SONY

nevion

Whitepaper:

An easy guide to DMF and MXL

Aug 2026

Contents

About this whitepaper

This whitepaper provides a high-level, accessible introduction to the EBU Dynamic Media Facility (DMF), the Media Exchange Layer (MXL) and the work of the JT-DMF (Joint Taskforce - Dynamic Media Facility). It is intended for readers who want to understand the key concepts and the role these technologies play in modern broadcast infrastructures without going into detailed technical specifications.

The paper explains how broadcast facilities are evolving from fixed, hardware-centric architectures to software-defined environments running on shared commercial off-the-shelf (COTS) or cloud infrastructure. It introduces in generic terms the DMF as an architectural framework for deploying, orchestrating and managing media applications consistently across these environments. It also provides a brief overview of MXL, which enables efficient media exchange between software components.

Readers seeking a more detailed technical understanding of either technology can refer to the dedicated documentation available on the EBU website.

Broadcast infrastructure is **changing**

Broadcasting has always adopted new technologies when these deliver clear operational or economic benefits – as long as they preserve the core requirements of professional media, i.e. quality, reliability and deterministic performance.

The most recent industry shift is from dedicated hardware appliances to software-based functions running on standard computing platforms or the cloud. This is largely driven by the desire to have more flexible operations and cost models (e.g. OPEX-based rather than CAPEX) and accelerated by the fact that COTS servers' performance is now comparable to dedicated hardware.

This shift creates more agile facilities, but also new challenges: applications must be deployed, monitored, updated and orchestrated; compute resources must be allocated efficiently; and components from different vendors must exchange control data and media reliably.

Traditional integration and interoperability models are poorly suited to environments where applications can be created, moved or removed dynamically. The EBU Dynamic Media Facility (DMF) addresses this by defining an architectural framework for deploying and managing software-based media applications across shared infrastructure. Within it, the Media Exchange Layer (MXL) provides an efficient way for software components to exchange media. Of course, established standards such as SMPTE ST 2110 remain important for interoperability with external systems.

It is worth noting that the growth of software-based solutions will not mean the end of dedicated hardware. The latter is still needed where specialized processing or direct access to high-performance interfaces is required. In practice, modern broadcast facilities are likely to combine hardware appliances with software-based processing – be it on-premises or in the cloud.

Technology explainers

This document includes technology explainers of some of the computing concepts that are useful for understanding DMF and MXL. While some readers will already be familiar with these ideas, others may not be. The intention of these explainers is to help the more casual reader understand better the key concepts.

The Dynamic Media Facility (DMF)

Why is DMF needed?

Traditional facilities are built around relatively stable physical devices, fixed connections, defined standards, common timing references and established operational procedures.

A software-defined facility is more dynamic: applications may start, stop or move between servers as production needs change, while multiple vendors share the same infrastructure. This requires more than replacing hardware with software: it requires an architectural framework for coordinating the whole system.

The Dynamic Media Facility (DMF) provides that framework.

The DMF is backed by:

- The EBU Technology & Innovation group
- The North American Broadcasters Association (NABA)
- The Linux Foundation (specifically for MXL)
- The Advanced Media Workflow Association (AMWA)
- Several broadcasters and media networks including the BBC, CBC/Radio-Canada, the Olympic Broadcasting Services (OBS), SVT, France TV, RTÉ and SRG SSR
- Many vendors including Sony, Nevion, Grass Valley, Lawo, Riedel, Appear, Amazon Web Services (AWS) and NVIDIA

An architectural framework

DMF defines an architectural approach for building software-defined media facilities (whether ground-based, cloud-based or hybrid), allowing applications from different vendors to run together on shared infrastructure while remaining portable, interoperable and manageable.

Rather than defining how applications are built internally, DMF focuses on how they are deployed, discovered, orchestrated and connected consistently.

On COTS or cloud

Modern media applications can run on COTS hardware or in the cloud. Although often presented as different models, both rely on similar computing technologies.

COTS deployments use standard servers, storage and networking in a broadcaster's facility or data center, while cloud deployments provide comparable infrastructure as a managed service.

Well-designed software should be able to operate in either environment with limited changes.

While the DMF puts a lot of emphasis on COTS deployment, its principles apply equally to cloud deployments.

Thinking in resources rather than devices

Perhaps the biggest conceptual change introduced by DMF is a shift in how broadcast infrastructure is viewed.

Traditionally, production workflows involve connecting pieces of equipment with a defined location and dedicated functions.

Within a Dynamic Media Facility, functions become software applications that consume shared resources. Instead of allocating physical devices to a production, the facility allocates computing capacity, memory, storage and networking resources on which the required applications can run. Familiar functions, like a video switcher, become software services (or “media functions” in DMF terminology) that can be deployed wherever suitable resources are available.

For operational staff, this provides greater flexibility: capacity can be increased by adding software instances rather than hardware, applications can be updated independently of servers and resources can be reassigned as priorities change. A greater efficiency and utilization of resources can be achieved because, unlike many hardware devices, they can run a variety of functionality and be used for different productions throughout the day.

Orchestration

A modern production may involve dozens or even hundreds of software services distributed across multiple servers, so manual configuration quickly becomes impractical.

DMF therefore assumes a high degree of orchestration and automation.

An orchestration system deploys applications, allocates resources, monitors health and coordinates interactions. Operators define the required production workflow, and the system decides where applications run and how they connect.

This is similar to resource management in modern IP networks, where engineers define the desired connectivity, and the orchestration determines how traffic is routed.

Media exchange

In a workflow, software components must also exchange media efficiently with other devices and software components.

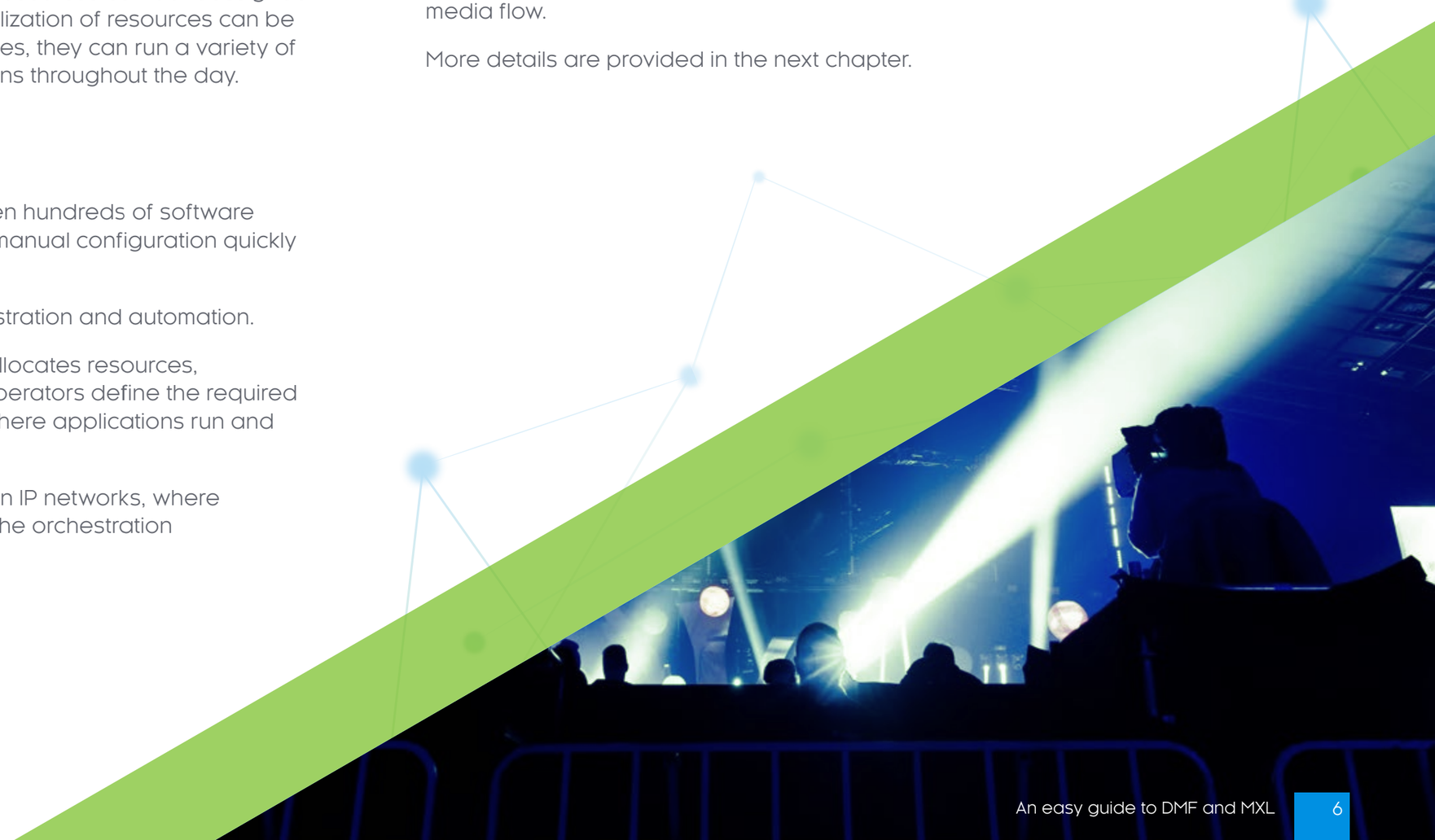
When communicating with external equipment, applications continue to use established broadcast standards such as SMPTE ST 2110 and NMOS where appropriate.

Within software-defined infrastructure, components may run on the same server, nearby servers or a tightly coupled compute environment, where conventional network transport can add unnecessary overheads.

DMF therefore includes the Media eXchange Layer (MXL), which provides a very efficient way for participating software components to exchange media, i.e. video, audio and meta-data.

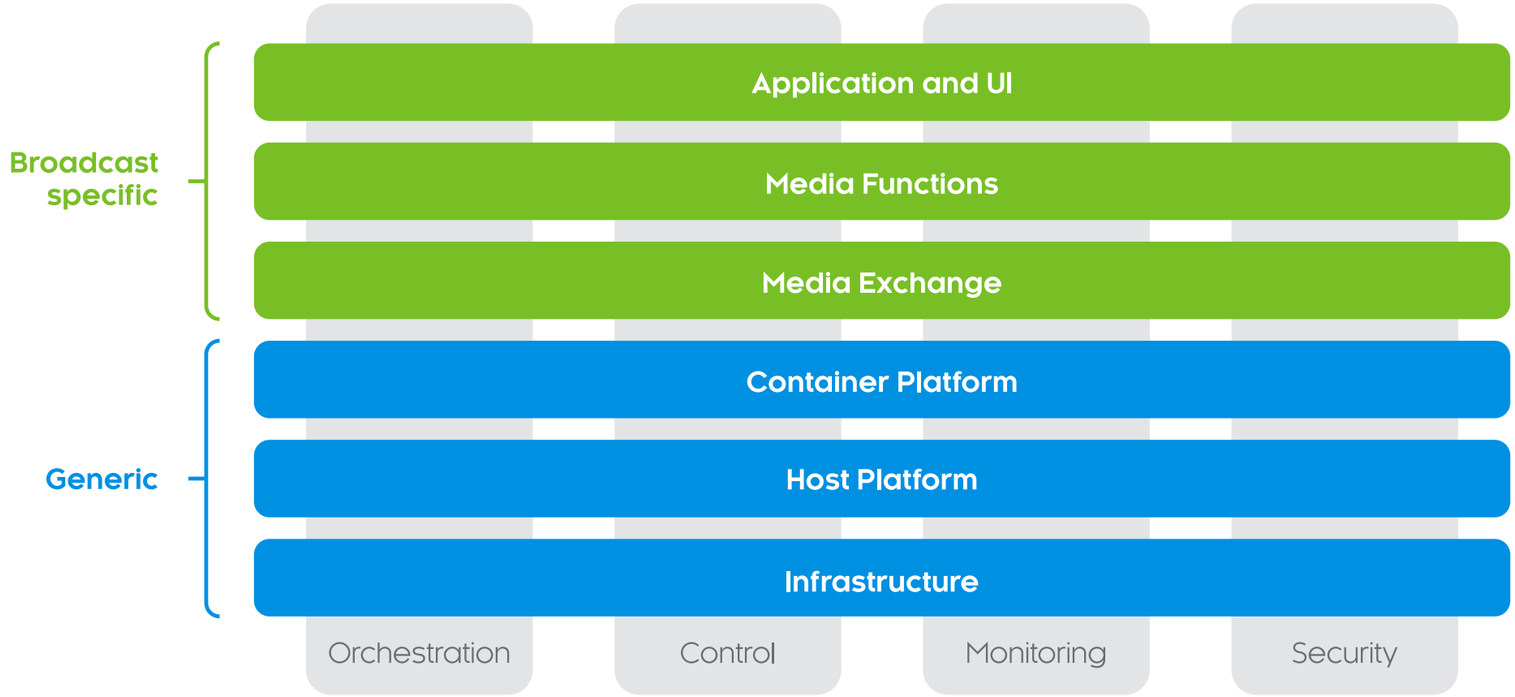
MXL complements rather than replaces established broadcast transport standards, allowing applications to use the best mechanism for each media flow.

More details are provided in the next chapter.



A short overview of the DMF model

The DMF architecture operates like a matrix: the horizontal layers describe how the software functionality is delivered, while the vertical columns describe aspects that span the entire horizontal layers.



The 6 horizontal layers

The six horizontal layers describe the main functional parts of the system.

The first three layers (starting from the bottom) are generic computing layers, i.e. could apply to any industry:

- **Infrastructure:** The physical equipment the system runs on, such as servers, networks, storage and timing systems.
- **Host platform:** The software environment on the servers, including the operating system and virtualization tools (e.g. virtual machines) that allow the hardware to be shared.
- **Container platform:** The layer that runs and manages software applications packaged as containers, typically using tools such as Kubernetes.

The next three layers of the DMF model are specific to broadcast and media production:
[See next page for bare metal, Virtual machines and container technology explainers]

- **Media Exchange Layer:** The high-performance layer that helps software components exchange media efficiently (covered in the next chapter).
- **Media Functions Layer:** The software functions that perform broadcast tasks, such as encoding, decoding, audio/video processing and multiviewer generation.
- **Application and User Interface Layer:** The layer, which includes the GUI, where operators control workflows, monitor production and interact with the system.

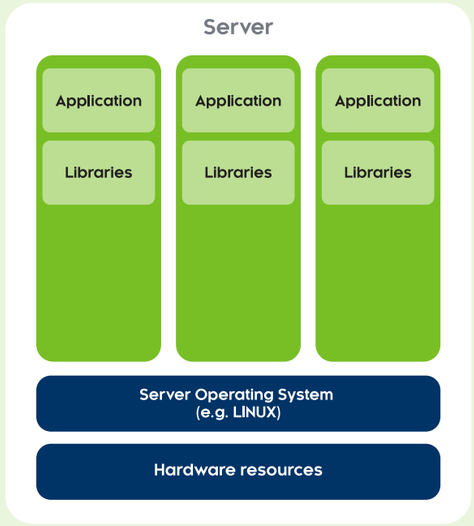
The 4 vertical columns

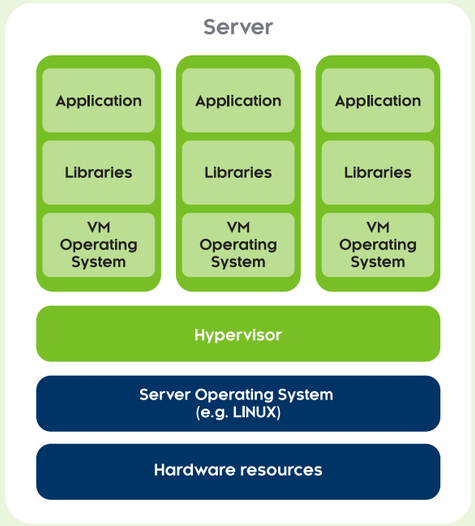
The four vertical columns of the DMF model represent concerns that apply to all horizontal layers, and that ensure system cohesion, performance and reliability:

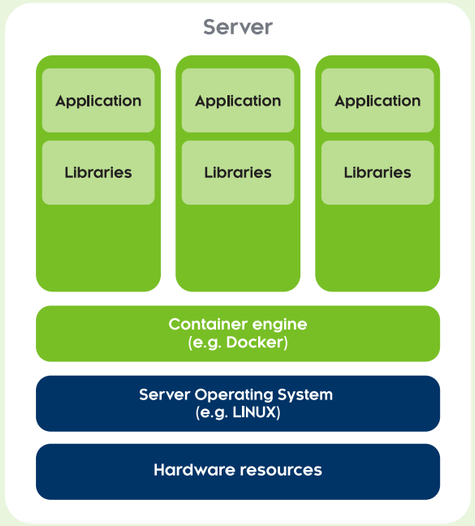
- **Orchestration:** Automates the deployment, scaling and lifecycle management of media workloads.
- **Control:** Manages the configuration and operation of media infrastructure and services.
- **Monitoring:** Collects telemetry to monitor performance, health and operational status.
- **Security:** Protects media systems through identity, access control and continuous security enforcement.

Technology explainer – bare-metal, VMs and containers

Several software deployment models are commonly encountered in modern broadcast systems.







Bare-metal deployment

In a bare-metal deployment, an application runs directly on a physical server and its operating system. This gives the application direct access to hardware resources (e.g. memory, network cards, CPU/GPU), potentially delivering the highest performance.
The trade-off is lower flexibility: applications are more closely tied to specific servers, harder to move or scale dynamically, and can be more complex to maintain across large facilities.

Virtual machines

As facilities grow, running multiple independent software environments on one physical server becomes useful. Virtual machines (VMs) are one way to do this.

A VM emulates a complete computer, including its own operating system (which may differ from the server's). A hypervisor running on the server allocates hardware resources to each VM, allowing several isolated systems to share the same hardware. VMs provide strong application separation, but each VM carries the overhead of a full operating system.

Containers

Many applications now use containers. Unlike VMs, containers share the host operating system while keeping applications isolated. Each container packages an application with its required libraries and dependencies, so it can run consistently across servers. Containers can run directly on the host operating system or inside VMs.

For broadcasters, containers simplify deployment and maintenance. A media processing application can be installed, updated or moved using the same package, reducing configuration differences between systems.

A foundation for future broadcast facilities

The Dynamic Media Facility does not define a new production workflow, nor does it replace the engineering principles that underpin professional broadcasting, including established standards like SDI and SMPTE ST2110.

Instead, it provides a reference architecture for software-defined broadcasting. It is designed to align the media industry (from media organizations, system integrators to developers), as it transitions media production from rigid, hardware-dependent workflows to flexible, software-first environments.

More information

It is outside the scope of this document to describe the DMF in any detail. More information can be found on the EBU website:

https://tech.ebu.ch/publications/white-paper-2026-04-15

The Media Exchange Layer (MXL)

DMF provides a framework for deploying and managing software applications. These applications also need to exchange media.

Standards such as SMPTE ST 2110 are designed for media transport between devices connected via an IP network. COTS servers are of course usually connected via IP, so those transport protocols can be used.

That said, in software-defined environments, media often moves between applications on the same server, nearby servers or a tightly integrated computing platform. Treating each application as a separate network endpoint can add unnecessary overhead and complexity.

The Media eXchange Layer (MXL) provides a common abstraction for media exchange between software applications in a Dynamic Media Facility.

In short, what is MXL?

MXL is an open-source software project and SDK (software development kit) designed to standardize how live video, audio and metadata are shared between software applications in modern, cloud-based and containerized broadcasting environments.

A common media interface

MXL lets applications exchange media efficiently through a common interface.

Applications do not need to know whether media comes from the same container, another server or elsewhere in the facility.

They use the Media Exchange Layer through a common API, and MXL determines how the media is transported between software applications.

This simplifies development and improves application portability across deployment environments. It also allows transport mechanisms to evolve while keeping a stable interface for media applications.

Selecting the most appropriate transport

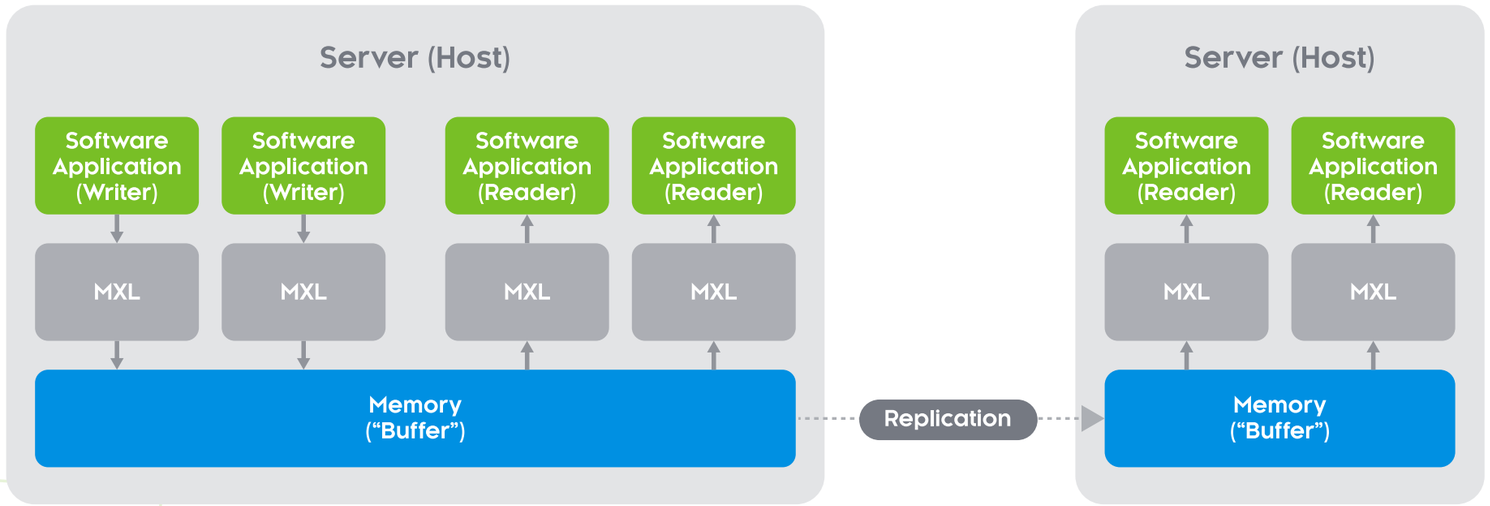
For interoperability with external equipment, SMPTE ST 2110 remains the natural choice for exchanging professional media across broadcast networks.

Within a Dynamic Media Facility, other options are possible.

Applications on the same server can exchange media through shared memory (DMA – Direct Memory Access - see next page), avoiding unnecessary network processing and additional memory copies for each application.

Where software applications are deployed across different servers, Remote Direct Memory Access (RDMA - see next page) provides a more efficient media exchange method than conventional protocols.

Neither of these technologies are new, but MXL provides an abstraction layer that standardizes the way media software applications use them. MXL also relieves the applications of having to work out whether DMA or RDMA should be used.

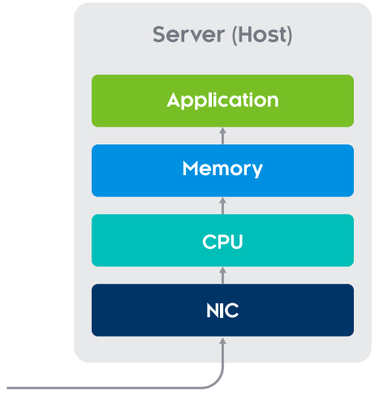


Technology explainer – DMA and RDMA

When applications run within the same software-defined infrastructure, such as the same server or server cluster, conventional network transport can add unnecessary processing, memory copies and latency. This is particularly important in live production, where the amount of data transported (e.g. video frames) is huge.

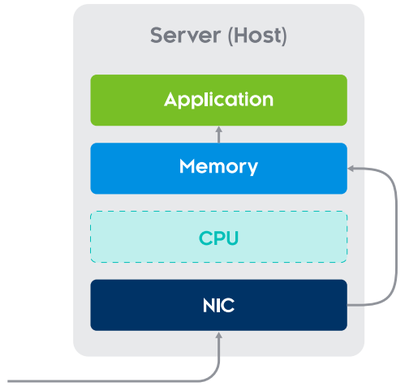
Conventional transport

With conventional transport, a video frame arriving on a COTS server's network card (NIC) is copied into memory by the CPU before the application can process it, adding to the CPU's load.



DMA – Direct Memory Access

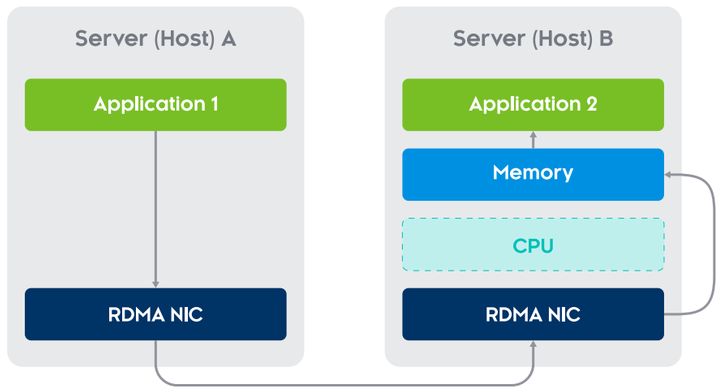
DMA lets hardware, in this case the NIC, write data (e.g. video frame) directly into memory without the CPU needing to do the copying. This is faster and frees CPU resources for other tasks.



RDMA – Remote Direct Memory Access

RDMA extends DMA across a network, allowing an application running on a server to place the data (e.g. video frame) directly into the memory of another server for the receiving application to access.

A network protocol typically used to deliver RDMA is RoCEv2 (RDMA over Converged Ethernet version 2).



Note:

- In the examples above, content is arriving via a NIC. Of course, where applications run on the same server, they can exchange content directly via memory (DMA) rather than having to transit in and out of the NIC.
- MXL will typically use the open-source libfabric networking library to handle RDMA.

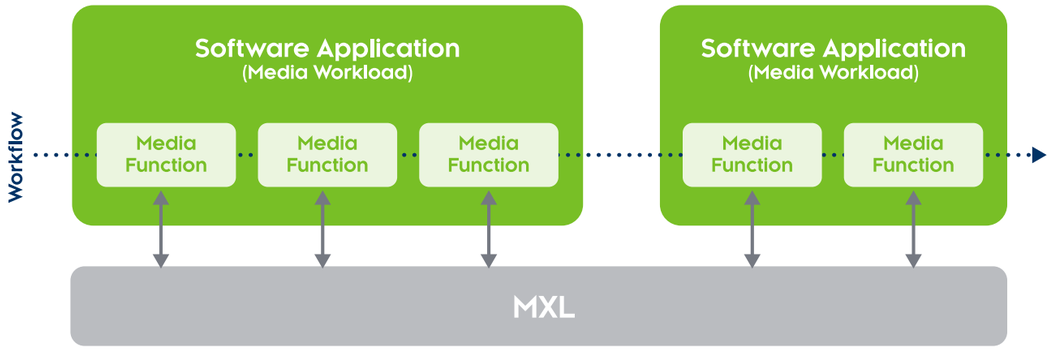
Decoupling applications

Because applications interact through MXL rather than directly with one another, they are less tightly coupled.

A media processing application does not need to know the implementation or location of the applications it exchanges media with. If each application uses the same MXL interface, they can be developed independently and deployed flexibly.

This separation means that software can be developed in modules, which can be linked to achieve the desired functionality. For example, separate media functions for adaption, encoding/decoding and format transformation can be linked to perform some media node functionality. In multi-vendor environments, new applications can also be added without changing existing ones.

Applications can also be moved by the orchestration system without changing their media processing behavior. Whether they use shared memory, RDMA or conventional transport is handled by the infrastructure.



Complementing existing broadcast standards

MXL complements rather than replaces existing broadcast standards.

SMPTE ST 2110 remains preferred for media exchange with external broadcast equipment and systems outside a Dynamic Media Facility. Existing timing, synchronization and discovery mechanisms continue to support professional IP production.

MXL (and the work of the JT-DMF – see later) focus on software-to-software communication, preserving interoperability and delivering the performance required by software-defined media facilities.

Looking ahead

As media processing becomes more software-defined, computing technologies such as faster interconnects, new memory architectures and hardware acceleration (e.g. GPU or FPGA-based) will continue to improve media transport.

Because MXL separates the application interface from the transport mechanism, these advances can be adopted without changing how applications interact.

MXL therefore provides both efficient media exchange today and a stable foundation for future software-defined broadcast systems.

More information

It is outside the scope of this document to describe the MXL in any detail. More information can be found on the EBU website: https://tech.ebu.ch/dmf/mxl

High-level information can also be found in the DMF whitepaper: https://tech.ebu.ch/publications/white-paper-2026-04-15

JT-DMF working groups

Through the DMF, the EBU has produced architectural models, terminology, use cases and reference architectures for software-based production. However, this work has deliberately stopped short of defining detailed interoperability standards, aside from MXL. This obviously limits its practical implementation.

For example, the DMF states that “applications should be able to request compute resources”, but it does not specify the mechanisms required to make those resources visible to applications, and how they can be reserved, released or interacted with.

This is where the JT-DMF comes in.

Set up as an EBU and AMWA joint taskforce, the JT-DMF’s aim is to provide the “how” to the DMF’s “what”.

At the time of writing (August 2026), the JT-DMF consists of 4 working groups, responding to clear standardizing requirements.

About AMWA

The Advanced Media Workflow Association (AMWA) develops open specifications, including the Networked Media Open Specifications (NMOS) suite, to help broadcast and media systems from different manufacturers discover, connect and operate together reliably.

End-to-end synchronization working group

Synchronizing media signals is an essential part of live production. The End-to-End Synchronisation Working Group is defining common approaches for timing, latency management, timestamping and synchronisation so that video, audio and metadata remain correctly aligned as they go through multiple software services, compute platforms and network paths.

Compute resources management working group

The Compute Resources Management Working Group is developing methods for managing the pools of compute resources (e.g. shared CPUs, GPUs, FPGAs and other accelerators) that underpin a Dynamic Media Facility. The group is defining how applications can discover, reserve, use, monitor and release shared resources efficiently and automatically.

Flow connection working group

The Flow Connection Working Group is defining interoperable ways to discover, establish and manage media flows in a dynamic facility. Building on technologies such as NMOS, it enables applications and services from different vendors to connect automatically as software services are created, changed or removed.

Business working group

Whereas the other working groups focus on technical aspects of the DMF, the Business Working Group considers how broadcasters, vendors and service providers can successfully transition from traditional infrastructure to software-defined facilities. Its work includes identifying business drivers, developing migration strategies, assessing operational impacts and promoting common terminology and use cases that help align technical developments with real-world industry needs.

New working groups

New groups will be formed to tackle specific issues as they arise, for example security.

Timetable

Like with NMOS, JT-DMF is a long-term industry initiative supporting the gradual evolution of broadcast facilities towards software-defined, dynamically orchestrated infrastructures. The task force is producing interoperable specifications and best practices incrementally, allowing broadcasters and vendors to adopt Dynamic Media Facility concepts as the technology and operational maturity of the industry evolves.

Conclusion

The transition to software-defined broadcasting is not a simple replacement of hardware with software. It requires new architectural thinking, careful orchestration and efficient media exchange between applications. DMF and MXL address these needs by combining the flexibility of modern computing with the reliability and performance expected in professional live production. They do not replace existing broadcast standards, but extend the broadcast technology toolbox so that facilities can evolve at their own pace, using the right mix of hardware, software, COTS infrastructure and cloud services.

Appendix A: Networked Live and DMF/MXL

Sony's Networked Live is an open ecosystem of integrated solutions, products, services and partners that simplifies the transition to more agile and cost-effective hybrid (ground/cloud) operations. The solutions are designed to work seamlessly with existing equipment, familiar user-interfaces and third-party technologies.

Below are some of the Networked Live products that leverage DMF/MXL.

MOXELA

Nevion MOXELA is a software-native processing platform for COTS and cloud that is designed to perform a variety of real-time functions for a wide range of applications including contribution, GCCG (ground-to-cloud) transport, Software Defined Broadcast (SDB) and cloud production.

The MOXELA platform is designed to prevent vendor lock-in by supporting existing and emerging standards at every level of the software stack – from support for multiple hardware and operating systems, through containerization and onto the industry-specific Dynamic Media Facility (DMF) architecture, including support for MXL (Media eXchange Layer).

For more information:
https://nevion.com/moxela/

M2L-X

M2L-X is a software switcher for cloud and on-prem production environments which is especially suited for sports and event live production.

M2L-X has been developed for the Linux (Ubuntu) operating system and the Kubernetes container platform, which are the recommended technologies for DMF. Sony is now evolving M2L-X further to become fully compliant with the DMF and MXL architecture.

For more information:
https://pro.sony/products/video-switchers/m2l-x

VideoIPath

As mentioned, DMF assumes a high degree of orchestration and automation. Nevion VideoIPath is a media orchestration platform that is at the core of Networked Live's software-based offering.

VideoIPath uniquely combines broadcast control, network orchestration, cloud orchestration and monitoring capabilities. It is ideally positioned to enable hybrid ground and cloud workflows that combine hardware and DMF-based software solutions.

For more information:
https://nevion.com/videoipath/

Nevion is a Sony Group Company

pro.sony/
networked-live

nevion.com