of the Logic Module the tool is to support. Any authoring tool is very tightly coupled to a Logic Module.

This document has been using the Game of Thrones as an example content. In this example, the rules and interpretation of the coded state machine have to be very precise and coordinated. Much like writing a virtual machine interpreter, the rules of the virtual machine are laid out and fixed.

## 18.2 Managing Content Specific Libraries.

The output of any tool is a Content Specific Library. As described earlier, this file is the data embodiment of content.

### 18.2.1 Tool Creation Support

ADTV provides a set of tools and libraries to help tool writers with the integration and proper generation of Content Specific Libraries. Some of these tool routines will change over time as the system matures so tool writers will need to include them in the creation of their own tools

### 18.2.2 Auto Generation of Preambles

Preambles that cause the auto load of ADTV Logic Modules at runtime are essential for deployment and execution. A standard set of tools will generate this preamble for tool writers.

Same for the generation of symbol tables and data maps to aid in the dereferencing of media assets. A standard set of libraries will be provided and maintained.

## 18.3 Debugger

There is a standard debugging interface that will allow content to be run on closed systems and generate debug information to assist in both tool development, content testing and trouble shooting.

# 19.0 Open Source

All components of ADTV will be open source. The interfaces between components will also be published to allow multiple vendors the ability to mix and match parts.

The central library for ADTV Logic Modules will be freely accessible and operated by SIMC. Content producers can publish Logic Modules to the library. This lets all infrastructures interoperate with the same content. Publishers can choose not to make their modules available to other publishers.

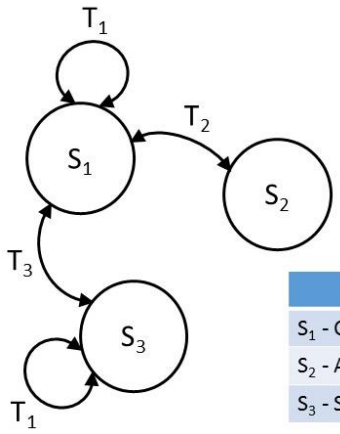
# 20.0 Examples

These are the same content use case examples discussed earlier but viewed from the stand point of technical execution.

## 20.1 Game of Thrones

Assume the system that was described in the Meta section of Content Specific Libraries exists and was used to create this content. The story is completely described in a state/transition graph.

A viewer has selected a character and is part of a group of characters. The video that viewer sees is unique to that character and the state of that character and that includes the state in relation to other character's states.




| | $T_1$ – Sleep | $T_2$ - Struck | $T_3$ – Yelled |
|---|---|---|---|
| $S_1$ - Calm | $S_1$ | $S_2$ | $S_3$ |
| $S_2$ - Angry | UNDEF | $S_1$ | UNDEF |
| $S_3$ - Sad | $S_3$ | UNDEF | $S_1$ |

State Transition Table

A sample event sequence might look like this

1. A segment that is being broadcast to a viewer contains a signal that indicates that user input is required.
2. The Signal Manager detects this and sends it on the the Behavior Manager.
3. The signal indicates a state transition question and has an identifier associated with it as well has a Logic Module ID as well as the viewer ID
4. The Behavior Manager calls the Logic Module with the embedded command
5. The Logic Module takes the command and by looking up in the Content Specific Library, retrieves current state of the viewer and can then look up the transition text for that state/character.
6. The Logic Module returns the transition text and the command to loop a wait video.
7. The Logic Module also puts the viewer in a 'waiting for input' state.
8. The Behavior Manager signals the stream manager to loop the wait video, the Behavior Manager might dereference the actual video asset (TBD)
9. The Behavior Manager passes the text question to the I/O manager to display the text of the question, "Kill the King <YES/NO>" on the correct device. Again, either the Behavior Manager or the I/O Manager will be responsible for the dereference.
10. Now the video stream is looping the wait video clip and the question has been displayed appropriately to the viewer.
11. Once the viewer has returned an answer, the I/O Manager returns only that <YES> or <NO> has been returned.
12. The Behavior Manager calls the Logic Module with the INPUT, Logic Module ID, User ID, data <YES>.
13. As before the Logic Model looks up the current state, user, character etc and would find a new state to transition to. This new state would include a new video stream to switch to. It returns this new video stream to the Behavior Manager.
14. Just as before, the Behavior Manager instructs the Stream Manager to switch to the new stream for that user (dereferencing TBD).

This seems complex but the entire sequence has rules that are fixed and what is reflected to the viewer is controlled by data generated by a robust tool.

## 20.2 Interactive Ad

Refer to the Interactive Ad Use Case presented earlier.

Here is a possible sequence of events to execute the interactive ad described in the example.

Since ADTV uses SCTE 35 digital cue tone technology, it can issue a standard VAST call to ad networks, If what is returned is a standard video impression, it can be inserted and managed as a standard video ad break.

If what is returned is a ADTV interactive ad, the data payload would be information that would drive the experience. Much like VPAID does now, the data payload would be a Content Specific Library. It would be assumed that the Logic Module would have been included in the content's source preamble.

Much like the Game of Thrones example, the experience is a set of input questions whose answers are associated with states, transitions and video streams.

A number of new events would need to be created such as: show ad at the end of broadcast, send email etc. All the data generated would be sent back to the advertisement source using standard means. In fact part of both the Logic Module and the Content Specific Library could specify this.

A set of general tools would be produced that would aid in the production of these advertisements but not box advertisers into any specific model.

## 20.3 E-Commerce

Refer to the E-Commerce Use Case Example.

Here is a possible sequence of events.

1. If users want to access e-commerce opportunities, they are instructed to open a browser on another device.
2. The Signal Manager detects an offer 'mark-up' for a particular dress and SKU
3. The Signal Manager passes the command on to the Behavior Manager.
4. Since it was known in the preamble for the content the Logic Module has been loaded.
5. The Behavior Manager sends the command to the Logic Module
6. The Logic Module can take the SKU for the dress and formulate a URL for that item
7. The Logic Module issues a command to the Behavior Manager to show a particular URL on a device to the viewer.
8. The Behavior Manager sends to the I/O manager the instruction to display the URL and to have the Stream Manager generate an on screen indicator that an item is being displayed on the viewers device.
9. The I/O manager gives the viewer the option to pause the video (this is handled in previously described mechanisms). It then switches the display to the e-commerce page with that item.

## 2.4 Re-targeting And DCO

[Note- add detailed discussion here- this may become a separate specification]