# Agile Media Blueprint

## Creating and monetizing content using the Internet technology platform

Richard Cartwright - CTO - Streampunk Media Ltd.
Brad Gilmer - President - Gilmer & Associates, Inc.

April 2, 2018

A white paper from
Streampunk Media Ltd.

**Streampunk Media**

# Introduction - What is the Agile Media Blueprint?

The Agile Media Blueprint (AMB) is a plan that enables media companies and suppliers to build highly flexible and scalable systems that run on the same platform as the Internet. This allows media companies to create and monetize content more effectively than if they were using conventional facilities. The AMB makes use of all of the hardware, software, networking and associated components used to run world-wide huge-scale systems such as Twitter. We refer to all of this equipment and software as “the platform”, but please remember that every time we use that term, we are simply referring to the technology that runs the Internet.

The AMB is the topic of a discussion paper of the Advanced Media Workflow Association (https://amwa.tv/) and much of the content in this document is common with that paper. What this white paper does in addition is to provide technical detail around the shape of a possible contribution to the AMWA’s technical processes, much of which is backed by Streampunk Media’s open source prototypes. Please use the table of contents to access the sections that are more relevant to you as a reader.

## What problem does the Agile Media Blueprint address?

It would be nice if, when a new business opportunity presented itself, a media company could quickly spin up the appropriate infrastructure and resources to take advantage of that opportunity. But with vertically integrated facilities and tightly integrated software applications, it becomes difficult, perhaps impossible, to take advantage of a new opportunity, particularly if the new opportunity involves even a modest level of personalization. This can be especially true if the opportunity is not initially expected to generate revenue, and if at the same time, the opportunity requires development of new infrastructure or workflows.

Today, it is not uncommon to find at least two major technical operations in a media company; one operation dedicated to traditional linear feeds, the other dedicated to serving web content to mobile consumers. This did not pose any particular problem when the web operation was small. However, as web operations have grown, visibility has increased. An Internet outage, whether for a major media company or for Twitter, gets noticed.

This leaves the media company in the unenviable position of running two major operations that have similar business requirements - to deliver content to viewers - but that are far from identical with regard to technology and infrastructure. Furthermore, the skill sets necessary to run and support the two operations are different. To make matters worse, the increase in operating cost related to running two technical operations comes at a time when the industry is experiencing competition from companies whose stock-in-trade has been developing and deploying Internet Technology. Those technology companies are bringing the expertise they developed in online

retailing, banking, and stock trading to bear on the media industry, creating new products that cannot be matched by broadcasters using traditional facilities.

Media companies need a way to streamline operations, reduce the friction involved in monetizing content, and increase their flexibility in delivering content to an ever-widening number of consumer platforms, while at the same time holding the line on costs. They also need a way to deliver higher quality streams (e.g. 4k, 8k), and future-oriented products that may have more in common with video games than traditional broadcasting. The Agile Media Blueprint provides a way forward for media companies.

# Architectural Overview

## Grains

The Agile Media Blueprint is based upon an architecture that captures, manipulates and transports essence streams in the form of data “grains”; small units of video, audio, and data essence such as teletext or closed captioning. Cameras, microphones and other sources emit flows of grains onto a network. Each grain is stamped with a Precision Time Protocol (PTP) timestamp, recording the time at which the visual image or sound wave was captured. Each grain is also given a unique and persistent identity which is permanently tied to, and associated with that grain. Monitors, speakers, and other receivers can reproduce flows of grains as images, sound or data.

Readers may find it helpful to refer to the Agile Media Blueprint diagram in the appendix as they read through the following material. Those looking for more detail are referred to the section titled, “Technical Discussion”.

Grains are persisted (kept) in an object store. This is the same generic storage technology used by millions of Internet applications today. A very fast, very large cache sits above the object store, temporarily storing and serving out essence that is requested frequently. Essence that is requested less frequently is retrieved from the object store, since it is not found in the cache.

## Composable atomic functions

Workflows are accomplished by chaining together the required composable atomic functions. As figure 1 below shows, these functions are small bits of software (atomic) that take in content, perform a transformation, and emit the transformed content. A transform could be a fade between two video sources, it could be a key function that superimposes one video on top of another, or any one of a number of other functions we commonly perform on content. Each function is given an identifier which allows it to be unambiguously referred to and reused. This is shown by the small “ID” tag attached to the function in the figure.

We believe it should be possible to develop an open, collaboratively developed API (a content API) that serves as a consistent interface between the function and other parts of the system. The software interface for all functions would be through this API. This is shown at the bottom of figure 1. The Content API provides a uniform way to store, search, and retrieve grains from the object store, and provides an enabling layer of interoperability. Suppliers are free to do their best work, implementing functions and systems on top of the Content API.

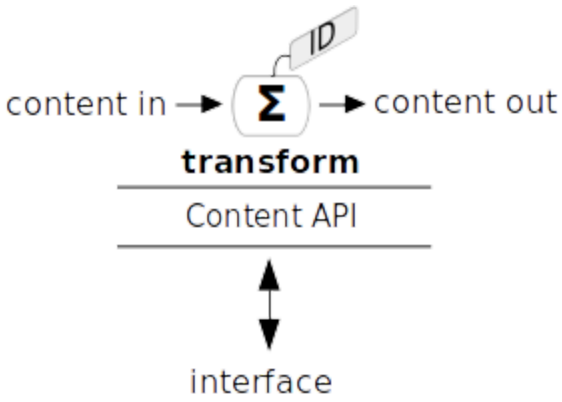



**Figure 1. Composable Atomic Functions**

Importantly, the function exists by itself. It has no knowledge of other functions, and it makes no assumptions about the workflow either preceding or following it. In this way, the service this function provides stands on its own, and may be used in any number of workflows where the function is needed. The notion of putting together a number of different functions is referred to as composability, and chaining together these different functions is sometimes referred to as service chaining.

### Service chaining

As figure 2 shows below, functions ‘a’, ‘b’, and ‘c’ are chained together, perhaps in a master control switcher. Function ‘a’ is a video cross-fade. Function ‘b’ is a key function that superimposes a moving ‘bug’ on top of the primary video feed. Function ‘c’ inserts a watermark for tracking purposes.

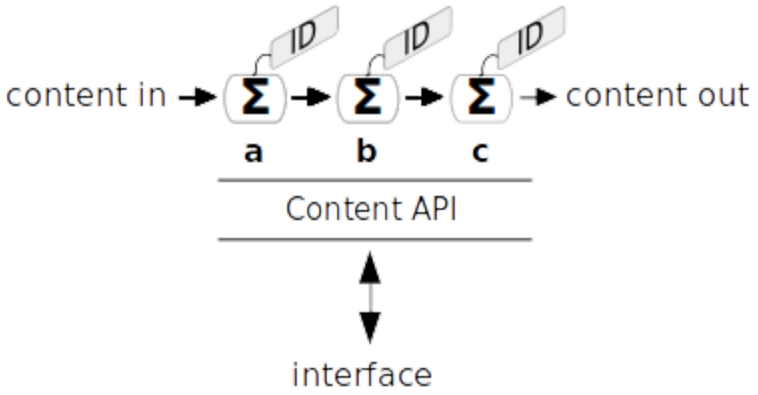



**Figure 2. Chaining together a master control switcher workflow.**

Taken together, it should be possible to replicate the functionality of a modern master control switcher, for example, as a chain of atomic functions.

When the need arises, it could be possible to chain together another group of functions to perform a different workflow. Some of these functions might be the same as were used in the master control switcher and some might be different. This is shown in figure 3 below. Note that in this case a new function, function 'f' has been added. Functions 'a' and 'c' are reused, but they are in a different order. Function 'f' might be a function that synchronizes audio to video. The exact purpose of this workflow is not important. The critical points are that each function performs a small piece of work; that the function stands alone; that it uses a standard interface, the Content API; and that the functions may be chained together to create different workflows to accomplish a wide range of tasks.

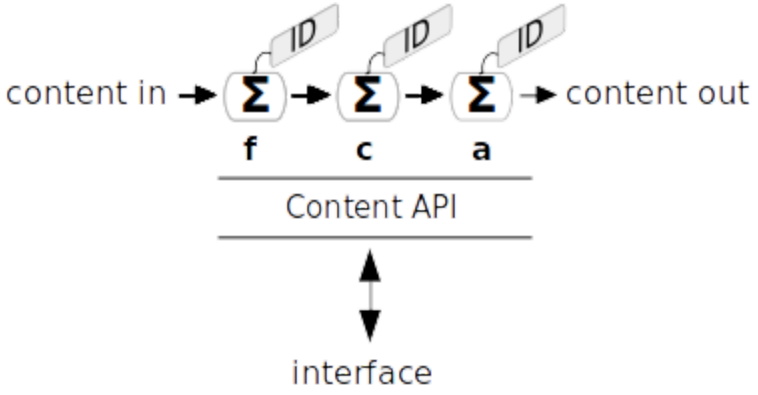



**Figure 3. Services may be reused to create different workflows**

## Service chain automation

One of the strengths of working with composable functions is that workflows may be built and torn down as required. As you might imagine, the construction and deployment of these service chains may be orchestrated using widely-available tools that are used to build workflow chains for the back-ends of major Internet applications. Different approaches to function composition can be applied to enable and optimize media workflows, for example:

- Service Function Chaining[1] (SFC) in the network;
- microservices running in containers, such as kubernetes[2];
- software modules linked ad-hoc at runtime by distributed message queues (e.g. reliable queue in Redis[3]).

In addition to building service chains, some automation tools dynamically create new instances of functions in the workflow chain if a function were to stop responding, or if the load on a particular service were to become too high. This dynamic, automated scalability is one of the keys to using professional media in a cloud-fit environment where performance needs to be monitored and actions need to be taken if failures occur.

## Security

Security is built into the platform, and by extension, the AMB, from the outset. The threat to media companies is continually evolving. This threat only continues to grow as media companies increasingly adopt IT technologies. By leveraging the platform, the AMB takes advantage of cutting-edge best practices in security, using approaches that rely upon national security frameworks and that comply with the latest requirements of corporate and legal security auditing.

## Metering

Metering is, perhaps, an unfamiliar concept for some in the media industry, but it can enable new business models for media companies and suppliers alike. New consumption models are also enabled where a media company may choose to pay for the types and amounts of functions they use. This allows them to try out new business ideas without incurring the fixed costs normally incurred in traditional broadcast architectures.

## Dematerialized characteristics

This blueprint embodies the concepts in the Joint Taskforce on Networked Media (JT-NM) Roadmap of Networked Media Open Interoperability (JT-NM Roadmap[4]) specifically with regard to building dematerialized facilities[5]. We have introduced some of these concepts above. Other characteristics below will be discussed in greater depth in the body of this paper:

- Focused on people, content and monetization[6]

---

[1] See the IETF RFC at https://tools.ietf.org/html/rfc7665
[2] See https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/ for additional information.
[3] See https://redis.io/commands/rpoplpush#pattern-reliable-queue and search for related packages.
[4] The roadmap is available at https://www.jt-nm.org/roadmap
[5] See http://www.jt-nm.org/documents/DematerialisationBackground_2017_09_05.pdf for an in-depth discussion of dematerialization
[6] We realize that a number of media companies interested in this technology are not-for-profit, and that monetizing content is not a goal for them. That said, frequently these companies want to add value to the

- Security is built-in from the beginning
- Capable of consuming and delivering SDI, SMPTE ST 2110 files and OTT-style streams
- Fast try/Fast fail - try new ideas quickly and at minimal cost. If they don't succeed, try something else!
- Format agnostic
- Frame rate agnostic
- Geographic diversity & business continuation built-in
- Takes advantage of the large pool of developers and tools for Internet technology

While the Agile Media Blueprint has yet to be demonstrated at scale, it offers significant benefits to broadcasters at a time when they are faced with serious operational challenges. And the architecture employed by the Agile Media Blueprint has been proven in some of the largest machines humans have ever built. Twitter is one example, where the mass-market social media applications conducts thousands of transactions per millisecond.

## What is the key enabling technology behind it?

The platform is the technology stack and architectural patterns behind social media, big data, Over The Top (OTT) video streaming, and billions of business transactions per day. The AMB is specifically designed to leverage the platform. Massively parallel with fast, non-blocking networking, its evolution has reached the point where it is more than capable of manipulating compressed and uncompressed media significantly faster than real time[7].

On-premises enterprise computing is being replaced by cloud computing and/or remote data centers. The systems, protocols and application development for the platform continue to be optimised around Internet applications and abstractions (e.g. layer 7 networking, Javascript), and have evolved beyond just a replication of enterprise computing off site.

A mass market of transferable software development skills exists that work with the platform. To cost effectively build its future, the media industry will need to use the platform and these people with relevant skills.

---

content they create or acquire, and they want to expose that content to their audiences efficiently wherever their viewers may be. For those readers, please consider whether you can substitute the concept of "adding value" for "monetizing content" as you read through this paper. In many cases, these media companies share a number of goals with their commercial colleagues, e.g. doing more with less, exploiting the long tail of the content they have, and providing more "channels" to individual viewers.

[7] See R I Cartwright. *An Internet of Things Architecture for Cloud-Fit Professional Workflow*. Proceedings of the SMPTE Annual Technical Conference, October 2017. (Also SMPTE Motion Imaging Journal, June 2018, to appear.)

# Use cases

The following use two cases are intended to lay out real-world scenarios and to describe how their requirements may be met by the Agile Media Blueprint.

## Use Case 1 - Master Control / Live Sport Remote / Live Studio

### Current Operation

This use case begins by describing the current operation of a media company with a linear channel that is distributed nation-wide.  Most of the time the channel transmits scheduled, pre-produced programs combined with short promotions and commercials.  The company also deploys OB vans (sometimes called Remote Trucks) to sports arenas where they are used to create a live feed.  This feed is sent back to a Master Control Room (MCR) for inclusion in the national broadcast.  During the live broadcast, the MCR facility takes additional live feeds from commentators on a stage located in a local production studio.  This studio programming is broadcast before and after the live event.  When the live event is finished, MCR again plays out pre-produced programs.  A simplified diagram of this use case, using conventional SDI and MXF technology, is shown in figure 4 below.

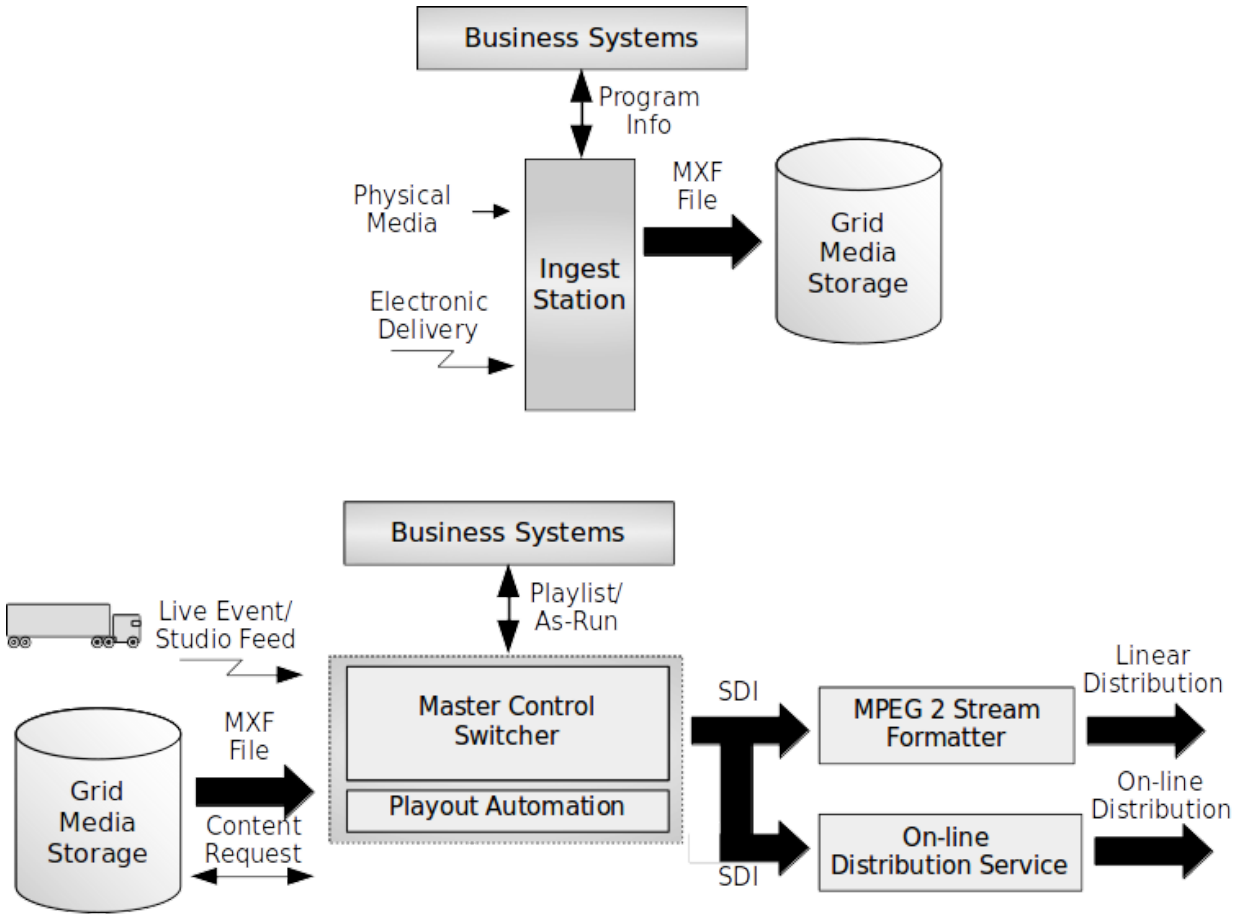



**Figure 4. Traditional SDI/File-based Master Control Facility**

Starting at the upper-left, pre-produced content is delivered to the facility either using digital video tape or a file delivery service. Tapes are brought to an “ingest station” and are loaded into a Video Tape Recorder (VTR). This VTR and a computer workstation with associated software (an ingest station) is used to transfer the tape contents to an MXF file, assign an ID to the content, and to populate a database with necessary metadata associated with the content. The ingest system interfaces to business systems, exchanging information about the content that is expected for delivery, assigned identifiers, and the status of the content.

A file delivery service (an outside vendor) provides a computer, on to which content is delivered. This computer interfaces with the ingest workstation, emulating a VTR. The operator transfers content using a workflow that is similar to that used for tape ingest.

The MXF file created by the ingest station is stored on a central grid media storage system in a proprietary internal format.

The MCR area, shown in lower portion of the figure, consists of the same grid media storage system, a master control switcher and a playout automation system. The automation system

interfaces to business systems and receives a schedule of material for playout, arranged according to time; called a playlist. The automation system interacts with the server and the MCR switcher at air time causing the appropriate content to be played, and causing the switcher to take that content and emit it on the linear output.

The output of the switcher is an SDI signal that is converted to an MPEG-2 Transport Stream. This stream is used to feed various linear distribution channels such as a transmitter or a cable system. This same feed is sent to an on-line media company for Adaptive Bit Rate (ABR) “chunking” and distribution via the Internet.

From time to time, the media company has the rights to broadcast live sporting events. Before and after the remote live event, the MCR facility takes a live feed from a local studio facility. During the live event, the MCR takes a feed from the OB van. MCR inserts commercials and promotional material using “live assisted automation”.

## Future Operation

Let us look at how this same use case is supported in a future facility implementing the Agile Media Blueprint.

The upper left corner of figure 5 is essentially identical to the current facility, with tape and electronically delivered programming going into an ingest workstation. The ingest station still exchanges information with business systems. However, there are two fundamental differences in this workflow. First, as content is ingested, it is converted to grains, and the associated identity and timing metadata is permanently bound to the content. Second, the content is stored as grains in a generic object store rather than in a proprietary file format on a specialist video server. Importantly, all interaction with the object store is through an open and publicly available Content API.

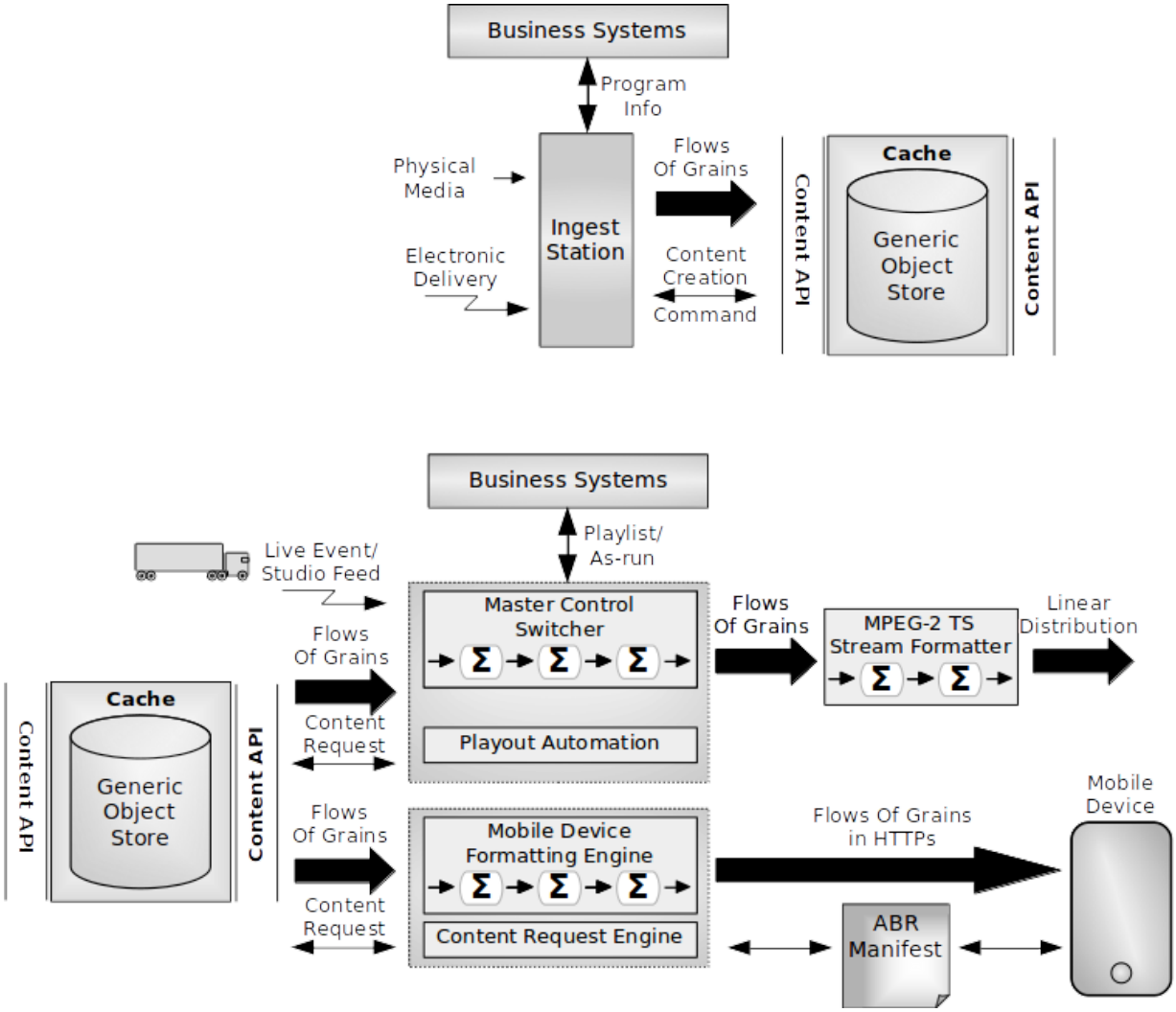



**Figure 5. Agile Media Blueprint Master Control Facility**

As in the SDI case above, the MCR facility is shown within a dotted box in the center of figure 5 above. The MCR switcher consists of a number of atomic functions such as “cross-fade” or “key” that can be chained together to achieve the same output as that obtained by the traditional SDI MCR switcher. The automation software interacts with the object store through the same open Content API that is used during the ingest step. The switcher emits flows of grains, which are sent to an MPEG-2 TS stream formatter, which also consists of a number of atomic processing functions. From there, the stream is sent to a transmitter or to other traditional linear distribution points.

Regarding the possibility of a software-only MCR switcher built upon cloud technology using chained atomic functions, a demonstration at IBC[8] showed live video being mixed through a smartphone using “soft” controls.

[8] https://twitter.com/twitter/statuses/910062784662183936

It would be possible for this linear feed to be sent to an on-line service to be chunked and distributed via the Internet. But the AMB can provide an attractive alternative. Due to the similarity of the atomic processing functions of the AMB and adaptive bit rate transcoders, it is possible to efficiently distribute the Internet feed from the same facility used for the linear feed. This provides a cost saving of bringing distribution stream processing in house. As an additional benefit, using the same facility, the media company can have a more personalized relationship with its audience. It becomes possible to stream personalized content to mobile devices and smart TVs using content that is either stored in the object store, or being streamed live. Other scenarios are possible with this infrastructure. See Use Case 2 below.

In figure 5, the live studio and all of the broadcaster's OB vans have been converted to Agile Media Blueprint technology. They emit flows of grains that are transported using HTTPS protocol to the object store. As soon as grains are written, they are available. The automation system requests these live grain flows from the cache, once again leveraging the Content API. From this point, those flows move through the workflow established by the MCR switcher and out to onward distribution points.

Critics may point out that the AMB cannot deliver the low latency required in this use case. But it is worth noting that given current technology and compression, "live" broadcasts as they exist today are far from live. Also consider that HTTPS can provide faster than realtime transport. We concede that there may be cases where the performance of technologies inside the platform may adversely impact real-time workflows. In this case, an AMB facility can sit alongside a traditional broadcast plant, providing enhanced capabilities to the broadcaster, and importantly, providing a migration path from SDI to the future.

This use case started with a traditional SDI workflow, and has shown how the same workflow may be supported using an Agile Media Blueprint design. While the move to IT changes much of the underlying technology base, those changes are abstracted away from the broadcaster, allowing us to present this use case in a way that is very familiar.

## Use Case 2 - Live Broadcast With Integrated User-Contributed Content

This use case assumes the facility described in use case 1 above. But in this use case, the media company has additionally secured rights that create new revenue streams. These opportunities are as follows:

- User-contributed content; live content from user devices is integrated into the stream created by the broadcaster
- User-contributed live event sports production; previously vetted and authorized end-users are allowed to "customize" their own live event broadcast based upon content assets made available by the media company. The public may choose to "follow" this end-user, and may also be allowed to "join" these broadcasts.

- Artificial Intelligence (AI) live event production; an AI system that is aware of a viewer's preferences creates a personalized live event, showing highlights of the viewer's favorite athlete, making available additional interstitial content featuring that athlete or his team and providing commercial opportunities to advertisers during breaks in live action at the venue.
- The OB van(s) are eliminated; live production is done in-house at the media facility.

## User-Contributed Content

Referring to the figure 6 below, in addition to cameras and microphones provided at the venue by the broadcaster, user-contributed content is available through apps written for mobile devices. Users running these apps, and with appropriate authorization, become potential sources of content to be integrated into the live broadcast. The apps on the devices produce HTTPS flows of grains with the required identity and timing headers.

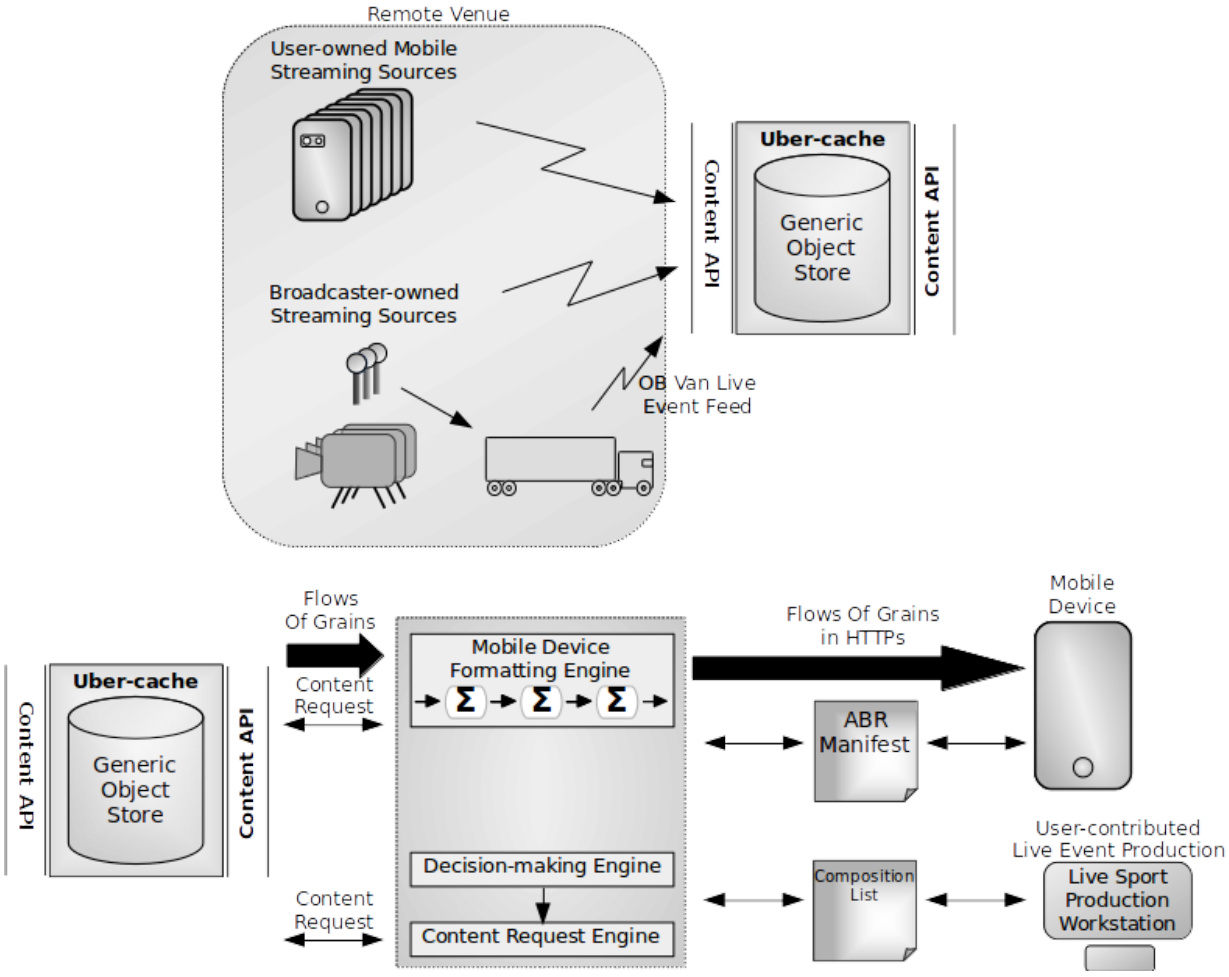



**Figure 6. User-Contributed Live-Event Content and Remote Sport Production**

## User-Contributed Live Event Sport Production

As shown in the lower right portion of figure 6 above, a “remote producer” creates a new “live event feed” using a Live Sport Production Workstation. With the proper authorization, that producer interacts with decision-making engines and content request engines to access live streams from the generic object store.

The application used by the remote producer interacts with the object store and clustered cache through the Content API, allowing him or her to select different camera sources. Manufacturers create new software tools that allow the remote producer to add graphics, perform video effects, and to do other tasks necessary in a live production.

The remote producer promotes his or her channel through social media (perhaps with the assistance of the Broadcaster), and viewers may “follow” the producer and join the live events he or she creates. The media company is free to create new business models that potentially include the remote producer, those end-users creating content at the venue, and commercial sponsors.

## AI-Created Live Event Content

An AI system running on the platform, having analyzed an end-viewer’s preferences, provides a personalized live feed focused on that viewer’s favorite athlete. The AI accesses the object store through the Content API, and interacts with the end-viewer. That end viewer has the option to give input to the AI, “liking” and “not liking” portions of the feed. As a result, the AI learns the preferences of that specific end-viewer and delivers a feed that reflects the viewer’s preferences at that moment, based upon real-time data. This scenario is shown in figure 7 below.

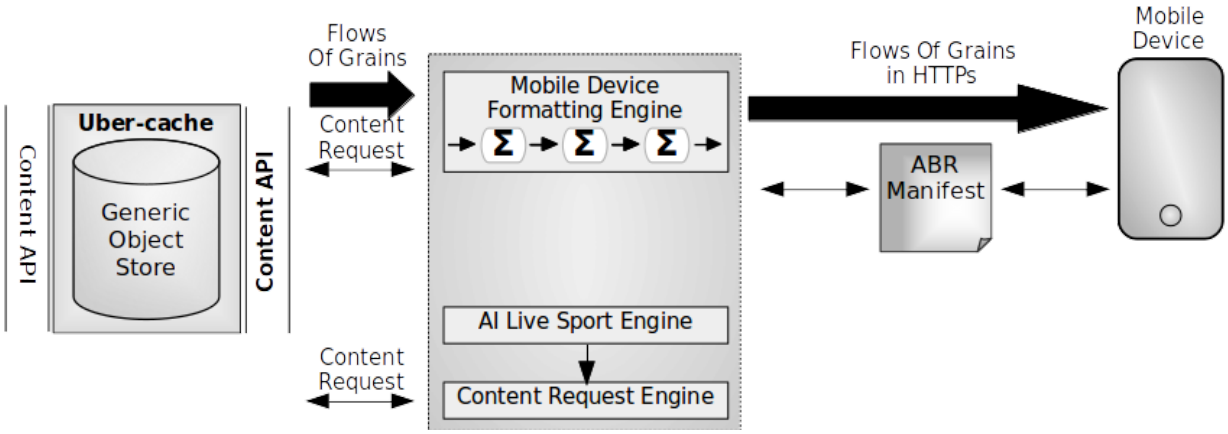



**Figure 7. AI-produced Live Sport Engine**

## Local Production of Remote Events

In this use case, given the capabilities described, it becomes feasible to consider eliminating OB vans entirely. High-bandwidth connections between venues and media facilities can allow the "backhaul" of all cameras, microphones and other sources, allowing the media company to produce the live broadcast in a fixed facility rather than in a OB truck at the venue itself. The cost savings represented by this change could be significant, as could the increase in the number of live events it is now possible to cover. This backhaul scenario is shown in the upper portion of figure 6 above.

## Use Case Conclusions

These use cases may seem unusual. The fact is that with regard to live sports, entirely different products are rapidly evolving[9] becoming ever more social-media-like. A broadcaster with a conventional SDI plant would, out of necessity, not be able to take advantage of the business opportunities presented in this use case. While we do not know if these specific use cases will evolve, applications such as PunditSports and Periscope[10] are redefining the meaning of live event video. The Agile Media Blueprint provides a way forward for media professionals who seek to provide competitive products in this application space.

## What is in it for me?

If I were to make the investment and go through the workflow changes that are required to implement the Agile Media Blueprint, it is reasonable to ask, "What is in this for me?". Consider the following:

- It enables you to put people in the best roles to create and monetize content effectively, both in traditional workflows, and in workflows that are future-facing
- It provides a path from wherever you are now in your technical operations toward a more adaptable architecture that is better able to meet future needs
- It opens up opportunities to apply Artificial Intelligence (AI) to your operation in ways that will benefit your business and make you more efficient
- When we are approached with innovative ideas and new business opportunities, it allows us to say "yes" to the business partners in our organizations rather than, "well, that will be rather difficult and costly"
- Integrate Broadcast Operations into the paradigm shift brought about by Internet Technology, rather than having it turned off at some point in the future.

## Advantages for suppliers

Suppliers will find a specific set of advantages compared to the traditional approach for the development and deployment of media technologies:

[9] See https://www.punditsports.com/ as an example of user-contributed content and live sport
[10] http://www.adweek.com/digital/periscope-now-allows-android-users-to-broadcast-live-360-video/

- It provides a common virtual facility design, allowing them to innovate on top of this shared platform (e.g. more efficient content processing, better quality images, AI creative tools)
- It allows suppliers to track the usage of products they develop so that they can measure and capitalise on their investment in a new way.
- Tracking will provide the opportunity to analyze end-user's likes and dislikes, giving the supplier important information on how to make better products
- It operates on top of a technology that is well understood by hundreds of thousands of software programmers, computer scientists and architects rather than using technology such as SDI and MXF which are well understood by only a few technologists and which are difficult to learn

# The Agile Media Blueprint In Context

It may be helpful to look at how the Agile Media Blueprint relates to a number of important initiatives and technologies in our industry.

## How does the Agile Media Blueprint relate to the JT-NM Roadmap?

The JT-NM[11] routinely updates a roadmap showing anticipated critical developments in technologies related to the deployment of IT technology in the professional media environment. The current version of the roadmap as of the publication of this paper is shown in figure 8 below.

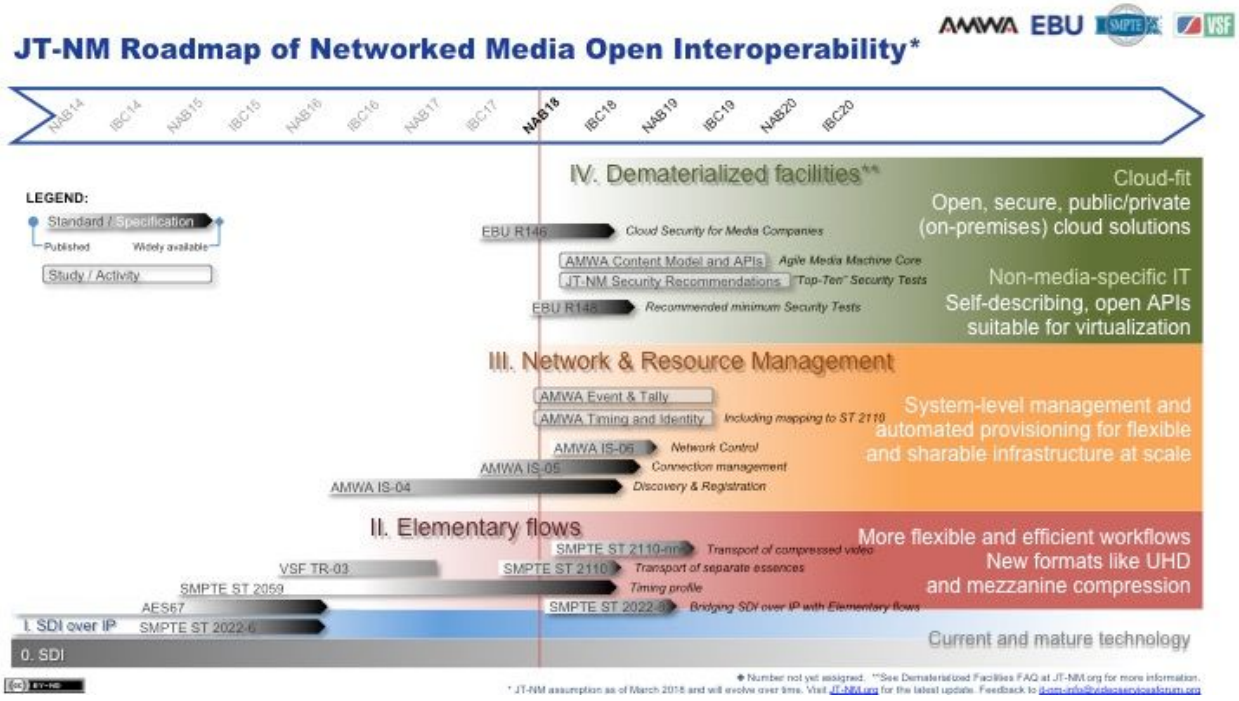



**Figure 8. JT-NM Roadmap of Networked Media Open Interoperability**

The JT-NM Roadmap contains a region near the top of the document labeled “Dematerialized facilities”. The term Dematerialized Facilities refers to a broadcast facility operating on generic IT equipment, and in some cases, housed in remote locations operated by others. More generally, it is a vision where media facilities no longer reside within the four walls of a physical

[11] JT-NM stands for the Joint Task Force on Networked Media. Additional information is available at http://www.jt-nm.org and https://tech.ebu.ch/home/groups/jtnm.html

location, but instead may be a combination of privately owned infrastructure that resides in a physical place, with connections to several different cloud-based platforms.[12]

## The Broadcast Industry Transition

As the industry transitions to IT technology, there is the opportunity to take advantage of current best practices in the IT domain. One can imagine a continuum from 1) dedicated hardware solutions to 2) software on dedicated servers, to 3) virtualized software, to 4) cloud-hosted systems, to 5) hardware-accelerated cloud platforms[13]. The Dematerialized Facilities region in the roadmap is intended to cover virtualized software, cloud-hosted systems and the hardware-accelerated cloud platforms.

The JT-NM observes that many media companies have migrated away from dedicated hardware, to a mix of software running on dedicated servers, and software-only solutions running in virtualized environments. In some cases, media companies are already making extensive use of the cloud. We recognize that media companies will choose the appropriate place for their organization on the continuum, and that the place where they sit may change over time. The JT-NM also recognizes that deploying “Non-media-specific IT” is a stepping stone along the road to “Cloud-fit” deployments based on open, secure public/private cloud infrastructure.

The dematerialization region of the JT-NM Roadmap shows this general trend and outlines a migration path along the continuum described above.

## The place of the Agile Media Blueprint on the Roadmap

If Dematerialized Facilities are a vision of the future, then the Agile Media Blueprint is a plan for how to get to that vision. It includes a description of the key components required, and how those components relate to each other. Agile Media Machines, separate from the Agile Media Blueprint, are things; concrete, representative instances of the Agile Media Blueprint.

When thinking about the place of the AMB on the roadmap, consider that it clearly consists of components and best practices from the dematerialization region, but it incorporates some of the characteristics of the Network & Resource Management layer, and easily produces and consumes content in conventional and emerging forms such as SDI, SMPTE ST 2110 flows, files in formats such as MXF and MOV, and other stream formats such as MPEG-2 TS. The specifics of how this is accomplished are described later in this paper.

[12] For more information on Dematerialized Facilities see *Information Sheet - IBC 2017, Dematerialized Faciliites* available at http://www.jt-nm.org/documents/DematerialisationBackground_2017_09_05.pdf
[13] The authors wish to acknowledge Chuck Meyer for introducing this continuum concept

## The Agile Media Blueprint and its relationship to the Cloud

While the AMB is “born” from Internet architectural experience and best practices, there is nothing about the architecture that prevents it from being deployed in a number of environments, including: running totally isolated from the Internet on bare-metal servers; on PC desktops; smartphones or tablets, or even on the diminutive RaspberryPi.[14] Of course, the AMB leverages technologies that are widely deployed in the Cloud, and many end-users are already making major media facility deployments in a cloud-based environment[15].

## How does it relate to other technologies?

The AMB is designed to be complementary to other digital and IP-based media transports. The table below describes various characteristics of: SDI, SMPTE ST 2022-6, the SMPTE ST 2110 suite of standards, SMPTE ST 2110 in combination with AMWA’s NMOS, and the AMB with HTTPS-based transport.

| | SDI | ST 2022-6 | ST 2110-* | ST2110-* + NMOS | AMB with HTTPS |
|---|---|---|---|---|---|
| **IP** | no | UDP | UDP | UDP - media, TCP - NMOS | TCP |
| **transport packing** | blanking included | bitstream with blanking | bytestream, packing reqd. | bytestream, packing reqd. | data blobs - packing opt. |
| **multiple streams** | interleaved | interleaved | elementary | elementary | elementary |
| **encoded** | no | no | no (not yet) | potential | yes or no |
| **grain-based** | no | no | no | with headers | yes |
| **data rate** | real time | real time | real time | real time, faster with headers | non-real-time / faster or slower |
| **concurrency & direction** | sequential push, clocked | sequential push, clocked | sequential push, clocked | sequential push, clock option | concurrent push & pull + backpressure |
| **format description** | signal | signal | via SDP | discovery & SDP | cable & headers |

[14] For more information on running serverless in an off-line environment see https://github.com/dherault/serverless-offline
[15] See, https://www.tvtechnology.com/news/discovery-finishes-move-to-cloud-for-us-network-playout

| | | | | | |
|---|---|---|---|---|---|
| **reliability** | redundant path | ST 2022-7 & FEC, dualed | ST 2022-7 & FEC, dualed | ST 2022-7 & FEC, dualed | clustering, N+1, TCP |
| **scaling** | SDI router | multicast | multicast | multicast / caching | caching |
| **secure path** | physical separation | SRTP or physical | SRTP or physical | SRTP or physical | HTTP over TLS, QUIC |
| **structure** | predefined by spec. | predefined by spec. | proprietary orchestration | discovery & grouping | DNS & logical cables |

The categories in the table are:

- **IP** - Does the media travel over IP-based transport and at what layer in the stack?
- **transport packing** - Does the transport include blanking intervals? Is it a bitstream or a bytestream? Does it require specific packing for transport or can packing be optimised per workflow?
- **multiple streams** - Are related component streams (video, audio, data) interleaved or carried as separate network flows?
- **encoded** - can the essence (pictures, sound) be compressed for carriage? For example, NMOS has been combined with RFC 6184 to do this for H.264 streams.
- **grain-based** - Are the essence elements uniquely identified in the stream, allowing carriage of identity and end-to-end timing through a workflow? Note: NMOS has been demonstrated with RTP header extensions to achieve this.
- **data rate** - Are streams transported in real-time, or transported faster or slower than real time?
- **concurrency & direction** - Can a single stream be transported concurrently using overlapping network flows? Does the sender push streams over the network timed to a clock, or does the receiver pull the streams? Is the rate of transmission of the producer regulated by *backpressure* from the consumer(s)?
- **format description** - How does a decoder determine the format of the stream?
- **reliability** - How is stream reliability achieved?
- **scaling** - How can the transport be scaled, allowing one sender to send the same stream to multiple receivers?
- **secure** - How can the stream be secured, either by physical separation from other network traffic or by encryption on shared infrastructure?
- **structure** - How are the structure and synchronization relationships between components of the same content discovered and expressed?

# Introducing the Agile Media Blueprint

The Agile Media Blueprint starts with people, in teams, within organisations and across organisations. As shown in figure 9 below, it begins here, not only because people are a key component in our craft-oriented industry, but because by starting here, one can address security at the outset. (See more about this topic in the Security section below.) Administrators can assign roles to people and grant them permissions based upon those roles. People in a different organizations may be given specific rights in your organization based upon business needs. For example, personnel from two remote production companies using two different OB vans may be given roles that allow them to collaborate for a one-time event. This approach allows people to securely collaborate using a shared Content API. The Content API can be implemented to interconnect cameras, microphones, speakers, multi-viewers and control surfaces.

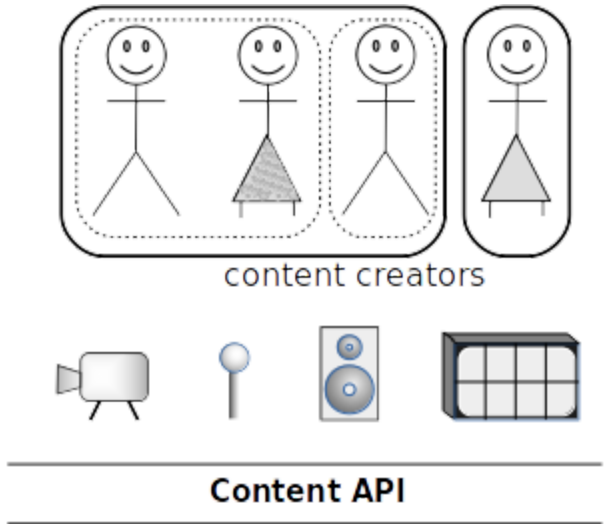



**Figure 9. People And Their Roles Are A Key Part Of The Agile Media Blueprint**

While the Agile Media Blueprint can be deployed to meet professional media production needs, it also may be used to efficiently produce new types of content that require direct interaction with end viewers. As illustrated in figure 10 below, the Agile Media Blueprint targets social consumers, through broadcast media, OTT and mobile.

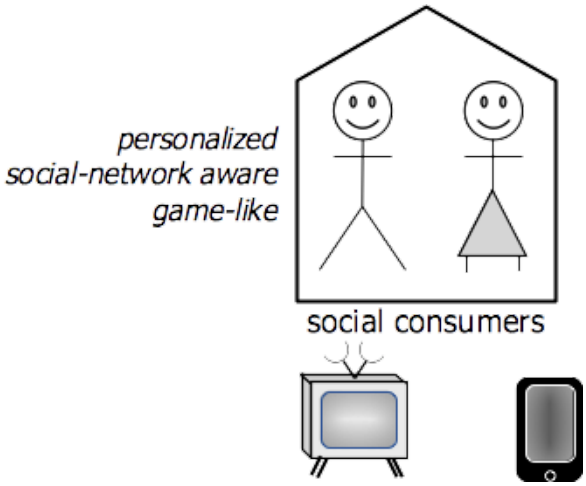



**Figure 10. Consumers Interact With Bespoke Versions of Content**

Conventional broadcast facilities could never provide individualized content to tens of thousands (millions?) of viewers - it is an anathema to the fundamental concept of “broadcasting”. But to deliver whatever a viewer wants to see when they want to see it means that it is absolutely critical that AMB facilities scale seamlessly. In many cases, in an “individualised content” scenario, content is cached and streamed to the viewer as a one-off event. Many broadcasters are already paying for bandwidth at a cost scaled per viewer, without benefiting from a closer, bidirectional relationship.

As figure 11 below shows, at the core of the Agile Media Blueprint is a grain-based Content API for element-by-element access to the media, backed by best-of-breed cloud-fit technology, including clustered RAM caches, AI and object stores. Microservices make new media elements to order as they are required; no more wasting time and resources by creating media elements “just in case” they are needed.

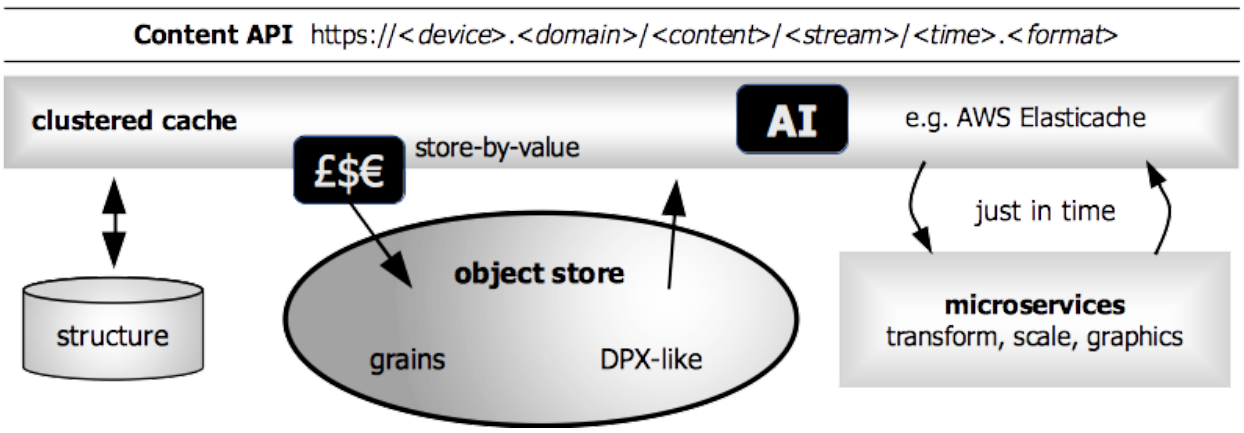



**Figure 11. A Common Content API Provides A Uniform, Open Interface To Essence**

Media transport in the Agile Media Blueprint is between Content APIs, with bidirectional links

operating in parallel, faster than, slower than, or at real time. As shown in figure 12 below, a time-to-bytes translation interface (bytes-to-time component) allows content to be read and written to common signal and file formats, and as described above, allows for compositions to be made on-the-fly.

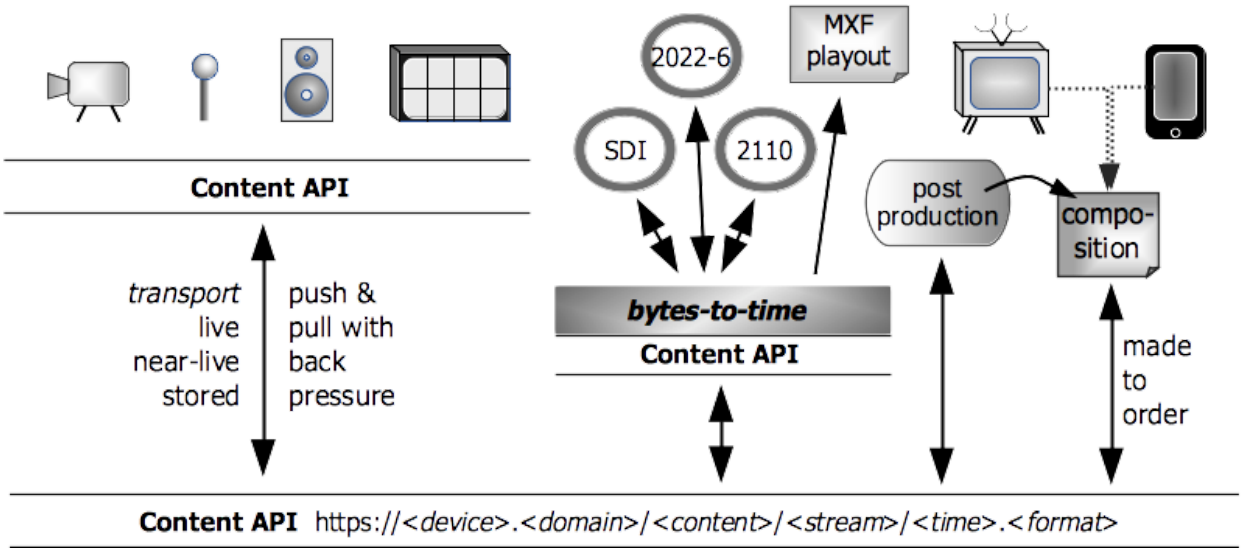



**Figure 12. Bytes-to-time Conversion Allows Creation/Consumption Of Existing Formats**

Libraries of files and streams can be migrated into an implementation of the Agile Media Blueprint, either just-in-time, or according to a schedule, using a bytes-to-time unwrap component, supported by an index database (see figure 13 below). This allows media factories to easily migrate from a file-based infrastructure to an API-based infrastructure.

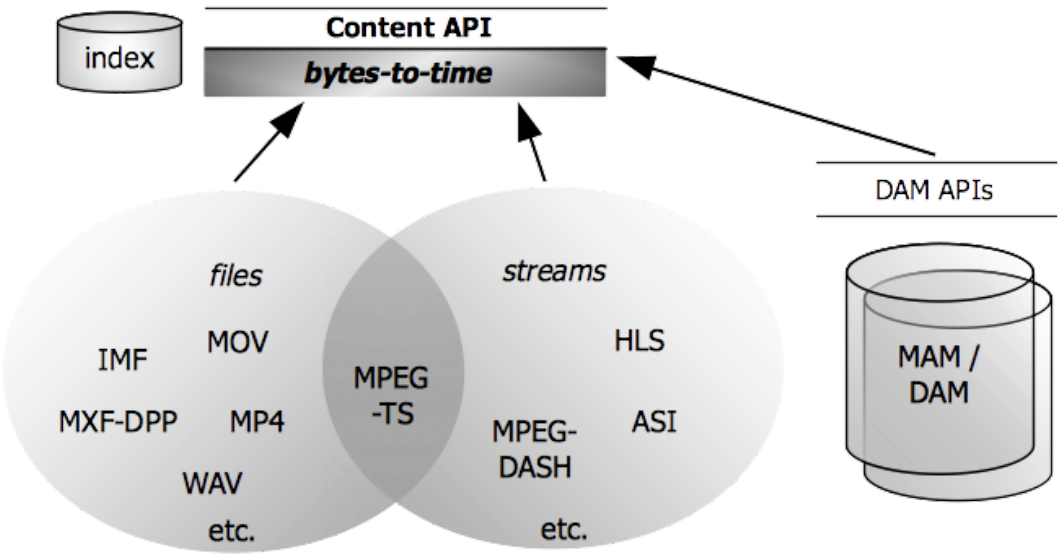



**Figure 13. Bytes-to-time conversion and an Index Facilitates The Creation Of Common Media Files and Stream Formats**

# Security

IT already has excellent mechanisms for managing users, groups, organisations, roles etc. These mechanisms support the safe sharing of resources between people and across organisations. IT also provides a set of highly scalable and securable protocols for data transport that have survived the evolution of the Internet. Until now, the media industry has not been able to leverage these mechanisms, largely because media has existed outside of the IT world as opaque serial streams (SDI) or in non-IT-friendly file formats (MXF).

Because our industry evolved separately from the IT ecosystem, security was frequently “bolted on” to our media applications. And because we built media systems with bespoke products from a multitude of vendors, we seldom got a joined-up approach to security. This left us in the unenviable position of attempting to create a hardened, cutting-edge security environment out of a patchwork quilt. This is a difficult, and some would say, impossible task.

To be fair, the early Internet evolved without a lot of attention paid to security, and in a largely collegiate environment (that said, the United States military initiated development of the Internet with very specific security requirements in mind). But two things have vaulted IT technology ahead of bespoke media platforms.

- Malicious people appeared pretty early in the evolution of the Internet. Bad guys, or perhaps just curious college students with a lot of time on their hands, started picking away at any server they could reach. This means that IT departments have been dealing with security threats for a long time, while Broadcasters remained insulated from the fray, at least until relatively recently.
- Because platforms such as Amazon and Azure provide a joined-up approach to security, developers on these platforms benefit from a common framework across a broad range of applications. This is a relatively recent development, springing from the fact that many cloud platforms offer a homogeneous approach to security that is then employed by application developers writing for those platforms. Over time, this change has allowed IT security professionals to develop a set of best practices, technologies, and even security-specific jobs around this topic.

While broadcasters and media companies might wish to “air-gap” themselves from the Internet and the security issues it raises, this is just not practical. Connections, and security liabilities, are inevitable for any company that wishes to actively engage in the media industry in the modern age.

Increasingly, government regulators and investors are requiring that media companies perform security audits. Suppliers may soon find themselves needing to provide certifications stating that their products conform to certain best practices, standards and laws. This trend is

accelerating, especially as more countries identify broadcasting and media as critical national infrastructure.

The number of security issues media companies need to track is not static. For example Web 3.0 and the increasing adoption of Blockchain technologies[16] lead to decentralization, putting users in control of their own data. This means that we face an ever-expanding task in evaluating security impacts and deploying these technologies. Would we do better to adopt the platform and the best practices it offers?

Finally, it is important to consider security from a developer's point of view. In the end, it is the developer that is charged with finding a way to produce applications efficiently while at the same time meeting increasingly stringent security requirements.

## A developer's view of the security audit process

A software developer works to deliver a high-quality product as fast as possible. To achieve this, that developer needs to work in an environment with the lowest-possible level of friction. One aspect of a low-friction environment is that the software he or she writes passes security audits prior to delivery. Typically, the developer must complete a checklist that is provided by their company's information security team. This includes items such as what ports need to be open, what protocols are in use, and what operating system patches need to be applied. This checklist is used to combat an ever-growing list of known attacks that the software must be hardened against[17].

Developers are finding it ever more difficult to make requests that their products have non-standard arrangements for deployment outside of the NoOps workflow[18]. The easiest way for a developer to produce software that passes security audits is to apply best practices when deploying to the platform.

Characteristics of this include:

- Using a cloud platform, where every element starts as completely secure and requires additional configuration to make it less secure
- Develop using serverless technologies, such as lambda functions, shared queues and notification services, independent from any operating system or installed product

[16] Note that we are not advocating the use Blockchain for media but using the large body of debate in the IT industry on this and similar topics as an example of the fast-moving context of security.
[17] As an example of secure coding rules and recommendations for Java, see: https://wiki.sei.cmu.edu/confluence/display/java/SEI+CERT+Oracle+Coding+Standard+for+Java
[18] NoOps is a cloud-fit development of DevOps, and provides developers with a continuous delivery pipeline of automated tools for managing every aspect of software development, testing and deployment, delivering to an unconstrained pool of resources. See http://www.computerweekly.com/opinion/When-DevOps-isnt-enough-try-NoOps

- Using high-level languages that execute inside virtual machines and/or through API's provided and maintained by trusted 3rd parties, eliminating the likelihood of security issues due to buffer overruns etc.

## IT Engineering Mindset

The AMB builds just enough professional media capabilities into an Internet technology framework. This means that the approach to failover, resilience, deployment, scale and monitoring are those from an IT mindset, not necessarily those familiar to a broadcast engineer. Some examples of this are:

- Using the bidirectional power of the network. Media is pulled as fast as possible with clocks at clients, possibly over multiple paths, rather than pushed in real time and synchronized to house clocks.
- Resilience through clustering and caching. Clustered caches across cloud availability zones provide high-availability to read and write immutable objects[19] with load balancing for scale, rather than dual path networking aka SMPTE 2022-7 and multicast.
- Over provision and monitor. Capacity is over provisioned using truly non-blocking switches and the rate of media movement is monitored, allowing hot-spots to be preempted.

# Business Summary

Moving to the Agile Media Blueprint presents a number of challenges to us from a business perspective, including changes to mindset and a fundamental change in the underpinning of the technology basis of our business. But as the discussion above has pointed out, there are major benefits to adopting this approach going forward. That said, two key business-related questions remain. First, how can I make money with this new technological approach, and second, is it real?

## How can I make money with it?

Media companies can make money with the AMB by being able to:

- leverage the investments being made in the platform
- by using an architecture that allows them to rapidly scale up and out, or down and in depending upon demand
- by making use of functions and capabilities on an as-needed basis
- by deploying an infrastructure that has security built in from the beginning

[19] The JT-NM Reference Architecture (RA) defines an Immutable Object as a Resource whose state cannot be altered after it is created. The RA may be downloaded at http://jt.nm.org/

- by creating new products that provide interaction with end viewers, resulting in personalized content delivered to them in ways that could never be done using conventional media platforms.

From the outset, the platform has provided monetization opportunities for IT vendors. Suppliers of media applications can take advantage of this fact. Metering provides a pathway to "de-risking" and sharing risk in media partnerships.

Wikipedia defines Serverless computing, in part, as, "a cloud computing execution model in which the cloud provider dynamically managed the allocation of machine resources." "Pricing is based on the actual amount of resources consumed by an application…"[20]. The AMB leverages many characteristics of serverless architecture. For suppliers, this means applications that are built to the Agile Media Blueprint are metered, and that new business arrangements charge for capabilities provided by the software, with use-based pricing. Media companies may be incited to look at this pricing model because it allows them to buy what they need and pay for what they use rather than having to pay a certain fixed amount in order to stand up a media-oriented service. This could be especially important where the user has an opportunity that they would like to explore, but they are unsure whether the venture will succeed.

## Is it real?

Yes and no. The technology needed to build the AMB exists in some form or another. Many large media organizations are already using cloud infrastructure to make and deliver media at scale (BBC Media Factory, Discovery). Organizing the components so that the AMB can be delivered as described is work in progress, as is measuring its performance and calculating the relative business value of particular use cases in a revenue-based model.

The key component is the Content API, which exists in draft form and will be brought forward for comment shortly. Tests have been run on the API to confirm that HD material can be transported over 10Gbs+ networks significantly faster than real time using HTTPS. Beyond the API, the following list is a summary of the technology that exists, either finished, in prototype form, in need of repackaging, or as research projects:

- Existing technology: Identity management for people and organisations in the cloud (e.g. AWS IAM, Azure Active Directory), managed clustered caches (e.g. AWS Elasticache, Azure Redis), object stores (e.g. AWS S3, Azure Blob Storage), computer services for hosting micro- and nano-services, adaptive bitrate streams, IT configuration and monitoring.
- Prototyped technology: The Content API. Connecting speakers, microphones, cameras and displays via a content API (e.g. naudiodon[21] - runs on Raspberry Pis). Bytes-to-time

[20] From https://en.wikipedia.org/wiki/Serverless_computing accessed February 7, 2018

[21] https://github.com/Streampunk/naudiodon

API for exposing MXF files and MPEG-TS via the Content API (e.g. byzantime[22]). Object-based rendering of content (e.g. BBC Video Context library[23]). Connections to SDI, 2022-6 and 2110.

- In need of repackaging: Monolithic applications for transcoding, picture scaling, graphics, etc., such as FFMPEG, Caspar CG, software-only versions of live vision mixing - to be reworked as software-only functions running as microservices manipulating grains. The efficient support of broadcast colour spaces (YCbCr 10-bit and HDR) optimised for appropriate hardware (CPU, GPU, TensorFlow) needs further investigation.
- Research projects: Language for describing media compositions (BBC UMCP[24]), gateways to, and the application of, AI for professional media production (e.g. AI in Adobe's creative tools[25]), store-by-value calculations.

The purpose of this white paper is to introduce the Agile Media Blueprint and the context within which it exists. As next steps, the Content API presented here will be introduced into the AMWA technical process for discussion and further development/improvement. As mentioned above, it is hoped that projects such as the BBC's project on object-based media composition could be folded into this work. Lastly, we are at a point where a deployment of these concepts in a media company is not only possible, but it would accelerate the work presented here significantly.

Readers should note that a number of media companies are already deploying facilities in the cloud, although none that the authors are aware of are using this grain-based approach with an open API. This indicates that the time is right for moving forward with this work in an open and collaborative environment such as the Advanced Media Workflow Association.

[22] https://github.com/Streampunk/byzantime
[23] https://www.bbc.co.uk/rd/blog/2016-04-videocontext
[24] http://www.bbc.co.uk/rd/blog/2016-09-object-based-composition
[25] http://www.bbc.co.uk/news/av/technology-41768758/ai-takes-centre-stage-in-new-adobe-tools

# Technical Discussion

Elements of the Agile Media Blueprint have been in development for many years, and are based on deep technical understanding of both advanced topics in computer science and digital media processing. The technical discussion that follows describes the core enabling technologies of the AMB. This includes the importance of moving to API-driven workflows, whereby the power of names must be exploited in order to to scale within the platform. This is followed by elaboration of the design of some of the AMB's components and discussion of the technical challenges associated with building them.

## Core enabling technologies

The sections that follow highlight core enabling technologies employed in the Agile Media Blueprint.

### The Content Application Programming Interface (Content API)

The most significant piece in the design of the AMB is the Content API - including a RESTful mapping to HTTPS and other protocols.  The API allows access to read, write and make grains on-the-fly. Any device, from a clip-on microphone to a central sports production facility can expose access to read or write content through a DNS name such as:

https://<device>.<domain>/<content>/<stream>/<time>.<format>

The path parts are:

- <device> - a local name for the device, possibly an alias
- <domain> - a global or .local domain name
- <content> - sub-resources that represent complete items of content, with optional extra services that help find the content, such as "search.html" or "content.json"
- <stream> - sub-resource that represents component parts of the content item, such as streams or graphic files, and additional resources that can be used to find, synchronize and process the components, such as "cable.json" or "description.html"
- <time> - for streams, access to grains or grain ranges identified by PTP timestamps, with relative PTP timings used as the ubiquitous timing model of the API. Other resources at this level are used to describe the stream, e.g. "technical_data.json"
- <format> - a means of describing the format that is being requested, which may include size, scale, codec, wrapping etc.. The format may be described further through headers, query parameters or in the body of a request.

Example 1: Workflow using the Content API:

Imagine you are looking for a particular episode of *The Blimpsons*, a fictional cartoon family. This cartoon series is produced by *ROCKS*, a large media company:

1. Assuming that you do not know in advance exactly what you are looking for, you might use the Content API to find the episode as follows:

   https://entertainment.ROCKS.com/search.html

2. A search page opens and you conduct a search for your desired episode *where Mart takes candy from a store and puts ants in it*.
3. The search returns a description page (description.html), and technical information about what is in the episode (cable.json). From there, you know that the video is in *format X*, English audio is available in *format Y*, and that captions are also available.
4. You execute a command using the Content API to retrieve the required essence flows, as grains, from the Object store using the URL below:

https://entertainment.ROCKS.com/blimpsons_s3_e42/audio_2/1521468933:171000000.wav

Example 2: GETting and PUTting content

Given the following URL:

https://sportprod.zbc.com/tennis_180112/video/0-1000.mp4/480p30

The following may be determined - this represents the first 1000 video grains of ZBC's high-definition (HD at 1080p60) tennis show from the 12th January 2018:

- When retrieving content via GET requests, if the content is not yet available in standard definition (SD at 420p30) at the moment of the first request then new chunks are transcoded just-in-time
- If contributing via PUT requests, the content is being uploaded in chunks of 1000 frames at a time in SD and will need upconverting to HD before use in the show

Whether content is being written or read, the primary use of the API is to transport media between end points (e.g. a camera and a monitor). The Content API facilitates an easy mapping of flows of grains to the HTTPS protocol. Since all connections are bidirectional, the rate at which material is moved can be throttled by back-pressure. (Back-pressure is a technique where the receiver may dynamically throttle a sender so as to avoid being fed data at a rate which exceeds its ability to process the incoming stream.) The resulting transfer speed is a consequence of evaluating a cost-function that balances available bandwidth against desired quality. Other protocols, such as RTP, web sockets and HTTP backed by QUIC may provide an alternative.

The AMB's Content API is designed to be complementary to, and compatible with, the JT-NM Reference Architecture and the AMWA NMOS content model.

The API may be used to provide, and even distribute, chunks of media and manifests required for Adaptive Bit Rate distribution. Moreover, in combination with a composition description (a new form of Advanced Authoring Format[26] (AAF) that is easy to read and process in small chunks) content can be rendered to suit who is viewing it. While traditional outputs from the post-production suite are still possible, this API would permit new kinds of outputs that can be dynamically rendered on a per-viewer basis.

## The power of names

DNS-

Names on the Internet are fundamental.  They are the key that enables scaling, high availability, maintenance reconfiguration, service migration and minimising global latency. Once you give a name to a thing, anyone can make reference to that thing unambiguously.

A stable name (sometimes called an immutable identity in computer science terminology) remains the same throughout the lifetime of an object, even as that object changes location and takes new forms. As an example, as individuals, we change our clothes, grow older and move to different towns and cities, but our name remains the same.

The hierarchical Domain Name System is a globally distributed decentralized name service. It allows you to operate a world-wide set of servers, or to run your own DNS server within your facility. Servers may be set up to either hand out names to computers, or to discover the name of computers as they connect to the network (via DHCP). DNS may be used to do load sharing by directing queries to a group of servers using various load-balancing algorithms. DNS provides a seamless way to redirect traffic from a failed facility to resources located in a different geographic location.

The power of Domain Name System (DNS) identifiers is that services and devices can change their location and even the physical hardware they are running on, transparently to the user. Services can be scaled up and out so that a name that once represented a single computer now represents a load balanced, highly-available cluster. In different parts of the world, a name may resolve to the most local content distribution network, reducing latency, retrieval time, and increasing reliability. None of this is possible if one uses identifiers that are tightly bound to specific pieces of hardware, or network locations such as MAC addresses or IP addresses.

DNS, generally speaking, is an underused resource in the professional media environment. Making use of DNS in the platform allows us to employ any number of techniques and best practices for business continuation.  Furthermore, the DNS system is continuously monitored

[26] https://en.wikipedia.org/wiki/Advanced_Authoring_Format

and maintained by some of the world's best security experts, and employs cutting-edge security technologies on a global basis.

URLs -

Unique Resource Locators (URLs) add a hierarchical path to a DNS identifier allowing us to assign an immutable identity to a resource. In combination with the HTTP, resources can be organised, published, referenced, downloaded and uploaded. Resources may exist on disk and be streamed to you by reading a file, or they may be rendered "just-in-time". In this way, resources can be made to order, often by having a main web page calling a set of other microservices. And HTTP allow resources to move, update, require authorization, fail, have multiple formats and so on.

How might URLs be employed for media? Names may be used to represent:

- people, teams, organisations
- devices - cameras, microphones, speakers, monitors, control surfaces - both with an actual name and an alias for their current use, e.g. "Camera 17" aliased to "Studio A Camera 3"
- streams - content flowing from a source to destination, whether it is being pulled or pushed
- content - collections of streams and stored content with associated data
- compositions - recipes for assembling new items of content from existing ones, just-in-time or just-in-case
- every grain - frame, audio sample, and event

## Virtualization

The mapping of several logical computer systems to one of more physical computing system(s), called virtualization, has revolutionized the way in which IT systems are deployed and managed. An application can be installed onto a virtual machine, along with just enough of operating system in an isolated environment. Multiple copies of the same virtual machine may be deployed on demand, allowing optimization of physical computing resources and providing on-demand scaling.

Every component in a system that implements the AMB is expected to be virtualized, and special consideration must be given to virtual systems when processing digital media streams:

- A virtual machine is by its very nature non-real-time. The mapping from a logical machine to a physical machine is achieved by providing time sliced access of the physical resources to the logical virtual machine.
- Many modern software applications are written to take advantage of processing across multiple computing cores. The operating system provides a preemptive multitasking

service, giving developers access to concurrent processing capability that is achieved by intermittently interrupting their programs.
- Modern software is written in languages that greatly simplify memory management by providing a garbage collector. This procedure slows or stops processing from time-to-time in order to clean up memory. This frees the developer from needing to worry about memory-related housekeeping.

In a virtual environment, it is not possible, to use many popular programming languages (e.g. Java, Python, Javascript) - or even lower-level languages such as C or C++ - to write software algorithms to deal with precision timing at a micro-second level. Achieving precise micro-second inter-packet arrival time is not a matter of fixing the software - it simply is not achievable given the number of levels of abstraction between the application and the hardware. Also, stratum-1 NTP and PTP clocks are not generally available in public cloud environments.

Non-real-time does not imply that the systems are too slow to process high volumes of media data faster than real time. Alternative solutions for high bitrate media processing need to be found that are a better fit for non-real-time virtual environments. The AMB and its ubiquitous timing model based on timestamps and not precise clocks, is one way to address this issue.

## Caching

In the illustrations of the AMB above, a clustered cache was shown between the Content API and the object store. The clustered cache plays a critical role in making the platform usable for media applications. The cache's job is to ensure that content is available within acceptable performance tolerances. Let start by looking at the two critical qualities of this architectural component; synchronization and scale.

Synchronization -

Video and audio are presented on time and in sync through the use of precision timestamps. Humans are extremely sensitive to "supernatural" stimuli; things that would never occur in nature. Sound arriving at the viewer's ears prior to the visual stimulus is an example of this. Media is different from most other IT-related content in that it frequently consists of multiple streams of data, which must be presented to the viewer with the same synchronization as these streams would have been captured at the origin. In addition to being sensitive to synchronization between streams, we are also sensitive to dropouts and pauses.

Scale -

Services and streams cannot slow down just because more than the anticipated number of users start to access them. Loads can be relatively evenly distributed across many geographic areas, and across many different pieces of content. But what about those few, but critically important "collective viewing experiences" like a Royal Wedding, for example. Or what if a

particular piece of content “goes viral” resulting in millions of requests for the content over a very short period of time?

A massive amount of investment has gone into developing caching layers and technology for the Internet, with an ever growing number of features. For popular websites, large distributed RAM caches - up to 9Tb in size - become the business logic hub for serving requests, mediating microservices and a gateway for applying artificial intelligence.

Clustered caches can serve thousands of requests per millisecond, with response times per request in the few milliseconds. The cache performance may be optimized by using algorithms that assist with:

- determining which content should be persisted to a long term store and what format it should be persisted in (a store-by-value function)
- finding when content should be read into the cache, e.g. a streaming operation has been detected
- decoupling live and near-live ingest and operations from storage capacity and performance

For caching to work, data must use immutable identifiers. If data with the same identifier is read from two different sources, the data itself must be identical. A benefit of this is that it allows the same data to be cached in multiple places, for example in London and Los Angeles, so that overall response times can be optimised.

# Agile Media Blueprint Components

In this section key elements and components of the AMB are covered in more depth. While reading through this section, it may be beneficial to refer to  the Agile Media Blueprint diagram provided in the Appendix.

## Grain representation and storage

At the heart of the AMB is the ability to consistently transport and persist media streams. Implementations will want to take advantage of object storage systems common in cloud platforms, such as AWS S3 or Google Storage. On premise storage solutions may also be used, such as Quantum StorNext, or a hybrid approach, such as IBM Spectrum Storage.

### Self-identifying object names and self-describing headers

As with the rest of the blueprint, the persistent storage by object is grain-based, with each grain being self-identifying - I know who you are - and self-describing - I know what you do. A typical object store has large collections (buckets) of objects that are a flat hierarchy, requiring the use long-form object names to ensure object uniqueness. To assist with object management over

large data sets, most object stores allow a user to add their own textual data about the object (metadata) and this is kept alongside to the object itself. For a grain-as-object, its long name includes the <*content_id*>, the <*stream_id*> and the grain's <*timestamp*>, with additional headers to complete its description.

| *<content_id>_<stream_id>_1293544460:080000000.raw* |
| --- |
| AMB-PTPOrigin: 1293544460:080000000<br>AMB-PTPSync: 1293544460:080000000<br>AMB-Timecode: 10:11:12:24<br>AMB-FlowID: 4223aa8d-9e3f-4a08-b0ba-863f26268b6f<br>AMB-SourceID: 26bb72a1-0112-495d-81ab-f5160ca69015<br>AMB-Packing: YUYP<br>AMB-GrainDuration: 1/25<br>Content-Type: video/raw; sampling=YCbCr-4:2:2; width=1920; height=1080; depth=10; colorimetry=BT709-2; interlace=1<br>Content-Length: 5184000 |

The name of the grain (top box) includes its <*timestamp*> that situates each grain in media time relative to other grains in the same stream. As used throughout the ubiquitous timing model of the Content API, the value is a PTP timestamp. The headers (bottom box) that are stored as object-store metadata with the grain are transported with the grain as it moves over the Content API.

## Hierarchical storage layout option

As each grain is self-identifying, it can be stored independently in an entirely flat hierarchy. However, operations on an item of content or on a flow can benefit from the hierarchical naming of grains on storage, following a pattern similar to the Content API structure. The management interfaces of object stores often provide hierarchical views of objects as if they were stored in the folder structures of file systems. This is based on inserting forward slash (/) characters into long names, replacing the underscore characters in the previous example as follows:

*<content_id>/<stream_id>/1293544460:080000000.raw*

Follow the dots and the object storage layout allows the object store's own HTTP/S API to be used directly as a Content API endpoint. This is illustrated for a single item of content in figure 14 below.

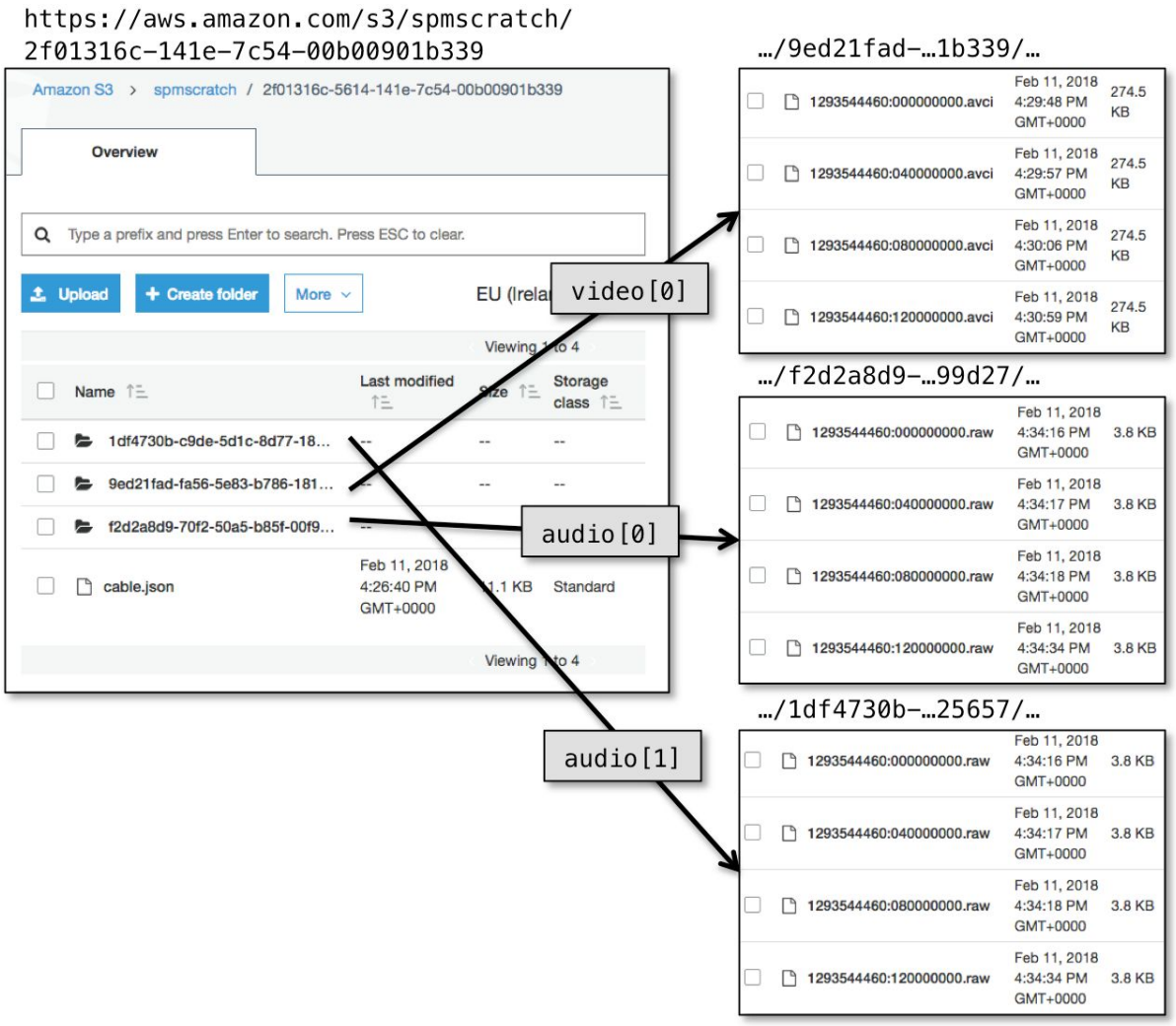



**Figure 14. Persisting Grains on Object Storage as per the Content API**

In the example above:

- A simulated folder has been created in an AWS S3 bucket called *spmscratch*. The UUID for the content item (2f0136c-...1b339) has been used as the name of the folder.
- The folder contains a copy of the content's logical cable description as a JSON object *cable.json*. This is an object that describes how flows of grains are combined into the synchronized streams of the content item, including the technical data required to decode each flow. See the example in the next sub-section.
- Each flow is stored in a sub-folder identified by its flow ID. The sub-folder contains an object for each grain, identified by its PTP timestamp. As illustrated in the table above, each grain has headers that further describe its origin timestamps, its flow and source ID, optional timecode information and a MIME type.

A benefit of the hierarchical storage view provided by the object-store API is that operations can be carried out on complete content items.  For example, using this view, it is possible to create a zip file archive or manually delete a complete content item.

In practice, the most optimal use of the object store may not be one grain per object. Different storage layouts may be optimised to suit specific media workflows.

## Logical cable

Logical cables bind items of content together by describing synchronization relationships, formats and relative flow directions (e.g. talkback) for associated streams. The name *cable* has been chosen because, from a workflow perspective, it is as if streams are travelling down a number of electrical wires all contained within one cable from one Content API instance to another.

The JSON snippets in figure 15 below are examples of the kind of information represented by a *cable.json* file. Note the use of SMPTE-style descriptors to represent technical data description, aligning the AMB architecture with pre-existing professional media format descriptions.

```
{
  "id" : "2f01316c-5614-141e-7c54-00b00901b339",
  "backPressure" : "video[0]",
  "video" : [
  {
    "flowID" : "9ed21fad-...85183",
    "sourceID" : "aa3444a1-...f88d6d",
    "start" : "1293544460:000000000",
    "name" : "video_19"
    "description" : {
      "ObjectClass" : "MPEGVideoDescriptor",
      "StoredHeight" : 544,
      "ComponentDepth" : 8,
      "StoredWidth" : 1920,
      "ActiveFormatDescriptor" : 0,
      "HorizontalSubsampling" : 2,
      "DisplayHeight" : 540,
      "DisplayWidth" : 1920,
      "DisplayYOffset" : 0,
      "VerticalSubsampling" : 1,
      "FrameLayout" : "SeparateFields",
       "SampleRate" : [ 25, 1 ],
       "ImageAspectRatio" : [ 16, 9 ],
       "VideoLineMap" : [ 21, 584 ],
       "PictureCompression" :
          "H264MPEG4AVCHigh10Intra...108050iCoding",
        "SampledWidth" : 1920,
        "SampledHeight" : 544
      },
    } ],
```

```
  "audio" : [
  {
    "flowID" : "f2d2a8d9-...c99d27",
    "sourceID" : "993795d4-...cf5818",
    "start" : "1293544460:000000000",
    "name" : "audio_1",
    "description" : {
      "ObjectClass" : "AES3PCMDescriptor",
      "QuantizationBits" : 16,
      "Locked" : true,
      "AudioReferenceLevel" : 0,
      "ChannelCount" : 1,
      "AverageBytesPerSecond" : 96000,
      "BlockAlign" : 2,
      "FixedChannelStatusData" : [ 133 ],
      "ChannelStatusMode" : [
            "ChannelStatusMode_Minimum" ],
      "AudioSampleRate" : [ 48000, 1 ],
      "SampleRate" : [  25, 1 ]
    },
  },
  … more audio tracks … ],
  … optional data tracks ....
}
```

**Figure 15.  Example of a Logical Cable Description**

## Clustered caches

The importance of caching has already been explained, but how do you achieve high availability, avoid single points of failure, and scale the capacity of the Agile Media Blueprint to meet demand? The answer is to deploy a clustered cache. Clustering allows cached data to be replicated across multiple cluster node instances, avoiding single points of failure. Clusters can be made highly-available through distribution across availability zones, where the different zones each contain a collection of similar hardware housed in a separate building and with independent power and cooling services.

Clustering techniques developed from best practice in database management and scale, include:

- *sharding* - partitioning data sets into roughly equal parts by hashing identifiers, allowing processing load to be distributed across the shards
- *leader election* - electing which node in a cluster is the leader - the current authority for a database or shard
- *replication* - automatic and asynchronous replication of data to multiple nodes, ensuring that data is stored in more than one place at once

The diagram in figure 16 below shows a minimal setup for a Redis-based cluster[27] and how it might be set up in the AWS Elasticache service[28].

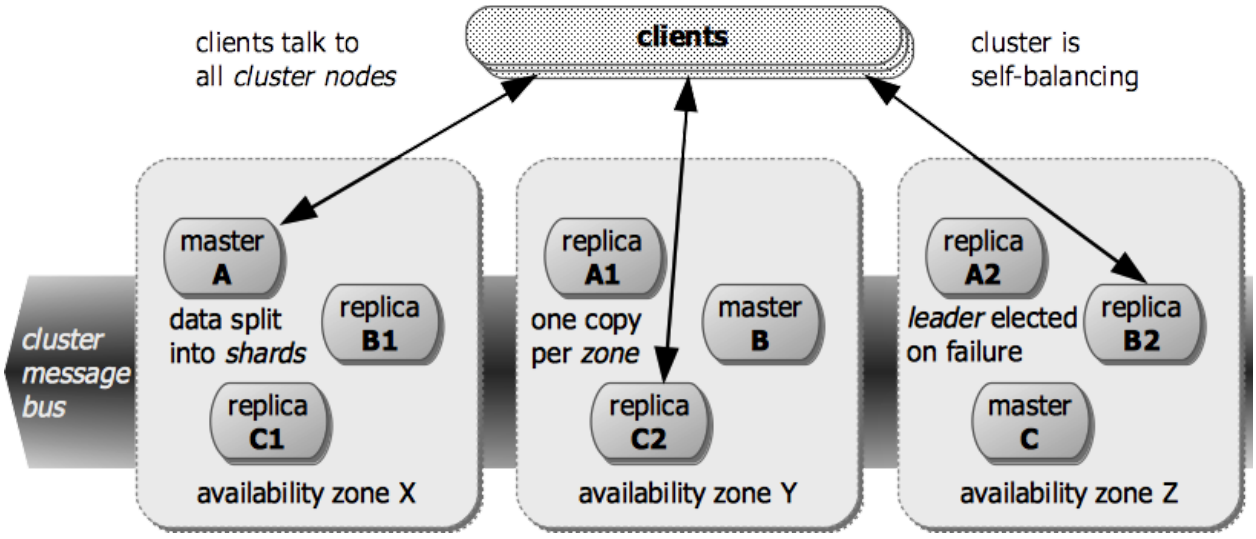



**Figure 16. Clustered Cache with Replicas Federated Across Availability Zones**

[27] https://redis.io/topics/cluster-tutorial

[28] https://www.slideshare.net/AmazonWebServices/unleash-the-power-of-redis-with-amazon-elasticache

In such a cluster, any cache client can talk to any cluster node. Requests are redirected over the *cluster message bus* (a backchat communication channel between all the cluster nodes) to the node that can deal with the request. A client which has been initially configured with addresses of only a couple of cluster nodes has the intelligence to learn about the current form of the cluster, including:

- which nodes are running or failed
- who to send which write requests to
- who best to make read requests from
- when to change the nodes it talks to due to loading issues or node failure

The diagram shows a configuration with three shards **A**, **B** and **C.** Each shard has a master node and two replica nodes **A1**, **A2**, **B1**, **B2**, **C1** & **C2**. The master and slave nodes are evenly distributed across availability zones so that any failure of a single server or a complete zone does not affect the overall operation of the cluster. In general, writes are redirected to masters and reads can take place from any cluster node, but a replica can accept a write in the event of a temporary loss of connection between itself and the master.

Automatic recovery of the cluster cache has one corner case. When the network between the availability zones fails (a *partition*), it is possible to lose the most recent write made to only one of the nodes. These occasions are rare and often result from multiple underlying system failures. This case can be mitigated with some application logic that executes after a partition is restored to determine if any data is missing.

## Jobs, microservices and workers

Caches have evolved to be business logic hubs, allowing queues of jobs to be executed by the next available worker in a pool of workers. Due to the inherent resilience of the cluster, it turns out that a good place to host the *job queue* is inside the cache itself, as shown in figure 17 below.

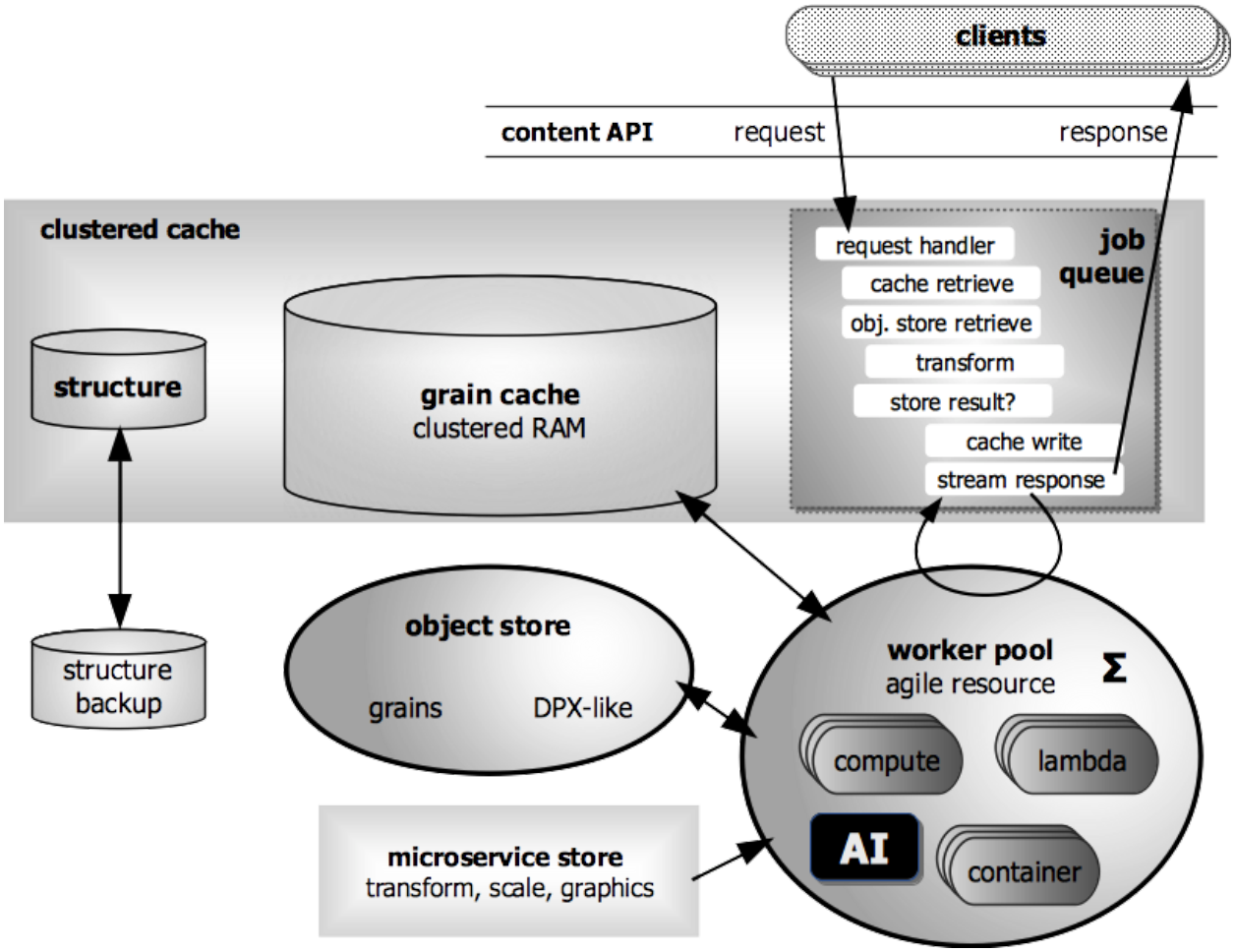



**Figure 17. Jobs Executed by Workers Running Microservices in the Context of the Clustered Cache**

The diagram shows the interaction of a client with a clustered cache over the Content API. The cache is serviced by a pool of workers - executing microservices that operate on data both from the cache and the object store. A job runs a function referenced by its name, implemented by a microservice and configured with input parameters.

The worker pool consists of compute instances for efficient processing of jobs using lightweight microservices, sometimes also referred to as *nanoservices* (e.g. Node.JS modules, Python modules). Less urgent jobs may be run by serverless-style lambda functions, streaming through big-data-style number crunching (e.g. OpenCL, tensorflow) or 3rd party services running in containers (e.g. docker, kubernetes). The microservices are themselves managed as versioned objects, often stored alongside the grains in the same object store.

The lifecycle of a typical request-to-response through the clustered cache is as follows:

1. A client makes a request for a grain that happens to be at a lower resolution than its original size, e.g. half height and half width.
2. The details of the request are recorded and a request handler job is put onto the queue.
3. The request handler job checks the structure database and determines that no grains for the requested source exist at the requested resolution, so it kicks off the process of making them. This starts with a race between cache retrieval and object store retrieval for the full resolution grains.
4. The request handler queues a scaling function as a job to run once the data starts to stream from the cache or the object store. At this time, the response to the originating client for the request can be started with a 200 OK status.
5. In parallel, the act of queuing of any transform work starts an agent that evaluates whether it is valuable to store the newly created grains, either for a short time in the cache, permanently in the object store or both. In case another client asks for the same scaled grain in the next ten minutes or so, the transformed grain is streamed back into the cache and not persisted onto the object store,
6. In parallel to 5, streaming data is now available to satisfy the client's request and so the cache starts to send the bytestream of the body of the response to the client.
7. Sending of the body of the response is complete and details about the work completed are placed in an audit log.

Caches such as Redis include helpful functionality for a job cue implementation, including, publish/subscribe messages that cause notifications to be sent when job status updates. The number of workers in the pool can be rapidly increased, to vary the overall system's capacity to do work, and subsequently may be decreased after demand has peaked.

Any important, non-transient and non-grain data in the cache cluster should be occasionally backed up to an external store in case of a catastrophic system loss.

## Bytes-to-time mappings

To store media on digital systems, content has to be quantized at a given sample rate and turned into blocks of data. Due to compression, variable bitrate encodings, and the need to interleave streams for efficient processing, the mapping between media timings and digital storage layout becomes complex.  By accessing media data through a time-oriented API, much of the complexity of mapping to bytes can be hidden. Accessing a particular segment of the media, is a matter of knowing a relative time reference.  There is no need to calculate the file byte offsets or compile index tables.

What follows are some examples of bytes-to-time mappings for various types of files and streams.

## MXF OP1a

The most common file format used in professional media production and playout is the MXF OP1a file[29]. This format is used in many playout servers, camera memory cards, and in the AMWA UK's DPP file delivery format[30]. One of the reasons for its popularity is its similarity to the tape formats that it replaced. Video frames and their associated frames-worth of audio and ancillary data are stored grouped together, avoiding the risk of an external audio file going missing or a video track starting to play out of sync with its audio.

MXF files are *write once, read many times* and the essence elements are stored at fixed byte offsets from the start of the file. Each essence element has a byte length. An indexer can scan through the file, building an index of where each essence element is stored, and determine which flow it is part of. The indexing process may use any index table stored in the file, although this is not essential if an indexing process reads from beginning to the end to create an external byte offset index. Once this is done - and it only needs to happen once - the index can be persisted and used to dereference the data for each grain by flow identifier and timestamp.

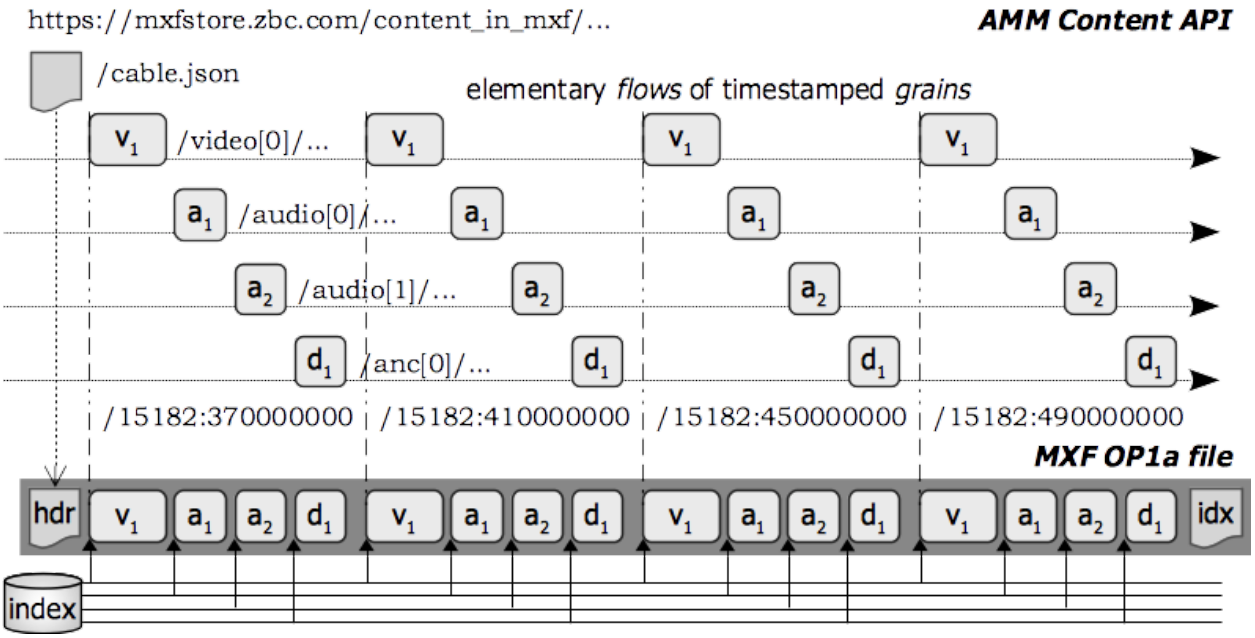



**Figure 18. Mapping the Essence Elements of an MXF OP1a File to the Content API**

Figure 18 above shows how the header of the MXF file is used to create a logical cable description. This creates flow identifiers for the tracks within the MXF file and a starting timestamp for each track, "/15182:370000000" in the example. The timestamps of each subsequent essence element-per-flow increments at the sample rate of the track. As an

[29] For more information on MXF files see, *The MXF Book*, Devlin et.al., ISBN-13: 978-0240806938 or *File Interchange Handbook* Gilmer et. al., ISBN-13: 978-0240806051
[30] https://www.amwa.tv/projects/AS-11.shtml

example, to address the raw audio samples for the second essence element of the second audio track, the following complete URL is used:

https://mxfstore.zbc.com/content_in_mxf/audio[1]/15182:410000000.raw

The reverse process can also be executed, creating an MXF OP1a file from a cable description using grains accessed through the Content API.

## IMF

The Interoperable Master Format (IMF) builds on the MXF file format (MXF OP1a mono essence files - typically one flow per file) by adding a composition playlist. This allows many different versions of the same program to be mastered without going through the expensive process of making a complete MXF OP1a file for each version. The example in figure 19 shows how an MXF OP1a Spanish master can be created from an IMF file containing a set of mono-essence files[31] representing English and Spanish language versions.

[31] Confusingly, mono-essence files may contain one or more audio channels, eg. mono, stereo or surround sound in one mono-essence file.

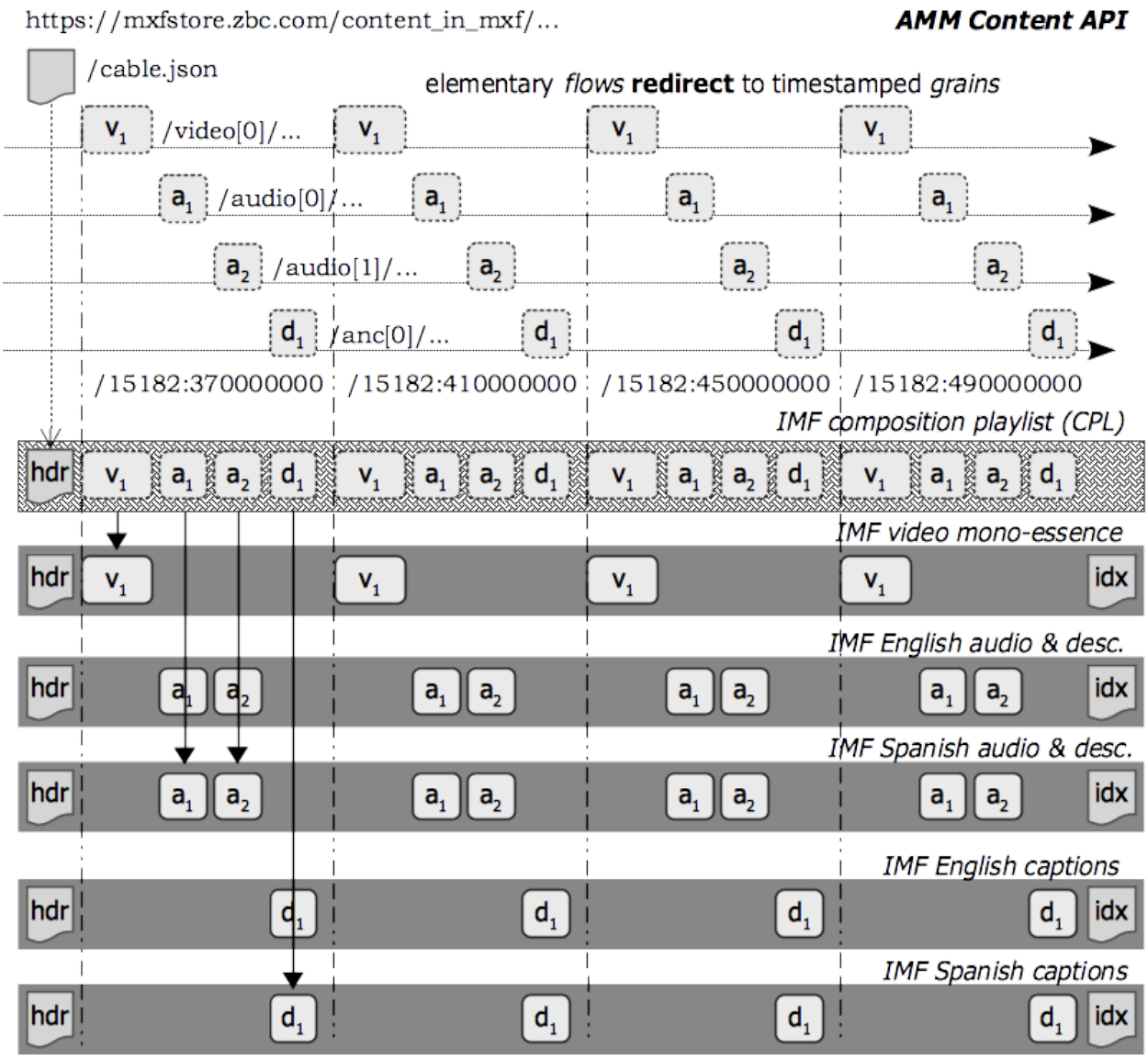



**Figure 19. Mapping Essence Elements of an IMF Package to the Content API**

For IMF-style processing using the Agile Media Blueprint, each mono-essence stream is a separate MXF file and an item of content in its own right. An IMF composition file references into the tracks that a particular master references, and is the basis of the master's logical cable description. In figure 19 above, the grains accessed through the Content API are represented with dashed lines, indicating that they references through to grains in other content items. These redirects could be to different stores. This kind of redirection, which could be implemented grain-by-grain and/or flow-by-flow, allows for the same mastering benefits as using the IMF itself.

## MPEG-TS

The MPEG Transport Stream (TS) is a stalwart streaming format used in digital video transmission, contribution circuits, satellite networks and for stream recordings as files. It consists of a sequence of 188-byte packets, a packet length originally chosen for its compatibility with ATM Networks. The transport stream is a wrapper for multiplexed streams of programs, with each program containing synchronized video, audio and data streams. For distribution, transport streams contain multiple programs. For applications within a facility, transport streams typically contain a single program.

Each packet has a *packet identifier* (PID) that links it to the stream that it is part of. All packets having the same PID are known as the *packetized elementary stream* (PES). Each PES stream is made up of a sequence of *PES packets*, each with a timestamp and payload. In terms of the AMB, PES streams are elementary flows and PES packets can be considered as grains. The mapping for a single program transport stream is represented in figure 20 below.

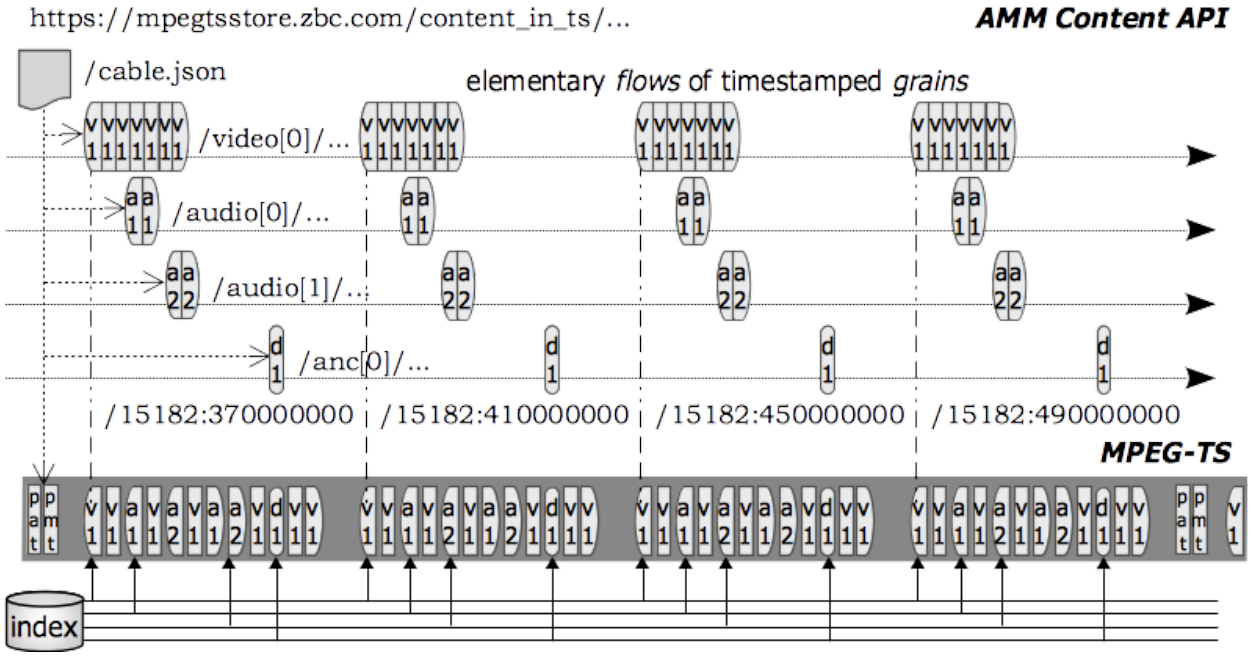



**Figure 20. Mapping MPEG-TS PES Packets to the Content API**

In the diagram, the first *Program Association Table* and *Program Map Table* of the stream are used to create the skeleton for the logical cable description. To complete the description of the stream, some additional information, such as *sequence parameter sets* or *picture parameter sets*, may need to be read from the stream itself.

An index of grains can be built by storing the PID and byte offset of every TS packet that has a *payload start indicator*, which signifies where a PES packet starts. Timestamped grains can be

extracted by reading the stream from that offset, filtering TS packets to those of the relevant PID, continuing to extract each grain as delimited by payload start indicators. An algorithm can be used to create a PTP timestamp based on the date creation of the stream and the presentation timestamp of the grain.

The reverse process of creating transport streams from grains is possible by using a small buffer and a multiplexing algorithm.

### Other transports

Uncompressed bytestream formats, such as SDI, ST 2022-6 and ST 2110, can be created from flows of uncompressed grains by a client that reads from a chosen subset of flows from a cable. By repeatedly incrementing a virtual clock, a set of synchronized grains is read into a buffer, formatted for transmission on the wire, and sent out. Processing of streams can be done in parallel, with the next set of grains being prepared while the current set is being sent. This process may add latency of a frame or two.  This is not that dissimilar to using a frame sync at the edge of a conventional facility.

### Live streams - bootstrap startup

When a client wants to chase the most recently produced grains for live and near-live streams, it makes a special request to the Content API to retrieve the most recent grain, e.g.:

https://entstore.zbc.com/blimpsons_s42_e27/video[0]/latest

In response, the client receives a redirection to the most recent grain, which includes its duration as a header parameter. Having started to retrieve that grain, the client can then take its timestamp, add the grain duration to get the next timestamp, wait a proportion of that duration and then make a request for the next grain. If it exists, great! If not, the client receives a *not found* message, waits a few milliseconds and makes another request.

In this way, a client bootstraps its connection to a stream and can balance the frequency of its requests to approximate real time. Streams can be transported in parallel, which is often more efficient on TCP networks. Uncompressed data can be split into sub-frame chunks to benefit from parallel transport and to keep latency low.

## Transformation microservices

In the AMB, vision mixing, graphics overlay and other functions are performed by microservices. Instead of being bolted into a rack and existing for many years, microservices exist and execute for only a few milliseconds. Their usage can be precisely metered and many versions of the same function may be deployed alongside each other, providing for scale-up.

This rest of this section explains how the AMB represents certain common media processing tasks as microservices.

## Media composition - transitions

A common operation in post production and live media operations is a transition from one video stream to another, usually an operation that lasts for a few frames. In non-linear editing packages, this is achieved by building a timeline, the output of which is a rendered representation of a sequence consisting of segments and transitions. In live operations, an output is created from two input streams mixed in proportion to the position of a T-bar lever. Figure 21 below shows how the equivalent workflow is achieved with microservices.

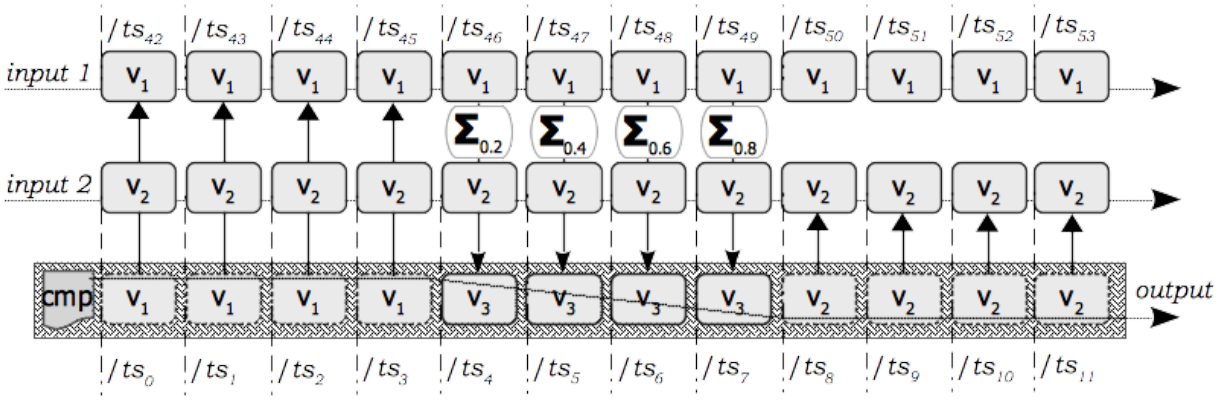



**Figure 21. Composition of Two Video Streams Using a Transition Function**

An example composition description (**cmp**) specifies:

- a new item of content with its own identity and flow of grains
- two inputs that are to be mixed, identified by Content API URLs
- the relationship between the timestamps of the two input streams and the timestamps of the output streams (*input 2* and *output* have the same timestamps in the example, *input 1* is offset by 42 grains)
- start and end timestamps markers for the transition and a further parameter requesting a linear rate of transition

As grains are pulled for the *output*, a composition render function makes the following choices:

- If the timestamp is before the start of the transition marker, redirect the *output* grain to the equivalent grain of *input 1*.
- If the timestamp is between the start and end, create a job to make a new grain for the *output* stream that is a mix of *input 1* and *input 2*, with the mix level in proportion to how far along the transition the requested grain is.
- If the timestamp is after the end marker, redirect the *output* to the equivalent grain of *input 2*.

The newly created grains may be cached for any future reference to the composition, meaning that the grains (‘$v_3$‘ in the figure) do not have have to be made again.

A T-bar vision mixer control, or its touch-surface equivalent works in a similar way except that instead of the transition being calculated from start and end markers, it is driven by the current position of the slider. Fully up then the output grains come from *input 1*, fully down is the output grains come from *input 2*, somewhere in between and a microservice is used to make the mix in proportion to the position.

Other mixing functions can be implemented in this way, including for example quad-split multiviewer, picture-in-picture, picture squeeze to make space for graphics etc..

## Graphics

A graphics functions works in a way that is similar to the transition, except that in this case, one of the input streams contains graphics with an alpha channel. The graphics input stream may consist of a single grain with infinite duration; a channel ident overlay for example. Or the graphics may change according to a wall clock, or timestamps, or may be a live stream from a graphics generator.

Graphics may have many versions, perhaps in different languages.  They may be designed differently depending on the target platform of the viewer - respecting the difference between a viewer on a large 4K television and one on a mobile phone. Using microservices to apply the graphics overlays as the stream is requested, allows for significantly more customisation and personalisation of streams without having to make a number of feeds just-in-case.

## Scaling, time squeezing, converting

Changing the resolution of the picture can be achieved on-the-fly, grain-by-grain, using microservices., Transforming the color depth, color model and chroma subsampling of an image may be achieved in the same way. These are one-to-one transformations - one-grain-in to one-grain-out - and operating on a whole content item at once may be achieved with multiple concurrent executions of transformation functions.

Other kinds of transformations, such as making a version of content that runs faster or slower than its source, are not one-to-one grain mappings. For example, to speed up some video by one third, four grains need to be squeezed into three. A function that takes every four grains and makes three, repeating every four frames along the length of a stream, is one possible solution. However, this will likely produce a visually unsatisfactory result. An alternative is to set up a transform function that accumulates a representation of the last few grains, combines it with next frame in sequence, producing both new grains and the next accumulation state. Frame rate standard conversions and film pull down techniques can be applied in the same way.

To achieve concurrency with accumulator functions, one approach is to use pre-processing to determine known good grains where processing functions should start; for example at cut points. Alternatively, functions may be overlapped at arbitrary boundaries, so that the last few grains of input into one function are also the first few grains input into its concurrent neighbor.

Finding the most efficient ways to perform these transformations, whether by CPU, GPU or FPGA processing, will require ongoing investment and development by manufacturers. Content creators will have a much greater range of choice as to how their content is filtered, scaled and converted via metered access to the facilities they want to use.  Suppliers will have the opportunity to compete with each other and create value for their companies.

## Transcoding, encoding and decoding

For many contribution, distribution and storage applications, bandwidth or storage volume constraints mean that compressing video essence is the only option.

Codecs come in three distinct forms:

1. Intra frame multi-pass codecs (e.g. AVCi, DNxHD) where one grain of uncompressed video is coded as a standalone data blob that can be decoded independently of those that proceed it or follow it.
2. Intra-frame low-latency compression (e.g. NDI, JPEG-2000, VC-2), also maintaining independent grains, that balance timeliness of processing with a lightweight form of compression that facilitates easier to transport within given constraints, e.g. HD video over 1Gbps networks.
3. Temporal codecs (MPEG-2, H.264, HEVC, AV-1), where a group of pictures or GOP, are coded together allow for a greater degree of compression by using motion detection between the pictures (grains). Although each picture still exists as an independent access unit, decoding a picture requires playing from the beginning of the GOP that contains it, and in some cases from the beginning of the preceding GOP.

Coding and decoding frames with intra frame codecs is a one-to-one operation that can be achieved with relatively low latency. Temporal coding and decoding is best achieved using an accumulator function that can build whole GOPs at a time. The use of independent GOPs with a fixed GOP length is commonplace for the chunks of ABR. To this end, the recommended representation for long GOP media in the AMB is one grain per GOP and, where possible, the use of closed GOP formats. This is so that each grain can be decoded independently.

Audio coding is similar. Some audio codecs have a fixed number of samples per blob, others are variable. The number of samples per blob should be reflected in the grain timestamps and in the grain durations stored with, or derivable from, the data payload of each grain.

# Technical Challenges

The technical details of the Agile Media Blueprint have been covered in previous sections. But if you want to build Agile Media Machines using the technology platforms available today, a number of technical challenges must be overcome. To understand these better, this section explains the difference between the AMB and current architectures. Further analysis looks into what the AMB does not do and how to re-use investment in SMPTE ST 2110 infrastructure.

## What is different about the Agile Media Blueprint?

What is different about the Agile Media Blueprint compared to conventional production and playout architectures in use today?

There has been an explosion! Media has been blown up into adaptive bitrate (ABR) chunks and it is these chunks, in combination with the power of the platform, that have become the default form of distribution for Internet media. Rather than pushing a signal down a wire from source to destination, clients pull the chunks of content they want just prior to displaying them. *Internet live streaming* is possible because the client is chasing the most recently produced chunks. As more content is accessed through Internet connections compared to broadcast streams, linear broadcast will become a client chasing the chunks, formatting them into a linear feed, and then broadcasting them.

The Agile Media Blueprint enables the construction of media factories that operate in a chunk-based world. The raw material inputs and outputs of that factory become chunks - flows of grains in JT-NM terminology. The Agile Media Blueprint is extremely pylant.  It makes no distinction between: live, near-live and file-based production; compressed and uncompressed formats; or various stream and file wrappers. Like the rest of the platform, the AMB works by elevating all media elements to self-describing data blobs.  As shown in figure 22 below, data in, data out and executing grain-by-grain data processing in the middle.

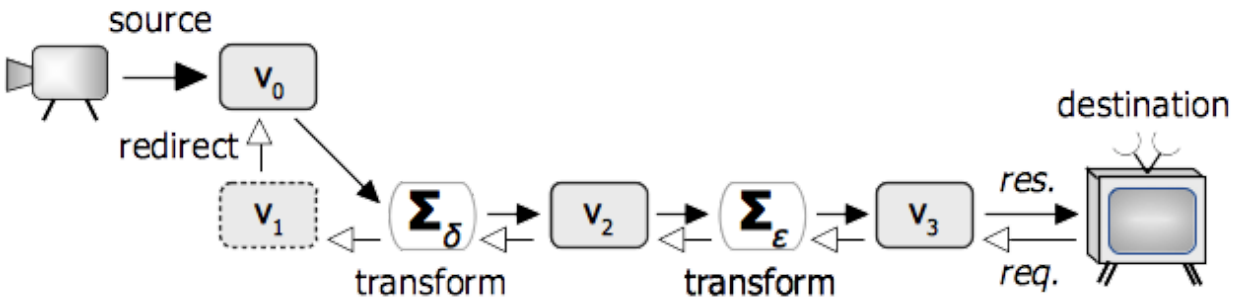



**Figure 22.  Connecting a Camera to a Monitor AMB-Style - Data and Functions**

In figure 22 above, a camera source called ‘Rocks HD camera 17’ is creating pictures in real time and pushing them to a stream recorder ($v_0$). An SD-resolution destination monitor is configured to view the live source, but requires a scaling transformation to shrink the pictures ($\Sigma_\delta$). It is a production requirement that timecode be overlayed onto the video. This is accomplished with an overlay function ($\Sigma_\varepsilon$). The monitor requests pictures frame-by-frame from the source (*req.*) via a transformation recipe and from a camera at a named location, e.g. ‘Studio F camera 3’. Today, that camera is configured to be an alias for the recording store ‘Rocks HD camera 17’. Responses for the redirected request flow back from the recorder to the monitor (*res.*), with each transformation stage executing in faster than real time.

Other examples of how the AMB is different from today include:

- The primary way to transfer content from a camera to a central production store is over 5G networks, not via camera memory cards
- Live streams are mixed together in a software-only microservice, based on timestamps, and controlled by a slider on a tablet
- The editing process uses streams to produce a personalizable composition description, a recipe for the final result, rendered on-the-fly without the need to make files and move them on and off local storage
- Playout servers do not have to be pre-loaded with interleaved files. Nor do they have to produce multiple outputs (e.g. clean, with graphics overlays) just-in-case, because the alternative outputs can be made on demand
- Immersive social TV - new kinds of content that combine live events or story telling with with social media content, e.g.: allowing a group of friends to collaborate to make their own commentary and vision mix of a sports event; or changing the direction of a story based on who or where or who you are

## What does the Agile Media Blueprint NOT do?

The Agile Media Blueprint is not a panacea. We fully recognize that there are a host of reasons why a media company may not move to this approach. The following section looks at areas where the AMB might not be a good fit in your media organization.

The Agile Media Blueprint does not:

- Deal with tight latency requirements
- Represent content as complete artifacts that are processed in batch mode
- Guarantee performance, other than through over provisioning

We will look at each of these points in turn.

Timing and Latency-

If you have tight latency requirements (not the same as synchronization requirements), then the AMB may not be suitable for your applications. That said, the capabilities and speed of the platform are continually improving, and the limitations of today will recede over time. One would be well advised to continue to evaluate the suitability of the AMB to even tightly constrained applications. (As a point of consideration, the performance of non-blocking network backplanes has increased 5 fold in five years[32].)

Before you rule out the AMB, study your workflows. If you can accept the fact that video, audio and data can always be presented synchronously, but perhaps with delay, then many new business opportunities are open to you. But if you are conducting interactive one-on-one interviews, or if you are in a scenario where delays from live action could lead to financial impacts (e.g. live sports gambling), then you may have to forego the benefits of the AMB.

Complete artifacts and batch processing-

If your operation requires the delivery of complete, “flattened” files, the AMB can produce those for you. It can efficiently produce subtly different versions as well. But if you need to check the overall packaging of a completed file in order to assure the quality of an item, then either the approval process may need to change, or you may need to continue using your current technology and workflow.

The AMB has the ability to create virtually any sort of technical representation of the media contained in its object stores, and it can do this “just-in-time”. This flexibility is highly desirable when feeding many different kinds of endpoints (many different makes and models of mobile devices, for example). In very high value sports events, however, media companies may prefer to render complete flattened files in order to de-risk their operation. The AMB can support this. However one would be giving up a lot of the value of the AMB if one always chose to make flattened files “just in case”.

Guarantee performance other than by over provisioning-

There is no fundamental guarantee of performance of the AMB beyond massive scale - millions of servers and switches, terabytes of connectivity, and world-wide diversity. the platform does not (with a very few notable exceptions) provide guaranteed priority of one type of traffic over another, or the provision of one service over another.

That said, the IT industry has developed strategies and best practices to deliver the performance that millions of companies find acceptable. It may be the case that for some

[32] See https://www.networkworld.com/article/3201043/private-cloud/arista-s-new-solutions-sets-the-standard-for-cloud-scale.html with 150Tbps equating to, theorectially, over 100,000 HD video streams.

high-value events the lack of guaranteed performance is an issue, and in these cases, the AMB may not be the way to go.

But does the fact that there is no inherent mechanism available to guarantee Quality of Service mean that AMB , at its core, is fundamentally unsuited for use by the professional media industry? No. However, if we are going to take advantage of the benefits the AMB offers, then we must understand its strengths and limitations.

## Re-use of SMPTE ST 2110 infrastructure

SMPTE 2110 is a good first step towards building IP infrastructure inside a facility. It enables the transition from point-to-point SDI to IP networking and elementary streams. It also supports more efficient carriage of media data payloads without blanking intervals. ST 2110 is a good choice in scenarios where network QoS can be assured, and where carefully managed hardware devices connect to a network.

ST 2110 devices can be used at the edge of fully virtualized AMB environments. They may also be used in applications where network management is outside the control of the user (e.g. in the cloud). But ST 2110 is difficult to read in modern software, and in cloud environments ST 2110 can be expensive or impossible to write. This is because 1) sufficiently accurate clocks may not be available in generic IT facilities and, 2) as previously explained, virtualized systems are non-real-time. However, with appropriate buffers and hardware/software, ST 2110 can be used as a gateway to the AMB.

Investment in ST 2110 technology is a move in the right direction. ST 2110 is complementary to the AMB Content API, and a step toward building a de-materialized facility.

# Conclusion

The AMB exploits a different kind of platform from today's facilities and will require a different way of thinking about how to execute current workflows. Do the large files and monolithic batch processing of current file-based media facilities transfer well to the cloud? Can you or should you replicate a precisely timed and highly synchronous point-to-point signal using a packet-based network?

By representing and manipulating media as grains synchronized by timestamps, you step away from the need to precisely time the flow of a linear bitstream down a cable, and into a structure for media that is much easier to transport, store and process on modern computer systems.

Non-real time, but not slow, these systems are asynchronous and massively parallel. They are connected to a collection of non-blocking networking optimised for thousands of concurrent connections. These networks contain computers with tens of CPUs, to thousands of GPU cores

in each computer. This sort of processing power is intriguing - for example, with a grain-based model and enough computing resources, whole feature film could be transcoded in parallel in a few seconds!

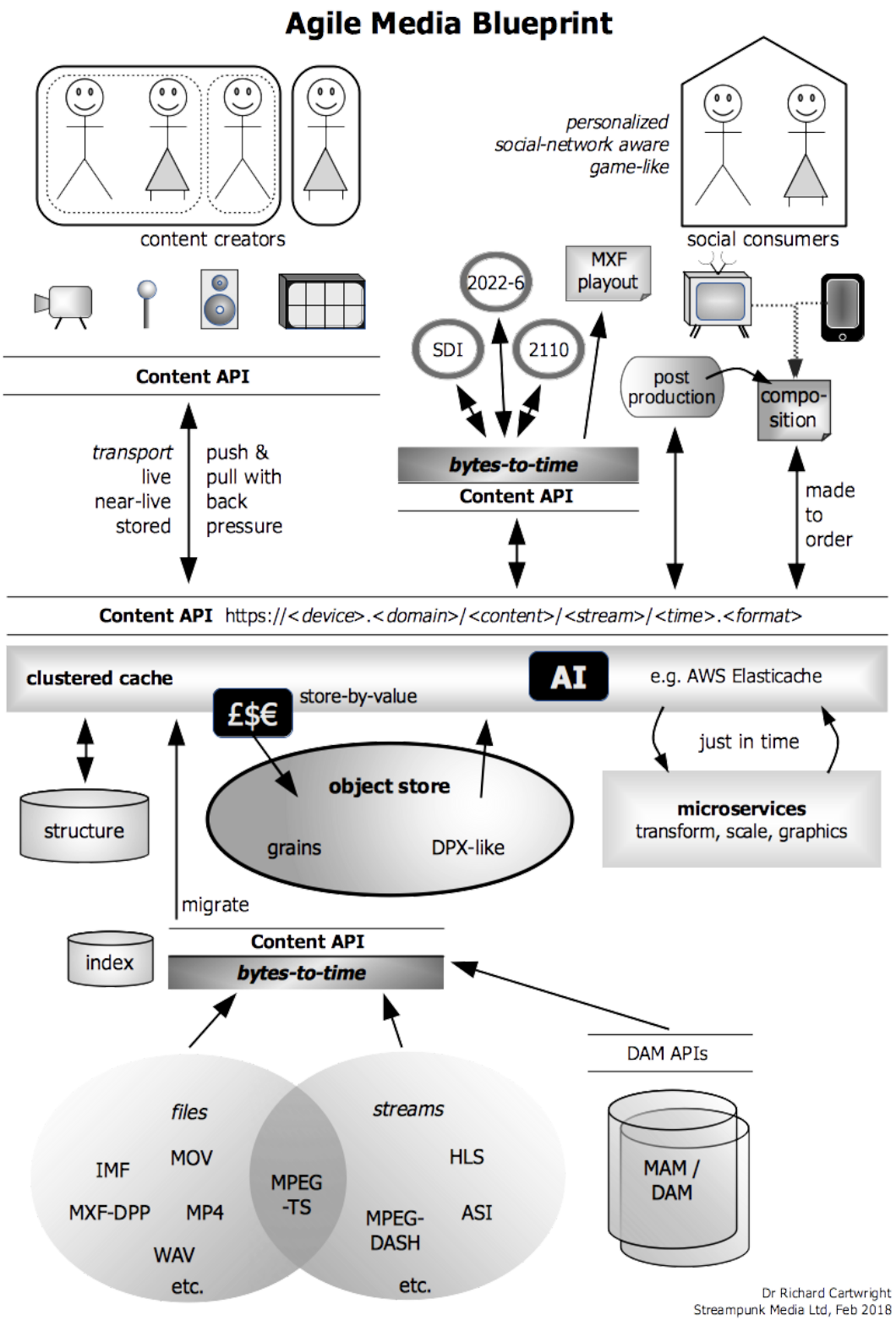
Whether you have performance sufficient for live becomes a question of what delay is acceptable and whether the vast majority of the grains are being transported faster than real time.

Many file-based workflows are batch-based, with processes that start at the beginning and read to the end. A grain-based approach is radically different; using a bytes-to-time API, file systems and object stores can provide random access into the grains wrapped inside each file. Files can be viewed as elementary streams, and in combination with virtual composition lists and efficient redirection, the versioning benefits of advanced containers such as IMF are achievable.

TCP, with its congestion control algorithms optimised for use within a facility, rightly gained a bad reputation for wide area network linear file transfer on congested networks. This created a market for UDP-based file accelerators. With major advances in TCP technology, TCP is now proving to be adequate for real time and even faster than real time media transport. This offers the option to do content creation and transfer just-in-time rather than file movement just-in-case. Moving files “just-in-case” is not only costly in terms of people and resources, it is also largely unnecessary in an AMB world.

In summary, many exciting opportunities for suppliers and media companies exist when the Agile Media Blueprint is leveraged to produce new media experiences.

# Appendix



# Biographies

## Dr. Richard Cartwright - Streampunk Media, Ltd.

Dr Richard Cartwright is CTO and founder of Streampunk Media Ltd. Richard holds a PhD in Computer Science from the University of Warwick, UK, where he studied that application of parallel computation to virtual reality models.

Richard has previously worked at the BBC, Snell Advanced Media, Red Bee Media and was Technical Steering Committee Chair of the Advanced Media Workflow Association. Now Developer Representative on the AMWA board, Richard is a co-author of the Joint Taskforce for Networked Media Reference Architecture and contributor to the JT-NM Roadmap.

He is a prolific developer of open source software for technologies including MXF, AAF, MPEG-TS, SDI and IoT workflows, in languages including C++, Java and Javascript. His ambition is to democratize professional media production ready to deliver a new breed of immersive social television.

Richard is an amateur musician, narrowboat captain and lumberjack at a community-owned forest.

## Brad Gilmer - Gilmer & Associates, Inc.

Brad Gilmer is President of Gilmer & Associates, Inc.  He is a founding member of the Joint Task Force on Networked Media, Executive Director of the Video Services Forum (VSF), and Executive Director of the Advanced Media Workflow Association.  Brad is also the Executive Director of the IP Showcase, an event held at both NAB and IBC, introducing and explaining the IP transition to the industry.

Brad is a SMPTE Fellow and the first recipient of the SMPTE Workflow Systems Medal.

Brad was previously employed at Turner Broadcasting System in Atlanta where he and his staff were responsible for Engineering and Operations for the Entertainment Division Worldwide.

He is an author and editor, contributing two times to the NAB Engineering Handbook, and served as Editor-in-Chief of the File Engineering Handbook.  He has written many articles on computers and networking for industry publications.

Brad as an expert witness in media-related court cases at the national level in the United States.

Finally, he is a national climbing and rappelling instructor, pilot, and blues harmonica player.