dpp®

MAKING INTEGRATION WORK

Delivering Seamless Integration

Enabled by

VUBIQUITY
An Amdocs Company

FOREWORD

With any new technology the path to success often presents intricate and complex challenges. So, as we explore and deliberate the power of integration, we ask ourselves, "Why is integration important? What benefits and insights does it offer? Which solution is best to implement? What connects all the data together?" These common questions likely elicit different responses, dependent of many factors. Certainly, we must step outside of the comfort zone of how we do things today. We must seek greater collaboration to unravel the true power of integration so that we can realize its maximum potential for tomorrow.

As media companies strive for effective integration of their systems and processes, its success is heavily predicated upon an ethos of flexibility. Various methodologies and approaches exist, some on premise and some cloud-based, each with varying degrees of functionality and capability. Big picture integration opportunities must address business applications, user experiences, multiple data sources, orchestration workflows, as well as security, monitoring, governance, and much more. We know the solutions exist, but then we wonder, "Who has the expertise to deliver?" Chances are that you'll need strong contributions from both your current talent pool, as well as outside experts and vendors, to ultimately meet your business goals.

Vubiquity proudly supports the DPP and its ongoing, steadfast efforts to share knowledge across its membership, and encourage collaboration to solve media industry problems. As sponsors of the *Making Integration Work* series of reports, Vubiquity is committed to shaping the evolution of integration to enhance its viability, drive its effectiveness, and produce measurable results for all parties involved across the media supply chain.

Raman Abrol
Chief Executive Officer, Vubiquity

MARCH 2022

MAKING INTEGRATION WORK

Delivering Seamless Integration

Introduction

In *The Integration Opportunity* we identified that media companies are increasingly reliant on integration capabilities to deliver their business goals.

Media companies increasingly rely on integration capabilities to deliver their business goals

Some of the key findings included:

- The seamless integration of software applications delivers effective and efficient end to end workflows.
- The integration and aggregation of data sources can uncover new business and operational insights.
- A flexible, composable approach enables companies to respond more quickly to changing context, unexpected threats, and new business opportunities.

So how, then, can these benefits be delivered?

In this report, we've taken input from 60 experts in the DPP membership to find out. It does not seek to be a detailed software engineering guide to integration methodologies, as there are plenty of existing rich sources for such information. Rather, it offers a necessary bridge between the strategic objectives identified in *The Integration Opportunity*, and the practical realities of implementation for media companies.

A bridge is necessary between the strategic objectives of integration and the practical realities of implementation

It uncovers the key components of integration, and the architectures and patterns by which those components are connected. And it asks when a media company would choose one approach over another.

This report considers integration practices that apply both on premise and in the cloud. However, readers may find it useful to review our guide to the key terms of ***Cloud Technology***.

MAKING INTEGRATION WORK

Delivering Seamless Integration is one of three reports in a series, *Making Integration Work*.

Each document in the series examines a different aspect of integration; the others focus on ***The Integration Opportunity***, and ***Integration Expertise***.

You may also be interested in ***The Cloud for Media*** series, which explores the broader topics of cloud migration for media companies, and examines specific use cases including automation, post production, playout, and streaming.

Executive Summary

1

Integration is evolving

Modern software integration has of course changed a lot compared to historic hardware integration. But software integration itself also continues to evolve. Focus is increasingly placed on integrating data and building a flexible framework to enable future change.

2

APIs and messaging are complementary

REST APIs are at the heart of the modern integration ecosystem, and are often the best choice for accessing data in real time. Message based architectures meanwhile are generally superior for one-to-many communication and asynchronous processes.

3

Abstraction enables flexibility

Although there are many point-to-point integrations in media, it is now considered best practice to use a separate integration layer to manage API access, enforce security policies, and perform functions such as data mapping between systems.

4

Business logic must be carefully managed

Custom business logic is key to delivering end to end workflows and data flows, and is best separated from the individual applications being integrated. Although this can be done in custom 'glue' code, deploying it in an integration or orchestration platform is usually more scalable and manageable.

5

Strategic integration enables business users

By providing access to data and functionality from multiple business systems, an integration layer offers the chance to remix components to meet new business needs. Low-code tools can even allow companies to make these capabilities available to users across the business.

Contributors

The content for *Making Integration Work* has been gathered through workshops and interviews with subject matter experts from across the industry. Valuable input has also been provided by our Lead Sponsor **MuleSoft**, and our Expert Sponsors: **mediaSaaS**, **Mux**, **Ross Video**, **Signiant**, **Skylark**, and **Vubiquity**.

Although the content of this report has been informed by these discussions, it should not be assumed that every contributor shares all the views presented here.

Stuart Almond
Industry Director –
Media & Communications, Microsoft

Richard Amos
Chief Product Officer,
Skylark

Peter Anderson
Senior Director, Video Systems,
VICE Media Group

Abhineet Asthana
Lead Solutions Engineer,
Telco and Media, MuleSoft

Bill Baker
CTO,
MediaWR

Hans Baumgaertner
Senior Cloud Solutions Architect,
Techtriq

David Bird
Founder & Managing Director,
dB Broadcast

Johanna Björklund
CTO,
Adlede

Dominic Brouard
Media Systems Architect,
VICE Media Group

Adam Brown
Co-founder & Head of Technology
& Architecture, Mux

Mike Bryan
Technology Director,
dB Broadcast

Kristan Bullett
Founder & CEO,
Humans Not Robots

Tom Burns
CTO, Media & Entertainment,
Dell

Bruno Cardoso
Head of Digital Integration, Europe,
Cognizant

Andrew Carroll
Customer Success Manager,
RSG Media

Gordon Castle
SVP, Technology and Operations,
Discovery

François Chabat
CEO,
BeBanjo

Paul Charleston
Senior Solutions Architect,
Qvest

Arran Corbett
CTO,
FooEngine

Cory Crabb
Solutions Architect,
Rightsline

Margaret Craig
CEO,
Signiant

Sebastien Creme
CTO,
Nomalab

Stewart Curtis
Executive Director, Operations &
Portfolio Delivery, WarnerMedia

Daniel Elias
Global Director of Production &
Media, Workflows, VICE Media Group

Troy English
CTO,
Ross Video

Jon Finegold
Chief Marketing Officer,
Signiant

Mike Flathers
Chief Solutions Officer,
Signiant

James Gibson
Founder & CEO,
Ortana Media Group

Sean Gigremosa
VP of Product Management,
Premiere Digital

Peter Hajittofi
CEO,
coralbay.tv

Michael Harrit
Lead Architect,
BBC

Ashley Horne
Technical Director,
Simplestream

Dan Jones
Video Architect,
DAZN

Antony Joyce
Head of Architecture and
Technology Platforms, UKTV

Christoph Jurkuhn
Head of Product & Presales,
Hiscale

Fereidoon Khosravi
SVP and Chief Business Development
Officer, Venera Technologies

David Klee
VP Strategic Media Solutions,
A+E Networks

Tim MacGregor
Head of Strategy and Product
Development, Cloud, Telestream

Alexis Martinez
VP, Research & Innovation,
Iyuno-SDI

Ian McLaren
Technical Director,
Reuters

Michele Memoli
Managing Director,
100 Shapes

Joe Newcombe
Client Technology Lead,
Microsoft

Chris Oakley
CTO,
ZOO Digital

Chandni Patel
CTO,
Emotion Systems

Brie Pegum
VP Product,
Imagen

Alan Pimm
Founder & CTO,
mediaSaaS

Krishna Pothula
Head of Architecture,
RTL Nederland

Paul Randall
Consulting Software Engineer,
Avid

Martin Richards
Head of Broadcast Technologies,
Sky

Hilary Roschke
Vice President of Strategic
Operations, SDVI

Atul Saxena
SVP, Product Engineering,
Prime Focus Technologies

Jan Schütze
Head of Development,
Endava

Craig Seidel
CTO,
Pixelogic Media

Magnus Svensson
Media Solutions Specialist,
Eyevinn

Dave Travis
Director of Content,
Broadcast & Platforms, Sky

Shyam Visamsetty
Managing Director,
Navtech

Duncan Warrick
Product Manager,
Blackbird

Nick Wright
CTO,
Pixel Power

Oliver Wynn
Distinguished Solution Engineer,
Digital Transformation Office, MuleSoft

Joe Zaller
Founder,
Devoncroft Partners

1

The changing face of integration

KEY INSIGHTS

- The process of integration has evolved over many years, from hardware to appliances, to software and then the cloud
- Whatever technical interface is used, the common factor is data being exchanged between systems. That data must be properly managed and understood
- Integration is not a one time process; strong technical foundations will enable ongoing improvement, and agility in the face of changing business needs

One might assume that there was a step change in the process of systems integration when moving from hardware to software. That these two worlds require distinct approaches to integration.

However, the reality is that there was not one step that took us from hardware to software. Instead there has been more of a continual evolution.

Custom hardware began to give way to 'appliances': computer servers running pre-installed software, often with specialised hardware cards for features such as video input/output. Pure software solutions first emerged as 'monolithic' applications that include all functions in a common codebase and runtime, and now commonly use microservices architectures instead.

Increasingly, software applications – whether monolithic or microservices based – are made available via the Software as a Service (SaaS) model, in which the vendor hosts the software in a multi-tenant instance, and customers access it via web user interfaces and/or APIs.

This simplified description serves not as an authoritative history, but to underline two simple truths:

1 we have always needed integration

2 the nature of that integration undergoes regular evolution

The nature of integration is continually evolving

> Integration between vendors isn't new. Even integration at the software level isn't new. The first job I ever did at Ross 30 years ago was to write an API to control another company's product. But the mechanisms and the technologies have evolved significantly. We now have an opportunity to make it easier, because historically every single company had their own interface, and we had to write drivers for every product individually.
>
> TROY ENGLISH, ROSS VIDEO

Many of the underlying principles of application integration can be seen as logical extensions of the way systems were connected and controlled in the age of hardware and appliances.

However the rise of more modular software systems, and of widely adopted interfaces such as REST APIs (Representational State Transfer Application Programming Interfaces) has opened up a new level of flexibility.

The rise of modular software and REST APIs has opened up a new level of flexibility

> There are a lot of best practices learned 20 years ago that are still completely applicable today, when it comes to abstraction and representation. It's the method of integration that's significantly changed, as we've gone from RS-422 serial connections into REST interfaces.
>
> JAMES GIBSON, ORTANA MEDIA GROUP

Modern applications often use APIs very extensively, including for communication between the user interface and the rest of the application. Using the APIs in this way provides abstraction between the user interface and the application, which can be useful for developers. But it also means that those APIs can be opened up for others to integrate with, or develop their own user interfaces on.

> It makes such a difference and it's such a pleasure when you deal with an application that's been developed with an API first mentality. You can interrogate the API calls that the system's own web UI uses. It makes it so much more accessible for us to develop our own applications on top of their tools. It means we can use the API to build much more rich workflows.
>
> DOMINIC BROUARD, VICE MEDIA GROUP

> When you don't see that, you have to ask a lot of questions. And it may be that you just don't use that tool at the end of the day. I've spent too many years going through an API that's an afterthought, and it's just not fun. These days, most people who take it seriously are exposing a REST API that they use themselves.
>
> DAVID KLEE, A+E NETWORKS

One of the biggest changes comes from the use of SaaS, PaaS (Platform as a Service), and IaaS (Infrastructure as a Service) tools in the cloud. They allow the architect to focus on the workflows, data flows, and user experiences, rather than provisioning hardware and applications.

> Has integration changed? Yes and no. If you're architecting from scratch in the cloud, you can do very different integrations now. You can completely ignore the management and delivery of the platform you're using, whereas on prem you've got to look after the kit, and provision networks, firewalls, storage and so on. In the cloud you can be very efficient and streamlined; it's a completely different proposition. But it depends where you're starting from and what you're doing.
>
> ANTONY JOYCE, UKTV

Few projects have the luxury of starting with a clean slate

The reality, of course, is that few projects have the luxury of starting with a clean slate. Often there is a need to integrate legacy applications with newer, cloud native ones, which creates its own set of challenges. (The definition of 'cloud native' is discussed further in *The Cloud for Media*.)

> The biggest integration problem we have is software that wasn't designed to run in the cloud, and bring it into the cloud. Getting it running in the cloud and then trying to integrate it with more modern, microservices based applications.
>
> HANS BAUMGAERTNER, TECHTRIQ

Some of the challenges are the same whether on premise or in the cloud. Whether the interface between two systems is a dedicated cable, a file on disk, a service bus, or a REST API, the key is that the data flowing from one system to another is understood by both.

> In the cloud, you have more options on the infrastructure side, like using managed services or using serverless. But in all cases, the conversation very quickly turns from the technical aspects of how we're going to communicate to discussing what data we need to exchange and how your data maps to my data. And that conversation is the same.
>
> ALEXIS MARTINEZ, IYUNO-SDI

The conversation quickly turns to what data we need to exchange

Indeed, as we discussed in ***The Integration Opportunity***, a strategic approach to integration is often focused on making the right data available to the right teams across an organisation, so they can do more with it.

> The process of integration has changed. You're not having systems being integrated, but having business data being made available.
>
> MICHAEL HARRIT, BBC

To achieve this, there is now an ability to make use of a broader range of tools and techniques from outside our own industry. Of all the benefits offered by software based architectures and the move to the cloud, perhaps this is one of the most significant.

A broad range of tools and techniques is available by looking outside our industry

> One of the problems we have in the media industry is our strong desire that we're special, and we've got to create media specific APIs. But we've developed things that exist in the IT world, and we need to leverage those things.
>
> GORDON CASTLE, DISCOVERY

It's often said that the pace of change in our industry is faster than it's ever been. That likely is true, partly because of this ability to build on the developments of hundreds of industries and hundreds and thousands of developers.

But it's also because software, unlike hardware, can be easily changed and updated. So best practices in integration will continue to evolve. In fact, each individual integration will probably continue to evolve too.

> Just because you've integrated two systems, it's not the same as being able to do everything through that integration. You have to ask, how is it integrated? If you start with one integration to exchange data between two systems in one way, and then another workflow demands more data, then you can expand that integration or in some cases build another integration.
>
> MICHAEL HARRIT, BBC

To support continual development and improvement, you need to get the basics right

For a media company to ensure that it can support this ongoing development and improvement, it needs to have the basics right. It must have an integration strategy, as outlined in ***The Integration Opportunity***. It must have access to the right skills and capabilities, which are discussed in ***Integration Expertise***. And as we'll see in the following sections, it must put in place the right technical foundations.

2 Components of integration

KEY INSIGHTS

- The building blocks of an integrated organisation may be monolithic applications, microservices, SaaS tools, or most likely a combination of them all
- Hot folders are not generally considered to be good practice, but they have some useful applications, such as triggering workflows when content arrives in the cloud
- APIs are at the heart of most integration architectures; dominated by REST APIs due to their flexibility, lightweight nature, and relative ease of implementation
- Message based architectures are a stronger choice for some use cases, especially those involving asynchronous or long running processes
- Most companies need a combination of these approaches, with APIs and message based architectures being especially complementary

To understand the most common architectures for integration, one must first understand the components which comprise them: the software tools being integrated, and the mechanisms by which they are connected.

BUILDING BLOCKS

In *The changing face of integration* above, we outlined the fact that software components being integrated might include both large monolithic applications and small microservices, each deployed on premise or in the customer's cloud, or provided using a SaaS model.

Considering software as simply monolithic or microservices is an oversimplification

However, this differentiation into two groups of monolithic applications and microservices is an oversimplification. There is no single authoritative definition of microservices architecture; it is perhaps more a mindset than a strict set of rules. Some useful definitions include the following:

> Microservices architecture is a style of architecture that defines and creates systems through the use of small independent and self-contained services that align closely with business activities.
>
> THE OPEN GROUP

> A microservice is a service oriented application component that is tightly scoped, strongly encapsulated, loosely coupled, independently deployable and independently scalable.
>
> GARTNER

One can see how these definitions are relative. To a broadcaster's Enterprise Architect, a cloud transcoding service might be considered a 'microservice'. To them, it is a small discrete component in an end to end workflow or larger user facing application. This is especially true if it is provided as SaaS, because the inner workings of the service may not be visible to the customer.

But to the development team for that transcoder, the product may in fact be made of many smaller microservice components, such as an authentication and authorisation service, a job queue, a video processor, and so on.

The definition of microservice is relative; what matters is the function being provided to the user

When discussing integration, what matters most is the function being provided to users and the wider organisation. Signiant's CTO explained this well in ***The Cloud for Media***.

> The real sweet spot is somewhere in the middle. It's pieces of technology that address specific problems, and then are implemented in a cloud native way.
>
> IAN HAMILTON, SIGNIANT

Throughout this report, terms such as product and application are used broadly to refer to a software building block that delivers a particular function at a granular enough level such that it can be integrated into a larger system or workflow.

Whatever the building blocks, however, there are a number of different ways in which they can exchange data, media, and commands. Some of the most common are outlined below.

A HOT TOPIC

One of the oldest and most familiar methods of exchanging information between systems in media workflows is the 'watch folder', or 'hot folder'.

One system places a media file, metadata file, or other asset into a folder on disk. Another system watches that folder for new files, and then triggers some process when it finds one – such as ingesting that media or data.

This has been a widespread and effective pattern for creating media workflows for decades, and is especially prevalent in areas such as post production. In general, however, it is not considered best practice due to the fragility of the implementation.

> Hot folders turn into a nightmare very quickly when you're trying to scale things up. I have not seen an integration solution that involved hot folders that did not break frequently.
>
> ALEXIS MARTINEZ, IYUNO-SDI

Challenges can arise from the fact that the two systems have no knowledge of each other, or of the end to end workflows. There is no status reporting, and no easy way to troubleshoot problems without opening up a file browser.

> A lot of people have implemented things like hot folders, but it becomes problematic when it comes to troubleshooting, because you don't have that feedback loop across the whole supply chain. We prefer to do API level integrations whenever possible.
>
> MIKE FLATHERS, SIGNIANT

Hot folders become problematic when it comes to troubleshooting

In addition, it is increasingly uncommon to implement media workflows by moving media assets from one system to another. Especially in the cloud, it is now commonly considered best practice to bring the application to the media, in accordance with the principle of 'data gravity'.

> As data accumulates there is a greater likelihood that additional services and applications will be attracted to this data. This is the same effect gravity has on objects around a planet.
>
> DAVID McCRORY

None of this is to say that the hot folder is altogether without use, however. In some cases, a simple, pragmatic solution is the most appropriate. For low volume workflows using on premise storage, it may be that a hot folder is sufficient.

In the cloud, however, the hot folder takes on a new meaning. The cloud's object storage platforms generally offer richer functionality such as change notifications that can easily be used to trigger workflows or code execution.

> In the cloud, you can use something like an S3 bucket as a hot folder, and it gives you events, which makes it super simple. Because there's out of the box functionality like Step Functions, you can trigger a workflow quite successfully. For a video file, that might be sufficient integration. If you need to manage metadata through the flow, you will need APIs.
>
> IAN McLAREN, REUTERS

This functionality means that reliability is improved, and monitoring capability is much stronger. As a result, the use of 'hot folders' – or perhaps more accurately cloud storage triggers – is here to stay.

However, they are mostly used to begin workflows for files arriving into the cloud. For communication between applications and services within the cloud, APIs and message based integrations are much more common, and have considerable advantages.

TAKING A REST

APIs are at the heart of modern application integration. They can be used for accessing data, triggering processes, querying status, and more.

They enable application functions, data entities, and even business units to be encapsulated into reusable, composable building blocks. When used to their fullest, they give developers and architects the tools they need to realise new products, capabilities and workflows.

There are a few common types of API, including the simple object access protocol (SOAP). However, by far the most widely adopted approach in modern web and cloud based applications is representational state transfer (REST).

REST APIs are the most widely adopted pattern in modern web and cloud applications

REST was defined in 2000 by **Roy Fielding**, as a flexible set of architectural principles for building APIs using common web technologies, notably HTTP. It is a client/server architecture in which a client system makes a request to a server and receives a response. Requests and responses can use various data formats, with XML and JSON being especially common.

REST interfaces have become widespread because they are lightweight, flexible, and comparatively easy to implement. They offer an excellent way to make synchronous requests between two systems, where an immediate response can be provided. For example, a REST API is an ideal way to query a metadata database in order to retrieve information about a particular media asset, TV episode, or movie.

A REST API is an ideal way to query a metadata database

Nonetheless, like any technical solution, REST APIs have limitations. REST implementations are often not well suited to transferring large amounts of data, so while solutions are available, it is more common to transfer media assets via different means such as dedicated file transfer protocols, or by referencing a file in a shared storage location.

They also may not be the best choice in ultra low latency environments, such as certain playout, live, or editing scenarios, or where structured binary data needs to be transferred – buffer based protocols are more common in these scenarios.

Where the request requires a long running process to be triggered (such as transcoding a large asset), the server generally only responds to confirm that the request has been received. The client must then make subsequent (and often repeated) requests to query the status of the process. This is known as 'polling' for updates. Polling works perfectly well, but may not be the most efficient approach in all cases.

Failure handling can be another area of inefficiency in a REST architecture. If a request fails, then the client has no choice but to retry the request, meaning that error handling logic must be built into each client system.

And the synchronous nature of REST APIs means that if an architecture relies on a real time call to an API for a workflow to continue, the system providing that API must be online. Otherwise, processes will fail and progress will be blocked.

Not only does this increase the importance of high quality error handling, but it can also be a barrier to change management. If the implication of taking a system offline is failed workflows with unknown behaviour, it is very difficult to perform maintenance and upgrades.

For processes that don't need to be synchronous, message based architectures may offer more flexibility.

GETTING THE MESSAGE

The major alternative to synchronous API integrations is a message based architecture.

Those messages can be pushed from one system directly to another, using a method such as a webhook, which allows one system to send HTTP messages to an endpoint offered by another. This method can be used to augment REST APIs by offering a path for systems to return status updates without the need for polling.

Alternatively, a message based architecture may be employed, which decouples the sender and the receiver by means of a message queue or message bus.

A queue can be used to manage messages between a source and destination, using a 'first in, first out' (FIFO) process. In this way, one receiving system processes each message from the queue.

Alternatively, a message bus or broker may be employed to allow messages to be received by multiple independent systems. A publish/subscribe model can be used to facilitate this, in which one system can subscribe to receive messages from another system. The messages will then be delivered to that subscriber, or added to the subscriber's own queue, when they are published.

Consider the news feed of a social network. Each user posts updates as they wish, but you don't have to individually visit each of your friends' pages to see their latest posts. Instead those new updates are added to your incoming 'message queue', usually called a news feed or timeline. You simply view your feed when it's convenient to you, and find all the updates from the sources you're subscribed to.

Consider a message queue like your social network's news feed

Message queue architectures have some advantages, in that they are 'non-blocking'. In other words, the publisher of and recipient of a message are not waiting for each other in real time.

Message buses also allow one published message to be received by multiple subscribers. In an API model, each combination of source and destination would have to be aware of each other, whereas in a message based system the publisher can share a message with an arbitrary or even unknown number of recipients.

It can also be easier to scale message based architectures using horizontal scaling (a concept discussed further in ***Cloud Technology***), by running more instances of the recipient service to process the incoming messages in parallel.

Failure recovery is easier too, because a message queue provides a buffer to ensure that no messages are lost when one system is offline. Publishing or processing of messages can simply resume when the system comes back online, with no data lost.

> Message queues help with migrations and upgrades, because the publishing system doesn't know what's going to be consuming their message, so if we switch that over, they don't know. So it gives you a lot of flexibility if you need to make changes downstream.
>
> ALEXIS MARTINEZ, IYUNO-SDI

Of course, the same buffer also represents the primary downside of message queue based architectures. Latency is added by the process of queuing messages, and that latency can be variable based on the size of the queue.

Message queueing has the downside of introducting latency

Message based architectures may therefore not be the most appropriate choice where it is necessary to have confidence in operating on the very latest information from an external system in real time.

It is also worth noting that message routing and queueing constitutes an additional set of required functionality, potentially adding some extra complexity to the solution. There are excellent, widely used commercial and open source tools for handling this, and each of the major cloud providers offers their own PaaS service for messaging. The latter is often the simplest option for those deploying in the cloud, although the former can offer more control for those who require it.

NO SINGLE SOLUTION

The use of APIs and message queues are not mutually exclusive. Often they are used together to complement each other, or one application may offer both as integration options.

APIs and message queues are not mutually exclusive

> Most cases warrant a combination of APIs and message driven architectures. There might be circumstances where you want to surface certain functionality through an API in addition to message driven options; it really depends on what you're doing. The idea of a one size fits all solution often makes me nervous.
>
> MIKE FLATHERS, SIGNIANT

An application might receive a message on its queue, and follow up with an API call back to the source to retrieve additional data. Or an API integration might be wrapped with a message queue for resilience.

> Our most preferred integration option is event driven architecture with asynchronous APIs. In a microservices architecture each microservice does its job and then broadcasts an event. Other components that are listening to that event pick up and do their processing. Such an architecture also requires a proper API and schema definitions for messages and for retrieving further information.
>
> KRISHNA POTHULA, RTL

Our preferred option is a message driven architecture, and on top of that come APIs

> When you integrate systems that sometimes go down, you have to handle the problems. It's very important that when there's a problem we can review what happened. So when we integrate an external API we always do it with a messaging system so that every call is also messaged, and if there is a problem we can review it and redo the call. We do all our integrations like this, and sometimes it saves us.
>
> SEBASTIEN CREME, NOMALAB

REST APIs and message queues can be mixed with hot folders, or even with legacy APIs that don't conform to web services architectures.

> We commonly use multiple integration models to deliver one workflow. So one might be an API to request content, and another might be an S3 watch folder to actually receive the content.
>
> ASHLEY HORNE, SIMPLESTREAM

> We have certain devices we integrate with which are 'modern' devices that are on sale today, where their API is still based on sending a TCP packet with some binary data. Very much not a standard, modern methodology.
>
> JAMES GIBSON, ORTANA MEDIA GROUP

It is also important to understand that the use of message queues or REST APIs defines certain parameters about the mechanisms by which data is exchanged, but nothing about how the data is structured or what functions the API offers.

In other words, just because two applications have REST APIs doesn't in any way mean that they can interoperate or exchange data.

Just because two applications have APIs doesn't mean they can interoperate and exchange data

There can, therefore, be a perception problem. Those without an understanding of integration or software development may misinterpret what it means for a system to have an API, and hence underestimate the work required in integrating systems.

> Despite their benefits, one of the problems with APIs is that they can create the misconception of a magic box that just enables everything to connect. Often we find that once we get down to the technical detail the integration isn't as simple as was assumed and there's custom code required to get those APIs to connect.
>
> DUNCAN WARRICK, BLACKBIRD

> We can publish these APIs, but just because there are interfaces into your systems, you've still got to have a way of connecting them together. Just because you've got microservices, how do you make all your microservices talk to each other? The challenge is the glue between these fantastic APIs.
>
> NICK WRIGHT, PIXEL POWER

Whether data and commands are exchanged using hot folders, APIs, message queues, or a mix of all three, the architecture of the integration is crucial.

3 Integration architecture

KEY INSIGHTS

- Point to point integration is easy to understand and works well for closely coupled systems, but it becomes unmanageable at scale
- Separating business logic and integration functions into connecting 'glue' reduces the need to customise products, but still suffers from scalability challenges
- A range of integration and orchestration tools is available, from MAMs to media workflow tools, to API gateways and integration platform as a service
- No-code and low-code solutions offer business users the chance to access data and build their own applications, though not all are appropriate for business critical use
- Dedicated integration platforms offer strong governance, security, and monitoring, alongside the ability to reuse integrations and flexibly connect data sources
- For most companies, strategic integration is best achieved using specialist integration tools, though this doesn't rule out pragmatic use of point to point and glue as well

Having considered the components of an integrated organisation, we turn our attention to the architectural patterns by which they are connected together.

With a complex web of applications and systems to be integrated, should they be connected directly, or through a central integration layer? How should data translations and business logic be applied? And can the architecture maximise the potential for future adaptability and new integration opportunities?

STRAIGHT TO THE POINT

The first option to consider is point to point integration. This model is very common in the media industry today.

If system A needs to retrieve data from system B, then an integration is built directly between them. For example, system A is updated to call system B's API.

In many cases commercial tools have pre-built integrations with other products. This can be advantageous as it's simple to deploy and manage.

> A lot of vendors have out of the box integrations. So we are evaluating a lot of the bespoke parts, and consolidating our integrations.
>
> MICHAEL HARRIT, BBC

Point to point integration works well, is simple to understand, and can be quick to implement for any given use case. The challenge is that it may not scale well when more and more new integrations are added.

Point to point integration is simple and effective, but it may not scale well

Imagine a new application is added to the example above. System C must retrieve data from system A, and provide updates to system B. Integrations with each of these systems must now be created.

> Point to point integrations are generally considered quite tactical. For a low volume use case that needs to be done ASAP, and may not have any real strategic value – if there's just two systems and the scope doesn't need to evolve – then by all means point to point is probably the best and the cheapest option. But as point to point evolves, it can become a spaghetti mess.
>
> ABHINEET ASTHANA, MULESOFT

Where two systems are closely integrated, and there is a high degree of confidence that they will not need to expose data or functionality to other systems or business units, then point to point integration is likely to be the best choice.

> There are a few cases where point to point integration makes sense. In post production, for example, if a video needs to be edited in Adobe Premiere, I wouldn't expect an integration layer in between Adobe and a MAM. Point to point is also useful where integrations are a standard part of products, for example an integration between an ad server and an SSP. And some products offer a 'marketplace' with plugins that connect directly to other products.
>
> KRISHNA POTHULA, RTL

Point to point works well for a closed integration between two systems, but it lacks flexibility

However, the lack of flexibility and scalability is certainly a limiting factor. The vendor of each product must create and maintain integrations with any other products that their customers wish to integrate with. And the customer must maintain the web of integrations they have implemented, which may become burdensome as the organisation grows and changes.

> We have a lot of point to point integrations out of necessity. But in general, it makes sense to transition most of those to a service bus approach where you have centralisation for data mapping and orchestration.
>
> ALEXIS MARTINEZ, IYUNO-SDI

Point to point integrations also require one system to be aware of the other. If system A calls the API of system B, then system A must have awareness of system B's API, and must be able to send and receive data in the format that system B requires.

If any data translation or business logic is required, that must be contained within the applications themselves. This isn't always possible, and that's where 'glue' comes in.

A STICKY PROBLEM

Glue code, or simply glue, is a general term used to describe software (or scripts, or other tools) that enable one system to be connected to another.

Glue is a general term for external code that enables one system to be connected to another

Glue enables communication between two systems that have no knowledge of each other. It is often contained in a separate custom component such as a microservice, or serverless function.

The glue might retrieve data from the API of system A, and then send that data to the API of system B, for example.

> 'Glue' to me is about bringing together different services. You take standard APIs and put them together to make the systems talk to each other. So we're not building something entirely new, we're glueing the pieces together, doing the right thing for our organisation.
>
> ANTONY JOYCE, UKTV

This approach can be very useful when one needs to move data between two systems that both have APIs, but do not have a pre-built integration.

The glue might also translate, reformat, or otherwise process data in order to meet the requirements of each application being integrated. Or it might apply business logic that is specific to an individual organisation.

> A lot of what we spend time doing is looking at how we can query information from systems using software that vendors supply or tailored scripts, and then translating the data using our own tools in order to pass back to other systems.
>
> MIKE BRYAN, DB BROADCAST

> One of the biggest challenges is data translation. Everyone uses different data structures, and even different data types.
>
> TROY ENGLISH, ROSS VIDEO

Many media companies prefer to use commercial off the shelf (COTS) products wherever possible, and developing custom glue code might seem counter to this strategy. However, using self contained glue to implement custom logic keeps such customisation out of the commercial products.

Putting custom logic into glue helps avoid the need to customise commercial products

This can be a significant advantage as it makes those commercial products easier to support and maintain for both the vendor and the customer.

> 'Glue' for us is generally anything we use to parse and transform data between our platform and other tools, to save on bespoke, non-repeatable development work.
>
> DUNCAN WARRICK, BLACKBIRD

> We try to avoid writing custom code as much as we can, for our customers. Serverless functions in the cloud have helped a lot with that.
>
> CORY CRABB, RIGHTSLINE

> We use COTS products for our metadata management, customer management, SSO, billing, and payment. And we follow the best-of-the-breed integration approach where all the integration, orchestration, and workflow is implemented using our own microservices.
>
> KRISHNA POTHULA, RTL

For broadcasters with legacy internal systems, it can also be more cost effective and manageable to use glue for integration, rather than add new integrations into the applications themselves.

> It's really expensive to adapt code in our internal systems. And I don't want to do custom integration in a product that I might want to change in the future. So we do like to decouple everything and make it more flexible.
>
> MARTIN RICHARDS, SKY

We don't want to do custom integration in a product; we prefer to decouple everything for more flexibility

Some have found that elements of the glue they've created can be shared and reused, even if specific logic may need to be updated for each implementation.

> Ingestion flows are usually different between content providers and platforms, so what we try to do is reuse the integration we're doing for one customer with others. We do that in the form of open source. So we created an ingestion framework application that is open source, and we can develop plugins for that framework. Then we don't need to do the same work over and over again when we have new clients with the same need.
>
> MAGNUS SVENSSON, EYEVINN

Glue solves a number of practical problems around integrating systems that are not pre-integrated, managing customisation, and implementing business logic. However, it still represents custom code that must be developed and maintained. As such, it still suffers from scalability problems as a company's software infrastructure becomes more extensive, resulting in complexity and creating risks for maintainability.

This is where a dedicated integration platform may be a superior solution.

> When you're trying to build a simple workflow, it can be great to write a script, and maybe just run it in a serverless function. That is your glue. But when a system is getting a bit more integrated, and you're doing more and more, it makes sense to follow a stricter process. You might want to start using a single central API gateway, or perhaps a serverless orchestration system.
>
> DOMINIC BROUARD, VICE MEDIA GROUP

ABSTRACT THINKING

Those looking for a more sophisticated approach to achieving strategic integration capability often turn to a dedicated integration or orchestration platform. The promise is that abstracting APIs and integrations into a specific integration layer offers much more flexibility.

There are many different options available, offering a wide range of different functionality.

For many media companies, the media asset management (MAM) system has long been the default orchestration platform for a range of functions, such as transcoding, quality control (QC), review and approval, packaging, and delivery.

The MAM has long been the default orchestration platform for many media companies

Although this approach can work well within the domain of media processing, it still couples together a number of functions very closely. Mixing asset management with orchestration in the same product can reduce flexibility.

Increasingly, media companies are instead looking to separate out the process of integration and workflow orchestration from those of managing media assets, delivering greater abstraction between components.

> We're using an orchestration platform, and we've built an abstraction layer above it. Because we have the need to get more content onto more platforms, and to do it cheaper. In the past there's been a lot of custom work, and the way to get the cost down is to move from custom to commodity. So we see the abstraction layer as key to getting really fast deployments, and avoiding the need to rewrite code.
>
> ALAN PIMM, MEDIASAAS

The way to get the cost down is to move from custom to commodity

The integration layer can then be employed to perform the capabilities that would otherwise be built in custom glue, such as translating into the formats required for different systems.

> Normalising data through an abstraction layer seems to be the best practice. Because no matter what kind of crazy schema someone wants upstream, you need to be able to normalise that into a format that your platform understands, and to be able to produce whatever format you need downstream.
>
> ASHLEY HORNE, SIMPLESTREAM

One option is to use dedicated media workflow orchestration tools, which offer integrations with common media applications while also providing a central point for management and custom business logic. Some orchestration systems now provide the ability to coordinate infrastructure such as cloud resources too.

> Our media supply chain management system is the sweet spot. It abstracts away things that we don't bring value to, like maintaining infrastructure, spinning up and down computing instances, moving media, and so on. I don't need to know the transcoder's API or the QC system's API, I just know how to get the variables from there into a workflow. It gives us a common place for business logic.
>
> DAVID KLEE, A+E NETWORKS

> We use our orchestration more and more now as a service bus to orchestrate actions, not just around media, but more and more of our clients are provisioning systems.
>
> JAMES GIBSON, ORTANA MEDIA GROUP

Another option is to use general purpose integration tools which are not designed specifically for media. As more and more of a media company's infrastructure uses commodity IT technologies and software, and access to business data of all kinds becomes increasingly important, such tools become more and more relevant.

As media companies use more commodity IT technology, general purpose integration tools become more relevant

An API gateway, for example, can be used to manage the routing of API requests between systems, while also providing a centralised layer for managing security, monitoring, and so on. When compared to simple point to point integration, the introduction of an API gateway can significantly help with management, administration, and scaling of integrations.

For many organisations though, the more extensive functionality of a dedicated integration platform offers an additional flexibility. These functions include not only API management, but also the ability to integrate with message based systems, implement custom business logic, aggregate data, develop internal APIs, and more.

Such platforms are often offered as a cloud based service, known as iPaaS:

> Integration Platform as a Service (iPaaS) is a suite of cloud services enabling development, execution and governance of integration flows connecting any combination of on premises and cloud based processes, services, applications and data within individual or across multiple organisations.
>
> GARTNER

Having a central platform to manage integrations can help manage complexity, and reduce the time and effort required to deploy new tools or create new integrations.

> In a microservices environment, it's not just about development. Observability, monitoring, security are key topics to address. If you are starting from scratch, it takes time to build all the features, and this is where [integration platforms] help speed that up.
>
> KRISHNA POTHULA, RTL

Integration platforms can reduce the time required to build monitoring, observability, security, and more

When it comes to integration security, it is highly advantageous to have a central place to manage user authentication, role based authorisation, and policy application. This can be achieved using dedicated services, such as those provided by cloud providers, or through an integration platform.

> It's useful to have things like role based access control in place. As you quickly stand up new APIs or new integrations, then you can at a very granular level provide access rights, and turn them off very quickly as well.
>
> KRISTAN BULLETT, HUMANS NOT ROBOTS

Integration platforms also provide logging and monitoring, providing both audit capability and status monitoring.

> We're dealing with increasingly distributed workflows. So things like auditing of what's happening really matter.
>
> PAUL RANDALL, AVID

> Observability is important for integrations. It helps in understanding if all is well, or where a failure might have occurred. When integrations are orchestrated in a workflow, the combination gives a lot of control and confidence.
>
> KRISHNA POTHULA, RTL

> Really you have to understand status, and fault recovery. In the cloud, the APIs are often more mature, the error messages are generally a lot more helpful. But there's a mistake in assuming success. You need to assume failure and ensure that the system is able to react to handle it.
>
> JAMES GIBSON, ORTANA MEDIA GROUP

Governance isn't just about security, however. As integrations across an organisation get more complex, it becomes increasingly important to understand data flows, and ensure that data integrity is maintained across the organisation. It can help to have visibility across integrations, and a central place to apply data governance.

> In addition to workflow governance, we find that data governance is a key part of it. As more systems get integrated and you're mapping data, there needs to be governance around that data to avoid changes being made that break other integrations or future requirements that will be built.
>
> ALEXIS MARTINEZ, IYUNO-SDI

General purpose integration platforms and media specific orchestration platforms both provide tools to connect together data sources and orchestrate workflows using the different applications that have been integrated. The advantage is that business logic can be applied within the platform, usually using configuration or scripting, rather than in completely custom software components. This has advantages for management and maintainability.

> We should be aiming as an industry for configuration over customisation, both in the way we build our individual products and in the way that we glue them together. You want to lower the touch that's required, so you don't lock someone into how things look today.
>
> PAUL RANDALL, AVID

We should be aiming as an industry for configuration over customisation

Some platforms often go a step further, allowing technology teams to create new APIs which mix data or functionality from multiple sources into dedicated endpoints to meet specific needs. These additional layers of abstraction mean that a user facing application can consume an API dedicated to its needs, while the data is accessed and processed by multiple systems in the background, for example. This enables APIs to be built to precisely meet business or user needs, ensuring they deliver maximum value.

None of this is to say that an orchestration platform has to be all encompassing, however. Many media companies use a combination of point to point integrations for tightly coupled systems, with a media orchestration tool for video specific workflows, and/or a broader integration platform for connecting business systems and other data sources.

A single integration platform need not be all encompassing

> We define which pieces of information need to run through our integration platform. Not everything needs to. So for example in the production management system there's a lot of reporting that is only needed in that team and that system. The use case is very direct. But there are other pieces of information that are needed across the company, and kept for longer. Those are exposed in the data layer.
>
> DANIEL ELIAS, VICE MEDIA GROUP

> We shouldn't think about one platform to rule them all. It's about supporting an ecosystem, and not being precious about where we sit in that ecosystem.
>
> BRIE PEGUM, IMAGEN

DEMOCRATISING INTEGRATION

As discussed in ***The Integration Opportunity***, a strategic approach to integration can open up possibilities to reuse integrations, connect together data from different systems, and create custom applications.

This is usually done with little or no coding required, and so it is generally referred to as 'low-code' or 'no-code' development.

One way to achieve this is through dedicated SaaS no-code integration products such as Airtable, IFTTT, and Zapier. They offer the ability to start connecting systems together quickly, with many out of the box integrations to popular SaaS services, enabling business users to work with data from multiple systems quickly and easily.

No-code solutions enable business users to reuse integrations, connect data, and create custom applications

Many of our contributors use these simple integration tools for testing and prototyping, or for internal tooling such as project management.

> We use a lot of low-code solutions to do prototyping. It helps speed things up a lot because you don't have to build out a database and a user interface, and then we hook out to our API gateway to connect to other systems. It's offered us a launchpad to get going very quickly.
>
> ARRAN CORBETT, FOOENGINE

> More and more, my teams use no-code platforms that we can wire up to a database, and they have data that they can play with, prototype interactions, and test things.
>
> MICHELE MEMOLI, 100 SHAPES

However, they also cautioned against relying on such tools for business critical workflows.

> Within the company, we use a ton of no-code tools, but the barrier is "is this mission critical?" Going into production and serving customers almost never happens; I don't think it ever has.
>
> ADAM BROWN, MUX

Simple no-code platforms are great for prototyping, but not for business critical workflows

> If you're trying to make a low-code solution that can be the centralised hub of connectivity, then you need to deal with workflow governance. How do you hand out permissions and rights to workflows? What do you do with security?
>
> HANS BAUMGAERTNER, TECHTRIQ

Considerations of security, access management, visibility, and monitoring can lead some to avoid low-code and no-code tools altogether, in favour of stricter software development methodologies.

But many of these concerns can also be addressed through the use of an enterprise integration platform. Indeed, many platforms offer their own low-code or no-code methods for building custom data views and applications, all within that framework of governance, security, and monitoring.

> Often, developers hate low-code. But if you're working with enterprise architects, or business analysts, they generally prefer low-code solutions because they can prototype things a lot more quickly. And there are some low-code solutions out there that still give you granularity when configuring your endpoints, your connections, your workflows. So to some extent I think developers can get on board with these.
>
> ABHINEET ASTHANA, MULESOFT

In the end, a trade off is made between offering complete flexibility to users, and maintaining the integrity of systems and data.

> Obviously the governance can slow you down with changes, so there is a bit of resistance. But everyone's paid the price of people going rogue in the past. We've experienced systems not working properly until something is fixed, so everyone gets it.
>
> DANIEL ELIAS, VICE MEDIA GROUP

The availability of no-code tools can have a dramatic effect on organisational dynamics. Giving business users self service access to data points and even enabling them to build their own applications changes the cultural meeting point between technology and business teams.

Availability of no-code tools changes the cultural meeting point between technology and business teams

> I'm a big believer in no-code tools in order to empower your business, to enable operations people to do work without needing a developer to facilitate. Sometimes we put together the information that's important for a process, and then get that data out to run the more complex media processing work that really does require a development team.
>
> DAVID KLEE, A+E NETWORKS

Of course, the balance between no-code and code based tools will also depend on the skills profile available within the organisation, as discussed further in *Integration Expertise*.

> The skill sets required are changing. We see companies having success with low-code solutions, which can provide faster time-to-value and agility, but does trade off some control. Building from scratch requires a different level of skill, but may be warranted when there are unique requirements. In general, we're seeing less of that which is good for the industry in my opinion.
>
> JON FINEGOLD, SIGNIANT

MAKING THE RIGHT CHOICE

Just as with the different options for querying and messaging discussed in the previous chapter, no single architectural option is right for every use case.

The majority of our contributors agreed that using an abstraction layer, such as a dedicated integration platform, is the best approach for strategic integration.

An abstraction layer such as an integration platform is the best approach for strategic integration

> Yes, there is a right answer. You have to abstract it. If the only thing you ever need to do is to have these two vendors work together, that's fine. If that's all you ever need to do. But if you can't say yes definitively to that, then an abstraction layer is essential.
>
> GORDON CASTLE, DISCOVERY

Sometimes, however, pragmatic business needs necessitate a different approach.

> Sometimes business needs can't wait for the ideal technology solution. Inevitably there will be integrations that we may want to route through an integration platform in the future, but we're not there yet and we need to make the integration happen right now anyway.
>
> DANIEL ELIAS, VICE MEDIA GROUP

Ultimately, the right choice for any organisation is the one that most effectively helps them to reach their business goals. The decision to adopt a dedicated integration platform, or to use point to point integrations or glue, will be driven by factors such as the need to expose and reuse business data, and the need for centralised governance and monitoring.

For those trying to achieve business agility, however, the most compelling reason to adopt an integration platform may be the possibility to empower operational and technical teams to adapt to constant change.

> Point to point is probably not the way forward. There's a wide range of orchestration tools out there, and they all have lots of integrations. But the important thing is to make it easily accessible for customers and operators. Enabling them to write these integrations or glue themselves.
>
> CHRISTOPH JURKUHN, HISCALE

> I think we've moved on. The key things we should consider are what we expose to the business compared to what we do in the technology department.
>
> MICHAEL HARRIT, BBC

The key thing to consider is what we expose to the business compared to what we do in the technology department

CONCLUSION

An overflowing toolbox

Integration and orchestration in media used to be simply about moving media files between processing systems. Today, however, media organisations have much broader integration challenges encompassing business applications, data, and user experience.

> None of my problems are related to media. There are vast armies of people working on media problems. 99% of my problems are everything else.
>
> CRAIG SEIDEL, PIXELOGIC

Integration in media used to be about moving media files between systems, but now it's much broader

The world of integration technologies available to solve these problems can at first seem overwhelming. Monolithic applications, microservices, and SaaS tools might need to be connected together, with a mixture of APIs, message buses, and business logic. Some might connect directly together, while others might need custom glue.

Amid all this, the needs for strong security, excellent monitoring, and good governance are more important than they've ever been. And constantly evolving business needs demand a high degree of flexibility in the way integrations are built.

It's for these reasons that most companies are best served by moving away from point to point integrations, instead abstracting the integration and business logic from the individual applications.

> The most efficient, cost effective, and agile next-generation media technology stacks will comprise off-the-shelf functional building block products tied together with lightweight logic.
>
> MARGARET CRAIG, SIGNIANT

Some will create custom microservices for the business logic, and employ specialist solutions for API security, monitoring, workflow orchestration, and so on. The major cloud providers offer all of these components as part of their platform as a service offerings.

Other organisations will find that a dedicated integration platform offers a more convenient and manageable way to implement these functions, while also offering more capabilities to a broader range of business users.

Meeting the needs of those users is paramount, of course. That means providing the right functionality and data at the right time, no matter what underlying systems provide those functions and data.

> The key thing is user experience. It's about what people are doing, not about the systems.
>
> DANIEL ELIAS, VICE MEDIA GROUP

It's about what people are doing, not about the systems

The Integration Opportunity identified the benefits of a considered and strategic approach to integration. But being strategic doesn't mean being dogmatic. The truth is that every media company will necessarily have a mix of integration types, because no architecture suits every use case.

But by considering the right layers of abstraction, with appropriate data management, workflow orchestration, and governance, a more flexible integration infrastructure can be built. And that will help enable the business agility that media companies now require.

Making Integration Work was researched and authored by **Rowan de Pomerai**, and the reports were designed by **Vlad Cohen**.

The workshops were organised by **Abdul Hakim** and **Anh Mao**. Additional support and accompanying materials were provided by **Edward Qualtrough**.

About the DPP

The DPP is the media industry's business network. It is a not-for-profit company with an international membership that spans the whole media supply chain, covering global technology companies, production companies, digital agencies, suppliers, service providers, post production facilities, online platforms, broadcasters, distributors and not-for-profit organisations. The DPP harnesses the collective intelligence of its membership to generate insight, enable change and create market opportunities. For more information, or to enquire about membership visit

thedpp.com

About Vubiquity

Vubiquity, an Amdocs company (NASDAQ: DOX), is a global media and entertainment technologies, products, and services provider. Vubiquity provides established expertise and innovation across the whole media supply chain from content processing and distribution; direct-to-consumer streaming and monetization; through to systems integration and professional services. With an extensive range of core competencies that can be rapidly and flexibly deployed, the world's leading content owners and service providers trust Vubiquity to power their consumer-facing, entertainment experiences.