# AN ARCHITECTURE FOR CLOUD-BASED IP VIDEO PRODUCTION TOOLS

C. Northwood, R. Wadge

BBC Research & Development, UK

## ABSTRACT

Increasing use of IP networks in video production environments opens up many new possibilities. One of the benefits is that it allows us to apply architectural patterns from enterprise IT to the domain of broadcast and media production. This paper introduces a software system architecture that is being used by BBC R&D for development of its cloud-based media production tools. This architecture proposes a set of core functionality and separates the concerns of user interfaces for manipulating media and the actual processes which transform media. This is achieved through the definition of Application Programmer Interfaces (APIs) and protocols for this functionality, independent of the underlying implementation of the core and tools, building on broadcast industry open standards. It also explores how the in-development system was integrated with the BBC's IT estate, and explains how authentication, authorisation and security have been addressed.

## INTRODUCTION

The shift towards the use of IP networks at the core of production facilities brings a wealth of opportunities for redefining production and broadcast operations. To an increasing extent, infrastructure and workflow can be defined in software rather than hardware, and if properly architected, can enable systems that integrate off-the-shelf and bespoke components to fulfil the workflow requirements of an individual broadcaster. Architectures built on a foundation of generic IP networks are well understood in the IT community, and new broadcast infrastructure can benefit from this established body of knowledge.

Files have directly replaced tapes in most current non-live media production, but the resulting workflows often do not exploit the potential flexibility of networked media. In particular, interoperability of non-media data and metadata and the ability to trace production ancestry as file-based content passes through different stages and tools in the production process can be challenging.

The Advanced Media Workflow Association (AMWA) has recently published a family of Networked Media Open Specifications (NMOS) (1), which address some of the limitations of file-based workflows. These specifications take the data models and approaches outlined by the Joint Task Force on Networked Media's (JT-NM) Reference Architecture (2) and realise them using design patterns widely adopted in modern IT systems, such as

Representational State Transfer over HTTP (REST) as defined by Fielding (3). The most well-known use of REST is that of the world wide web, where individual web browsers make requests to web servers to access web pages, but its application to APIs for machine-to-machine communication has become the dominant pattern of web services today. RESTful services benefit from simplicity, extensibility, discoverability and scalability as they replicate the structure and semantics of the world wide web.

AMWA NMOS encourages components that implement a single service, such as managing connections for a specific device, with a well-defined API, in line with trends in IT towards microservices in a service-oriented architecture (4). Although the concept of a microservice is loosely defined, it can be thought of as a service that tackles exactly one problem. This is equivalent to the concept of the Single Responsibility Principle as described by Martin in Object Oriented Design (5) but at a service level. Systems with complex behaviour can be built by composing together microservices in a flexible, modular way.

A key feature of the JT-NM Reference Architecture is the separation of the abstract identity of elemental content (“sources”) from the specific encoding or representation of that content (“flows”). Identity of media is also globally unique, which means that any description or references can always be looked up and traced back to the same piece of media, regardless of which system generated the reference. Any transformations to media can be expressed in terms of these references, maintaining ancestry and other metadata throughout the production chain.

It becomes clear when trying to apply microservice concepts to an IP production toolchain that a separation is needed between the user interfaces that allow a user to manipulate media and the actual services handling the media. Well-defined interfaces and protocols built on top of the NMOS architecture can then expose this underlying functionality to the tools which make use of them, meaning the tools need only have access to proxy assets for preview and can use the NMOS identity model to hold references to the data across multiple systems.

## ARCHITECTURE

The proposed architecture uses the capabilities given by the NMOS specifications, with the constraints mandated by a modern, secure, cloud-based deployment target. It consists of a set of microservices that expose certain capabilities, a deployment pattern for them, and a data model. The data model is the NMOS implementation of the JT-NM reference architecture, but the system is most concerned with the concept of sources as a way of addressing media assets and their metadata, as well as describing transformations which are performed on media.

Figure 1 shows the major components of this software architecture. There are three key areas of the system: the production core, gateway services, and client code. The production core consists of a cluster around NMOS IS-04 Discovery & Registration servers, but these core services are not directly exposed to end users. The gateway services are services that are deployed to the production core, and register with the NMOS registry, but also expose services to the outside world. The final classification of system is that of client code, which runs on the end-user’s device, and uses the gateway services to

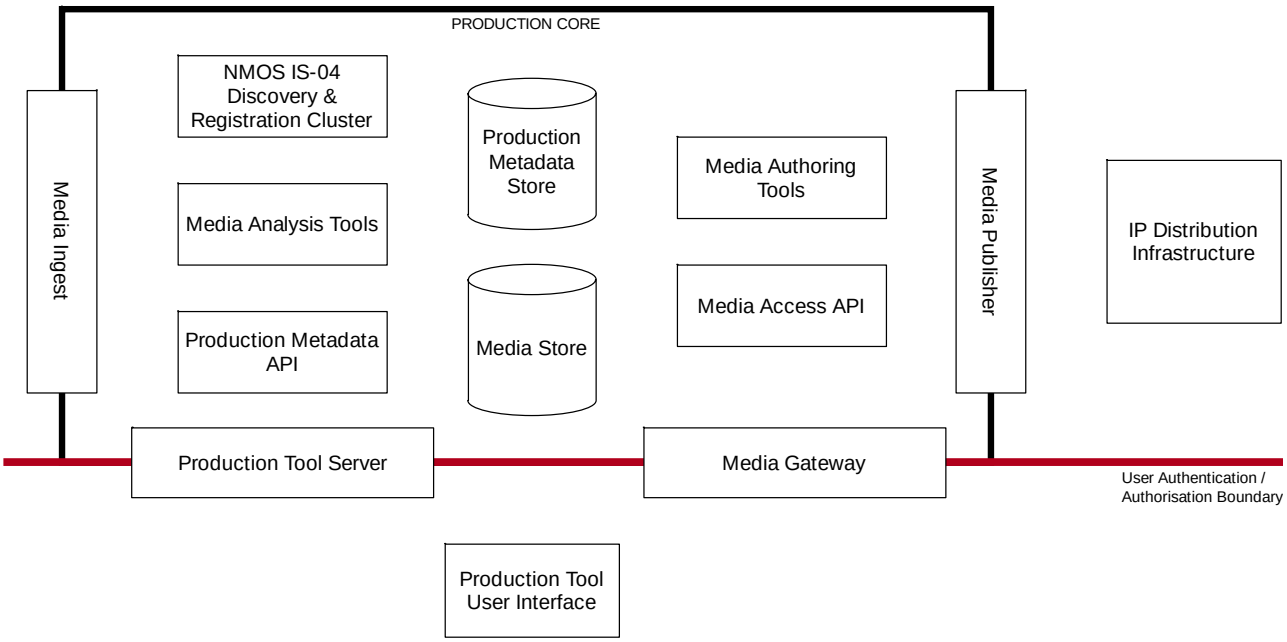


Figure 1 - System Diagram of Proposed Architecture

interact with the underlying media and make metadata manipulations. Other services may also exist to support live production use cases that can be integrated into the same production core.

The ultimate purpose of this architecture is to support production tools. These consist of two closely coupled components, a tool server which runs as a gateway service in the production core, and then a user interface which runs on the end user's device (for example, a web application in a browser). This is an application of architectural pattern named by Newman (6) as "backends-for-frontends". The tool server is then responsible for proxying through actions and data between the production core and the user interface, applying authentication/authorisation logic at that time. Any other intermediate business logic for the use case of that particular tool is also the responsibility of the tool server, for example storing tool specific session or configuration data.

Although the diagram only shows a single production tool, the architecture is flexible enough to either be deployed as a "black box" within existing production infrastructure, with each instance supporting a single tool, or as a single core supporting multiple tools. The benefit of the latter approach is that narrowly-focussed tools for different craft roles can be designed to work together in a co-operative way on the same underlying metadata and media assets. However, it is rare to have the opportunity for a green-field deployment of new production infrastructure, so the deployment of a "black box" approach for an initial tool with bridges to existing infrastructure allows you to start the move to IP production, whilst giving you the flexibility to grow the role of services in the production core over time as an organisation adopts an NMOS-compliant IP core.


**Security**

A core concern that led to this design is security. One of the benefits for end-users of cloud-based working is the ability to work from anywhere, and on their own devices. Exposing the NMOS services directly to the Internet may be an option, but a common approach to IT security usually involves minimising the exposed surface of your APIs to minimise any attack vectors. Additionally, the NMOS specifications do not currently specify how the concepts of authorisation and authentication should be implemented, but these are an important concern for any cloud-hosted system. In order to minimise the exposed surface area, and to avoid pre-empting any specification work in NMOS, this architecture uses the production core as an area where only trusted components can be deployed. Gateways allow the core to interact with other parts of a larger system, as well as for end-users who wish to access these tools from the cloud-hosted core. These gateways, and the “backends-for-frontends” pattern are mostly responsible for dealing with these concerns of authorisation, leaving the production core to focus on the production domain and limiting the exposed area of the system.

One mechanism for securing the production core is by using network-level isolation, such as physical separation of networks, the use of Virtual Local Area Networks (VLANs) and physical protection of network infrastructure. This can be useful as part of a layered approach to security, but cannot be the sole solution, as complete isolation is incompatible with a cloud way of working. From NMOS IS-04 version 1.1 we can use secure HTTP (HTTPS) with Transport Layer Security (TLS) client certificates (7), giving us an additional layer of security. Certificates are granted to an individual or application, allowing us to identify the caller of an endpoint and to encrypt communications in transit between different nodes. This can be combined with a per-endpoint whitelist of authorised users. Effective use of client certificates in HTTPS requires a public key infrastructure with vendor support for these, but no vendors have yet made this a priority, due to a lack of demand and the complexity involved with its management.

Although production end users do not have direct access to the production core, operations engineers do. One advantage of the TLS certificate approach is that operations engineers can access a cloud-hosted production core from any location provided they have their credentials, as opposed to being required to be on a particular network. Alternative approaches such as Virtual Private Networks could also be feasible.

However, the code running on an end-user’s device requires a mechanism to communicate with the production core. This is the role of the tool server. The tool servers, like other gateway services, face both the production core and the wider Internet, and so become areas of security-critical code. To address the requirement of user-based access to resources and tools, this architecture exploits the “tags” field of NMOS resources to tag resources with a project ID, which can then be used to determine if a user has access to a resource.

Each tool allows a user to pick a project to work on, and an authorisation service states which projects a user has access to. When a resource is created, the project tag is added, and when any resources are fetched, the tool server ensures that the tag on that resource matches that of the project the user is working on. Whenever any further manipulations are made to those resources, the resource’s metadata is requested to verify that the project

tags of that resource match that which the user has been authorised to manipulate. This workflow is shown in Figure 2.

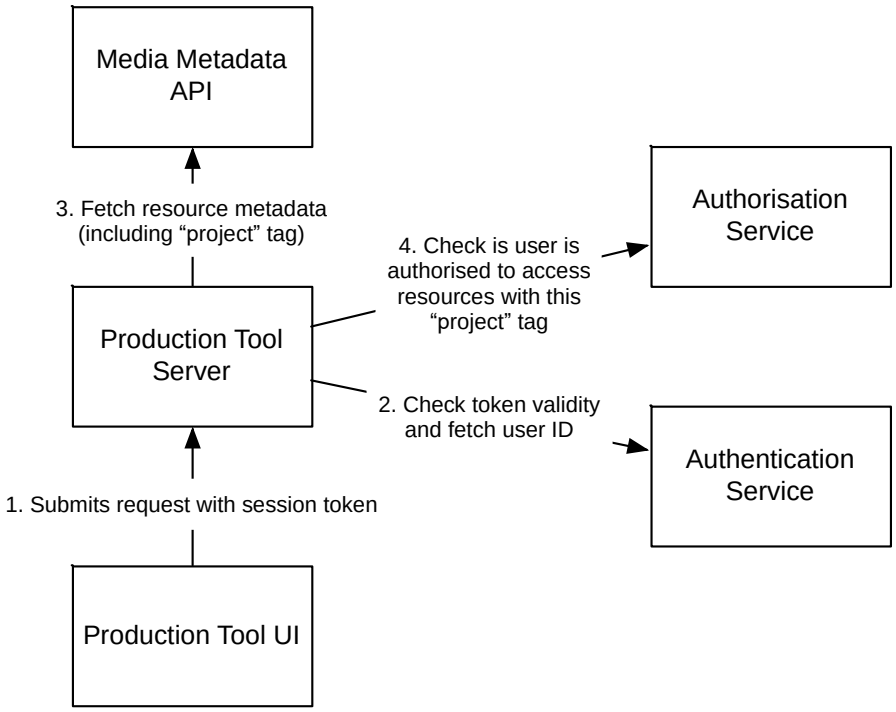


Figure 3 - Authentication/authorisation workflow

This integrates with a wider single sign-on system which issues session tokens to users who authenticate using that service. The tool server redirects users to the sign-on page if the token is missing or otherwise invalid. Additional protections can also be applied, such as using network isolation or firewall rules to limit access to a corporate intranet, but this loses some popular efficiencies of cloud workflows, such as “bring your own device” and flexible workflows.

**Production Core**

Inside the production core, we have identified several services which are needed to support production tools: a media store, a media access API, a metadata store, a metadata API, services to support media analysis, and services to support media authoring, in addition to the NMOS IS-04 discovery & registration services.

The media store component abstracts the underlying storage to provide access to media in terms of flows. Object stores as described by Mesnier et al (8) provide a suitable model to follow here, where an individual object within an object store consists of a segment of essence (video, audio or data). This introduces complexity for users who then need to concatenate or trim segments to get the time range they are interested in. Additionally, there could be many media stores within a system. To manage this complexity, a façade is needed over the media stores to provide access to the media. This could also be extended to support other types of store. This “Media Access API” exists to serve the needs of the tools which access the media. The interface is simple, it accepts a source ID, a description of client capabilities (for example, container and codec support in a HTTP Accept header) and a requested time range. The role of the media access API is then to locate appropriate stored flow segments from the underlying media stores and provide them to the end user in an appropriate container (e.g., as an MPEG-DASH adaptation set, described in a manifest, or as a single MP4).

In an NMOS-based system, raw data exists as an essence type alongside video and audio. This could include timed data such as subtitles or metadata such as logging notes,


or the output of automated analysis. This kind of metadata is treated like any other media flow, but with a payload of structured data, rather than a video frame or audio sample. Other types of metadata are needed too, to describe relationships about the structure of media and to assist in indexing media within a content store. For example, a multi-cam shoot may want to group which video sources and audio sources were used for a shoot, as well as time ranges which indicate at which times takes were made and hence the media is available for. Similarly, rendered output might want to be reflected as a group of video and audio to give sufficient metadata for a multiplexer to use for delivery. Mirroring the structure of media stores and media access APIs, this data is kept in a metadata store, with an access API abstracting across them to allow the tools access to this data.

The final two components of the production core deal with new software-defined services which become integral parts of the production chain. Content analysis is proving an extremely effective tool to assist with workflow, with the metadata generated from these tools assisting craft users in their day-to-day workflow. A simple use case might be to run speech-to-text against ingested audio to speed up discovery of content in rushes and archive material, or to perform automatic quality analysis on edited media. The output of these content analysis tools is a new NMOS data that is related back to the analysed media using the parent attribute in the NMOS source/flow resources.

Finally, as the tools themselves only need to describe transformations to media in terms of actions on sources, a way is needed to record these transformations that, when applied, author new sources from the raw ingested content. This is fulfilled by the media authoring tools. By making this part of the production core, dynamic ancestry data for a derived source can be retained as transformation and authoring data is recorded, giving users the ability to see the history of content through the production process. For example, for edited audio, speech-to-text only needs to be run on the original audio. Tools can then carry the results of this analysis downstream by using this ancestry, rather than having to analyse the newly authored content as if it were brand new.

### Gateway Services

In addition to the services in the production core, there are three key services which bridge the production core with external systems: the media ingest function, the media gateway, and a media publisher. In a complete IP-based chain, the media ingest function would be at the point of capture straight from the device, but for the foreseeable future, there are likely to be many ways of media entering an IP production core, from SDI capture, to ingest of archival material, or files transferred from a camera card. The media ingest function primarily deals with converting material into the grain format and ensuring they have an appropriate identity, but could also deal with matters such as transcoding flows into low resolution proxies for use in browsers. The media gateway then deals with making ingested material and proxies available to the client devices of users of the production tools. It works by applying a layer of security across the media access API in the production core. The media gateway provides content by proxying the media stores to outside of the production core, but does so using URLs that have secure, time-limited tokens in them. The tool server also uses the media gateway to generate these signed URLs for media so they can be accessed by client devices, usually by requesting a low-res proxy of the source.

Content must also leave the production core to be distributed to audience members. This is achieved using bridging gateways like those used for ingesting media. The gateways will typically produce files in standard interchange formats for delivery to distribution infrastructure, typically by multiplexing together video, audio and subtitle flows into an appropriate container format, or an appropriate streaming format.

## EXTENDING THE REFERENCE ARCHITECTURE

The JT-NM Reference Architecture is designed in such a way to allow for organisation-specific architectures to be built on top of it. BBC R&D is developing domain specific architectures for data using this capability, with the intent to make these available as open specifications for interoperability, in addition to organisation-specific extensions. RESTful services exposing various APIs are also being developed, which advertise themselves as services on an NMOS node.

One key data model and set of APIs being developed is one that describes the narrative structure of a story, required to support tools for script writing. This interoperates with a descriptive language of how different media elements are composed together dynamically on an audience member's device. This data model is further described by Cox et al (9), supporting a range of media experiences, from traditional linear, to non-linear immersive experiences.

Data models also exist to associate manually captured production information, such as scripts or logging data with the media in the system. The results of content analysis, such as speech-to-text, shot changes and face detection create relationships that are modelled in a similar way. This rich mesh of data can then be exposed to assist tool users.

To separate the production tools which manipulate content from the actual act of rendering the newly authored content, a protocol is needed to describe the transformations to media and compositions of media to create completed output.

In this protocol, production decisions relating to media composition and processing are captured and stored as events on a composition timeline, describing how to construct a new NMOS source. This comprehensive description is similar to a recipe that describes how to combine media ingredients through a defined set of processes to re-create the composition precisely. Since the ingredients are referenced using their abstract source IDs, the recipe can be created in real time by a web-based tool working with low resolution compressed proxy flows and subsequently applied to full quality representations.

The published recipe and the rendered result share a source ID as they are logically equivalent representations: the event-based format of the recipe can be directly streamed, and is also splice-able. A query to the Media Access API for a given source can return either format, based on which is requested and/or available, so a client can choose whether to optimise for simplicity or flexibility.

## INTEGRATION WITH IT ESTATE

As these tools are all software, they can be deployed and managed to an IT estate like any other production-critical IT system. Each component of the system can expose status pages and metrics for centralised monitoring by standard tooling, as well as providing log

files which can be aggregated for analysis by tools and to assist in debugging issues by an operations team.

This architecture also allows for packaging of application components using standard mechanisms for the deployment target and as such, management of the infrastructure and configuration management can be achieved using industry standard orchestration tools.

## SHORTCOMINGS & NEXT STEPS

There are issues with this architecture. One is that some layers of abstraction attempt to hide underlying complexity that sometimes need to be exposed due to implementation concerns. For example, utilisation of public cloud providers is attractive due to a low barrier to entry, but often cannot provide the necessary performance guarantees or be restricted by networking bottlenecks. A future extension to the architecture to support multiple production cores with gateways between them may allow deployments to take advantage of different environments but still behave as joined up IP production infrastructure. Similar issues due to this abstraction come when you try to scale media storage: the media store assumes that content is always online and available, which may not always be true or the most cost-effective way of managing media.

Another issue is that the security model based around project access is not fine-grained. A user either has complete access to all resources tagged with a project ID, or no access at all. More flexible protection will be needed to satisfy more complex production workflows.

The next steps for this architecture should focus on addressing these shortcomings, especially around the security model. Trials of the concept of multiple production cores with gateways between them will validate the wider applicability of this approach. This work could feed into open specifications addressing security, allowing multi-vendor interoperability for tools at the edge of the production core.

Other follow-on work could look at developing further organisational and domain specific data models, APIs and protocols that may evolve into open standards. One such model may be one for media rights, with the aim of simplifying reporting and compliance requirements.

## CONCLUSIONS

The architecture shown above has been implemented for a small number of trial systems inside BBC R&D using the BBC's standard IT deployment infrastructure and tooling, and satisfying the BBC's information security requirements. This shows promise as the foundation for future iterations of IP production infrastructure in the cloud as well as simplifying engineering effort by bringing best practices from IT architecture and operations management into broadcast infrastructure.

## REFERENCES


1. Advanced Media Workflow Association, 2016. Networked Media Open Specifications. https://github.com/AMWA-TV/nmos.
2. Joint Task Force on Networked Media, 2015. Networked Media Reference Architecture V1.0. http://jt-nm.org/RA-1.0/.
3. The Open Group, 2009. SOA Source Book. Van Haren Publishing.

4. Fielding, R. T., 2000. Architectural styles and the design of network-based software architectures. Doctoral dissertation, University of California, Irvine.
5. Martin, R. C., 2003. Agile software development: principles, patterns, and practices. Prentice Hall PTR.
6. Newman, S, 2015. Pattern: Backends For Frontends. http://samnewman.io/writing/.
7. Dierks, T., Rescorla, E., 2008. RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2. The Internet Engineering Task Force.
8. Mesnier, M., Ganger, G. R. and Riedel, E., 2003. Object-based storage. IEEE Communications Magazine. August, 2003, pp 84 to 90.
9. Cox, J., Brooks, M., Forrester, I., Armstrong, M., Stenton, P., 2017. Moving object-based media production from one-off examples to scalable workflows. Submitted to the 2017 International Broadcasting Convention.