Pixop FILE

Cloud-Based Video Processing and Enhancement for File-Based Media Workflows

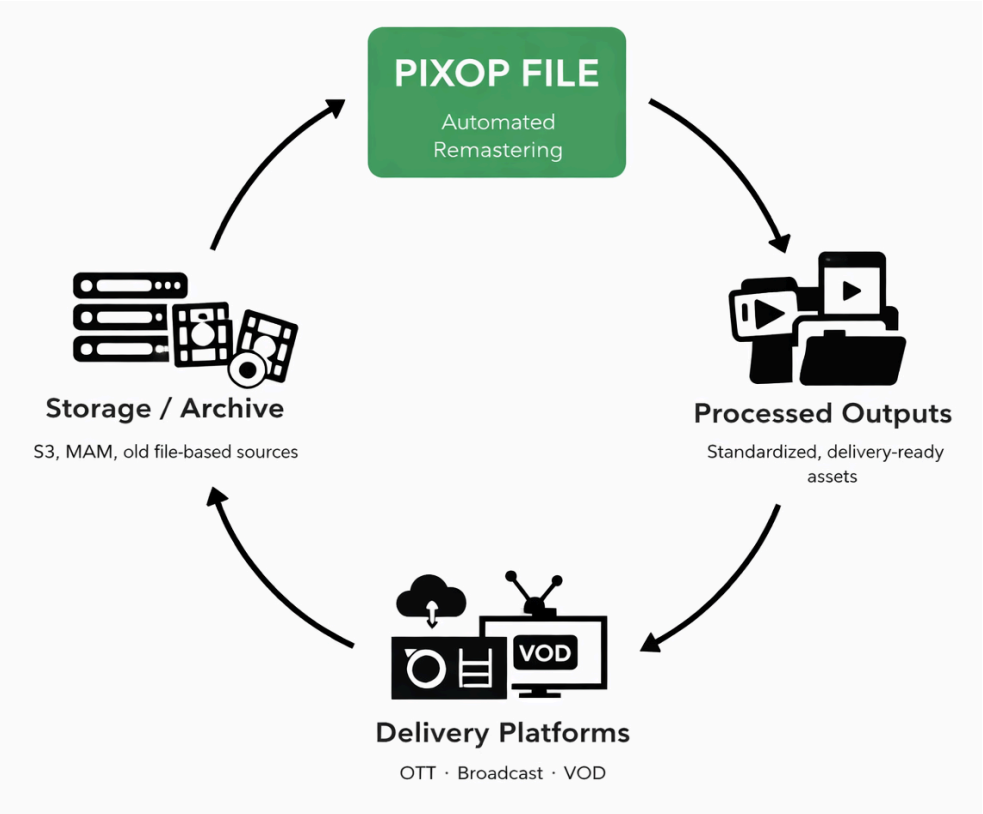


Table of contents

1. Overview 2
2. The Problem & Opportunity 3
3. Pixop FILE Overview 4
4. Core Capabilities 5
5. Architecture & Cloud Execution 7
6. Integration & Deployment 8
7. API & Control Model 11
8. Reliability & Operational Model 12
9. Use Cases 13
10. Technical Specifications 15
11. Conclusion & Next Steps 17

1. Overview

Pixop FILE is a cloud-based video processing platform for high-quality, large-scale transformation of video assets. It operates as a fully managed SaaS service within Pixop's AWS infrastructure, with no customer-managed compute, storage, or GPU resources.

The platform uses an asynchronous, resource-oriented execution model where assets, configurations, and jobs are managed as persistent entities through a REST API. Processing is executed as event-driven workflows, coordinating ingestion, transformation, and delivery across distributed cloud services.

Pixop FILE combines ML-based enhancement, format conversion, and signal processing into configurable pipelines operating on file-based inputs. These pipelines execute deterministically, ensuring consistent and repeatable results across large datasets.

The system is API-first, enabling integration into MAM, DAM, and custom workflows. Job lifecycle events are exposed via webhooks, allowing external systems to orchestrate processing in an event-driven manner.

Pixop FILE operates on externally managed storage, supporting ingestion from and delivery to systems such as Amazon S3 and HTTPS endpoints, avoiding the need for proprietary data migration.

A web-based application (app.pixop.com) provides interactive asset management, job configuration, and monitoring, with full parity to the underlying API.

The platform includes native integration with the Mimir MAM system, enabling metadata-driven workflows where processing is embedded directly into the asset lifecycle.

Pixop FILE is not a real-time system. It is optimized for quality, scalability, and automation rather than latency.

2. The Problem & Opportunity

2.1 Challenges in File-Based Video Workflows

Media organizations manage large volumes of video content across archives, production, and distribution. These assets vary significantly in quality, format, and metadata, including legacy and interlaced formats, compression artifacts, inconsistent resolutions, SDR content requiring HDR delivery, and incomplete metadata.

Improving and standardizing this content requires compute-intensive, multi-stage processing, often involving GPU acceleration.

Traditional file-based approaches introduce several limitations:

- Infrastructure complexity
  GPU environments must be provisioned, scaled, and maintained, leading to operational overhead and inefficient utilization.
- Fragmented orchestration
  Workflows are typically built from scripts and loosely coupled services, making them difficult to scale, monitor, and reproduce.
- Limited elasticity
  Handling large backlogs or burst workloads requires manual scaling or over-provisioning.
- Disconnection from asset systems
  Processing is often isolated from MAM/DAM systems, resulting in manual coordination and inconsistent metadata tracking.
- Non-deterministic outcomes
  Variability in tools and configurations leads to inconsistent results across large datasets.

2.2 The Opportunity: Event-Driven Processing as a Service Layer

These challenges create an opportunity to treat video processing as a cloud-native service integrated directly into asset workflows.

Pixop FILE addresses this by providing a managed, event-driven processing platform that:

- Abstracts infrastructure
  GPU compute, storage integration, and orchestration are fully managed.
- Enables elastic scaling
  Workloads scale automatically to support both continuous and batch processing.
- Standardizes execution
  Processing is defined as deterministic pipelines, ensuring consistent results.
- Integrates with asset lifecycles
  Jobs can be triggered by metadata and workflow events, including native Mimir integration.
- Supports event-driven orchestration
  Webhooks and API-based control enable automated coordination of downstream workflows.

This approach transforms video processing from a collection of tools into a unified, cloud-native system where ingestion, processing, and delivery are part of a single workflow.

3. Pixop FILE Overview

3.1 Product Overview

Pixop FILE is a GPU-accelerated, file-based video processing platform for batch and automated workflows. It operates as a fully managed service, enabling organizations to process video assets without managing infrastructure.

The system processes video as discrete assets using asynchronous execution, optimizing for throughput and scalability rather than latency. It supports a wide range of input formats and qualities, from legacy archives to modern mezzanine content, and produces standardized outputs aligned with delivery requirements.

Processing consolidates restoration, super-resolution, format conversion, and color handling into a unified pipeline, ensuring consistent and predictable results at scale.

Figure 1 illustrates the high-level flow of asset ingestion, processing, and remaster delivery within Pixop FILE.

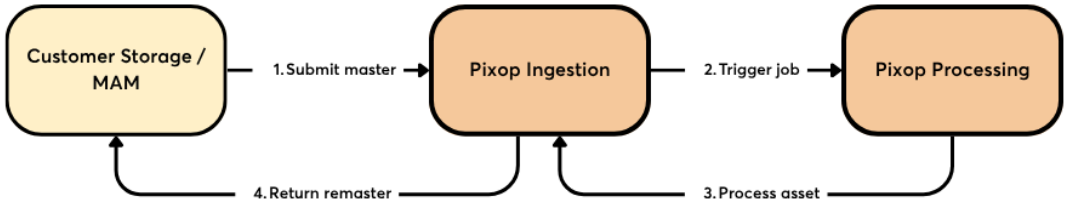


Figure 1 — High-level workflow for file-based ingestion, processing, and remaster delivery

3.2 Core Concept: Event-Driven Processing Pipelines

Pixop FILE is built around event-driven, deterministic processing pipelines.

Each job defines a structured pipeline applied to a video asset, with ordered stages such as decoding, enhancement, transformation, and encoding. These stages execute as a single process, ensuring each step operates on an optimized intermediate representation.

Execution is coordinated through asynchronous, event-driven state transitions across ingestion, processing, and delivery. This enables reliable scaling while maintaining clear job state and observability.

The system is deterministic by design. Identical inputs and configurations produce identical outputs, ensuring reproducibility across large-scale workflows.

3.3 Role in the Media Workflow

Pixop FILE operates as a processing layer within file-based workflows, positioned between storage and downstream distribution.

It is typically applied after ingestion into storage or MAM systems and before

delivery, packaging, or archive. Processing can be triggered manually, via API, or automatically based on metadata and lifecycle events.

By integrating directly into the asset lifecycle, Pixop FILE replaces standalone processing steps and establishes a consistent foundation for downstream systems.

3.4 Operational Modes (API, Web Application, Mimir)

Pixop FILE provides multiple interfaces to the same underlying platform:

- API-driven operation
  The primary interface is a REST API for managing assets, configurations, and jobs, enabling full automation and integration.
- Web-based operation
  The Pixop web application provides interactive access for asset management, job configuration, and monitoring, with full parity to the API.
- Mimir-integrated operation
  Native integration with the Mimir MAM system enables metadata-driven workflows, where processing is triggered and tracked within the asset lifecycle.

These modes are not separate systems, but different interfaces to the same execution model, ensuring consistent behavior across manual, automated, and integrated workflows.

4. Core Capabilities

4.1 Processing Pipelines and Filters

Pixop FILE processes video through configurable pipelines composed of discrete, composable filters.

Each filter performs a specific transformation, such as restoration, denoising, deinterlacing, super-resolution, or color conversion. Pipelines define how these are combined to produce the final output.

All filters execute within a unified pipeline, ensuring each stage operates on an optimized intermediate representation and avoiding inconsistencies from fragmented workflows.

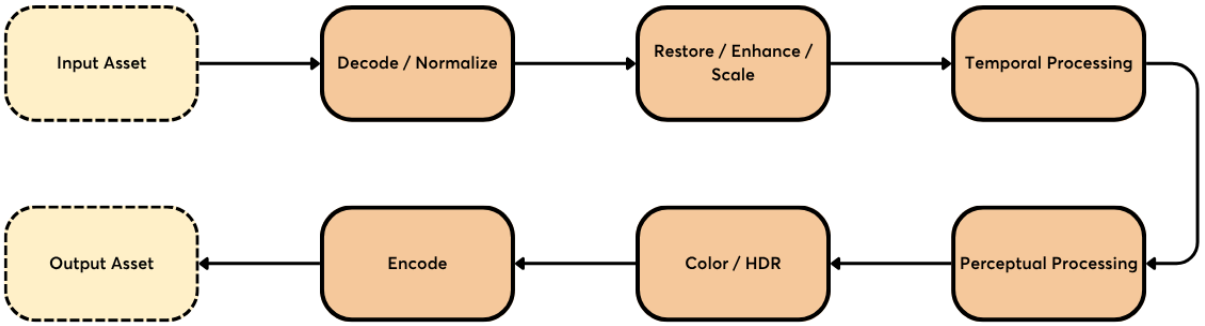


Figure 2 — Conceptual processing pipeline for restoration, enhancement, color transformation, and encoding.

4.2 Restoration and Enhancement Filters

Pixop FILE provides ML-based restoration filters to improve degraded or compressed video, addressing artifacts, noise, flicker, and instability.

The Pixop Deep Restoration 2 family includes multiple variants optimized for different content types. General-purpose models target broadcast and archive material, while specialized variants such as *selfie-style* are tuned for human-centric recordings.

This specialization ensures restoration is adapted to content rather than applied uniformly.

4.3 Super-Resolution and Spatial Processing

Pixop FILE supports ML-based super-resolution to increase spatial resolution and reconstruct detail from degraded inputs.

Super-resolution is applied after restoration, ensuring detail is generated from a stabilized signal. Additional spatial processing handles resolution alignment, aspect ratio normalization, and output formatting.

4.4 Temporal and Motion Processing

Pixop FILE includes filters for temporal consistency and motion handling, including:

- deinterlacing (interlaced to progressive)
- temporal stabilization and flicker reduction
- frame-rate conversion and motion interpolation

These operations are integrated into the pipeline to ensure stable behavior across frames.

4.5 Perceptual Processing

Pixop FILE includes perceptual processing filters applied in the final stages of the pipeline to improve visual quality and presentation.

These filters address artifacts introduced by compression and prior processing stages, including banding and loss of texture. Debanding reduces visible gradients and quantization artifacts, while film grain

synthesis restores natural texture and masks residual compression artifacts.

Perceptual processing is applied after restoration, enhancement, and color transformation to ensure that adjustments are based on the final signal. This positioning allows these filters to act as a finishing stage, improving subjective visual quality without altering structural content.

4.6 Color and Dynamic Range Processing

Pixop FILE performs color and dynamic range transformations as part of the pipeline, including:

- color space conversion (e.g., Rec. 709 to Rec. 2020)
- SDR to HDR conversion via tone mapping
- signal normalization to match target formats

Processing is applied in sequence to preserve visual intent while adapting content to modern delivery standards.

4.7 Pipeline Configuration and Reusability

Pipelines are defined as reusable configurations, enabling consistent processing across large datasets.

Configurations can be versioned and applied across workflows, ensuring controlled updates and reproducibility. Execution is deterministic by design, producing consistent outputs for identical inputs and configurations.

5. Architecture & Cloud Execution

5.1 System Overview (Ingestion → Processing → Delivery)

Pixop FILE is a cloud-native, event-driven system that orchestrates video processing across ingestion, transformation, and delivery.

Assets are ingested into cloud storage, processed through GPU-accelerated pipelines, and delivered back to storage or external systems. Execution is coordinated through asynchronous jobs and distributed services, enabling scalable and reliable processing.

Control, orchestration, and execution are decoupled, allowing each layer to scale independently.

5.2 Ingestion and Storage Model

Pixop FILE operates on externally managed storage, ingesting assets from sources such as S3 or HTTPS.

Ingestion is decoupled from processing. When an asset is registered, a job is created and queued, allowing ingestion to scale independently and supporting both continuous and batch workflows.

All assets remain in standard storage systems, avoiding proprietary storage and

enabling integration with existing workflows.

5.3 Event-Driven Orchestration

Processing is coordinated through an event-driven orchestration layer.

Jobs are stateful entities progressing through stages such as ingestion, queued, processing, and completed. Transitions are managed through asynchronous messaging and distributed control services.

Orchestration handles scheduling, lifecycle management, task triggering, and event propagation, enabling high concurrency and fault isolation without centralized bottlenecks.

5.4 Processing Execution Layer

Pixop FILE uses a hybrid CPU/GPU execution model.

GPU acceleration is applied to ML-based Pixop filters, while FFmpeg-based operations such as decoding, encoding, and format handling run on CPU.

Processing pipelines combine these stages within a single execution flow. Jobs are executed in isolation, with workers pulling input from storage, executing pipelines, and writing outputs back. The system scales horizontally by increasing worker capacity.

5.5 Data Flow and Asset Lifecycle

Each asset follows a defined lifecycle:

- ingestion → asset registered in storage
- job creation → configuration applied
- queueing → job scheduled
- processing → pipeline executed
- output generation → results written to storage
- completion → state updated and events emitted

This lifecycle is observable via API and webhooks, enabling integration into external systems.

5.6 Scalability and Throughput Characteristics

Pixop FILE is designed for elastic scalability.

Processing capacity scales horizontally with worker nodes, while ingestion, orchestration, and execution remain decoupled. This allows the system to handle steady workloads and bursts without manual intervention.

Throughput depends on GPU capacity, pipeline complexity, and input characteristics, while the architecture maintains high utilization across workloads.

6. Integration & Deployment

6.1 API-Based Integration

Pixop FILE integrates directly into media workflows through a resource-oriented REST API.

All functionality, including asset management, pipeline configuration, job submission, and monitoring, is exposed programmatically. Jobs execute asynchronously, with lifecycle events delivered via webhooks, enabling event-driven orchestration and downstream automation.

This model supports both high-volume batch processing and continuous ingestion workflows.

6.2 Mimir Integration (Metadata-Driven Workflows)

Pixop FILE includes native integration with the Mimir MAM platform, enabling processing as part of the asset lifecycle.

A Mimir tenant is mapped to a Pixop team via API credentials, allowing processing to be initiated and managed directly within the MAM interface. Workflows are exposed as custom actions linked to Pixop configurations and applied to assets.

Assets are transferred to Pixop for processing and returned to Mimir-managed storage, with results associated with the original asset. Job state and lifecycle updates are synchronized via webhooks, ensuring visibility within Mimir.

The integration supports:

- job tracking and visibility within Mimir
- usage controls (e.g., per-asset limits)
- automatic cleanup of intermediate data
- optional approval workflows

Processing can also be embedded into Mimir UI elements such as action menus and workflow triggers, enabling automation within editorial and archive workflows.

This eliminates the need for external orchestration, allowing Pixop FILE to operate as a native processing layer within metadata-driven workflows.

Figure 3 shows a screenshot of Pixop FILE integrated into Mimir.

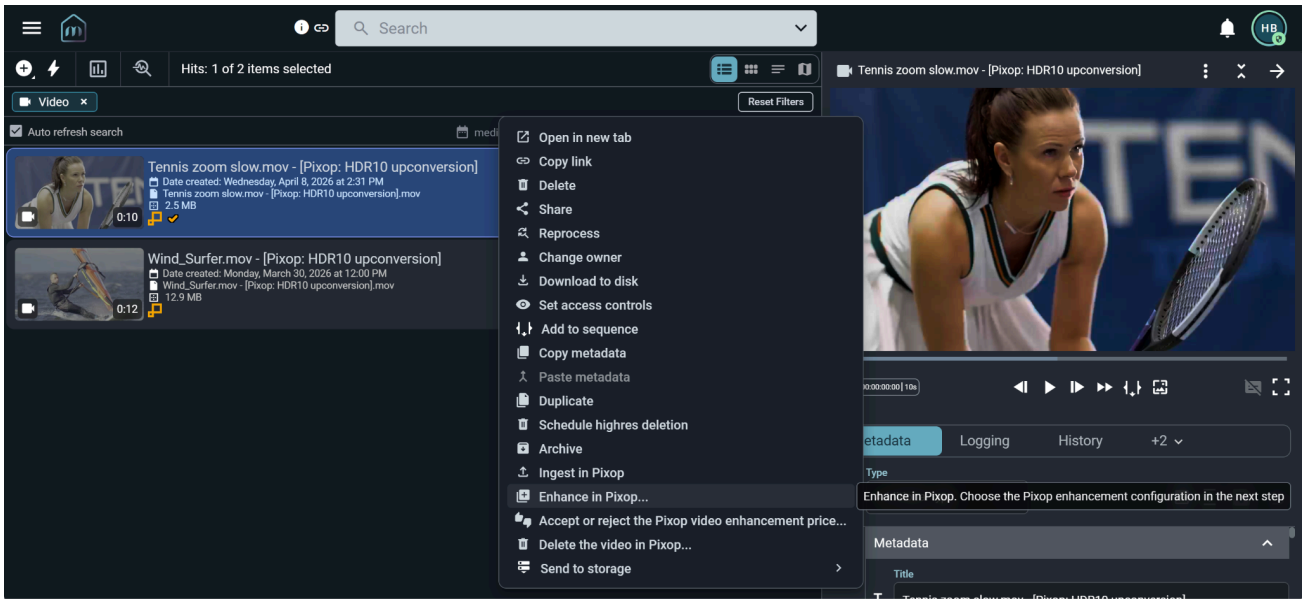


Figure 3 — Mimir user interface showing the context menu for initiating processing workflows.

6.3 Storage Integration

Pixop FILE operates on externally managed storage, ingesting from and delivering to endpoints such as S3 and HTTPS.

This allows organizations to retain control over media storage while using Pixop FILE as a processing layer, supporting cloud-native, hybrid, and archive environments without requiring data migration.

6.4 Workflow Integration

Pixop FILE integrates with MAM, DAM, and custom pipelines.

Processing can be triggered manually via the web application, programmatically via API, or automatically through metadata-driven workflows. This flexibility allows integration into existing systems without workflow redesign.

6.5 Deployment Model (Fully Managed SaaS)

Pixop FILE is delivered as a fully managed SaaS platform.

All infrastructure is operated within Pixop's AWS environment, eliminating the need to provision or manage compute, storage orchestration, or scaling. The platform is accessed via API and web interfaces (see figure 4), supporting both automated and interactive use.

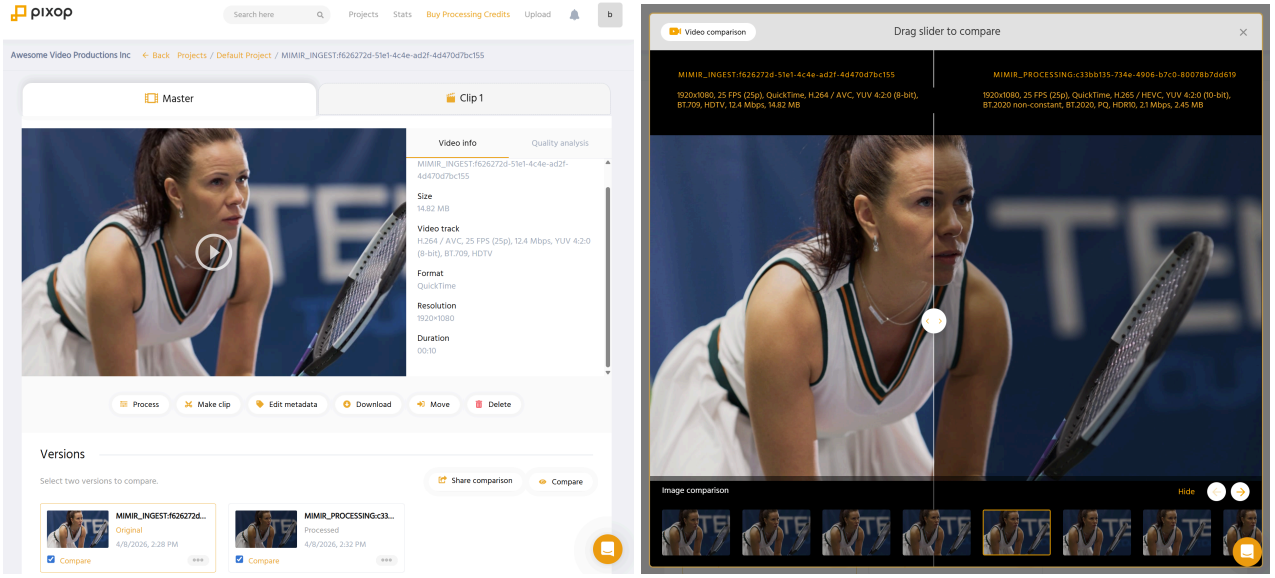


Figure 4 — Pixop web interface with asset view and visual comparison.

7. API & Control Model

7.1 Resource-Oriented API

Pixop FILE exposes a resource-oriented REST API where assets, configurations, and jobs are managed as persistent entities.

Each resource has a defined lifecycle and state, enabling consistent interaction across ingestion, processing, and delivery. This model allows Pixop FILE to operate as a stateful processing platform integrated into external workflows.

7.2 Job Lifecycle and Asynchronous Execution

Processing is fully asynchronous.

Jobs progress through a lifecycle (e.g., created → queued → processing → completed/failed) and execute independently without blocking client systems. This enables high-throughput batch processing and integration into distributed workflows.

Job state can be queried at any time for progress tracking and downstream coordination.

7.3 Webhooks and Event-Driven Orchestration

Pixop FILE provides webhook-based event delivery for job lifecycle updates.

External systems can subscribe to events such as completion or failure, enabling event-driven orchestration. Webhook delivery is asynchronous and may be out of order, requiring resilient handling by consuming systems.

This reduces the need for polling and supports loosely coupled workflow integration.

7.4 Authentication and Access Control

Access is secured via team-scoped API keys.

API credentials authenticate requests and isolate resources, such as assets, configurations, and jobs, between teams and environments.

7.5 API and Web Application Usage

Pixop FILE provides both programmatic and interactive interfaces to the same platform.

The REST API is the primary interface for automation, while the web application enables manual workflows such as asset management, job configuration, and monitoring. Both interfaces operate on the same system, ensuring consistent behavior across usage modes.

8. Reliability & Operational Model

8.1 Asynchronous Execution and Fault Isolation

Pixop FILE operates as an asynchronous, distributed system where jobs execute independently.

Each job is isolated, preventing failures from impacting others. Ingestion, orchestration, and processing are decoupled, enabling recovery from transient issues without cascading failures and ensuring stable operation across varying workloads.

8.2 Job Reliability and Retry Behavior

The system handles transient failures through automatic retry mechanisms.

Jobs are coordinated via managed orchestration and messaging services, allowing failed tasks to be rescheduled without manual intervention. Job state is persisted throughout execution, ensuring reliable progress tracking and deterministic recovery.

8.3 Observability and Monitoring

Pixop FILE provides observability through job state tracking, metrics, and logs.

Job lifecycle states are exposed for external monitoring, while internal metrics capture throughput, execution time, and system behavior. This supports both automated monitoring and troubleshooting at scale.

8.4 Distributed Orchestration and Managed Services

Pixop FILE is built on managed cloud services for orchestration, messaging, and state management.

These include stateless compute, distributed queues, notification systems, and managed databases (e.g., Lambda, Batch, SQS, SNS, DynamoDB). This architecture provides built-in scalability, fault tolerance, and high availability without custom infrastructure.

8.5 Data Integrity and Processing Guarantees

Input assets are treated as immutable, with outputs generated as new artifacts to ensure traceability.

Processing is deterministic, producing identical outputs for identical inputs and configurations. Job state is persisted and observable, enabling reliable verification and downstream integration.

8.6 Operational Control and Debugging

Job state can be queried via API and monitored through webhooks.

Logs and metrics provide visibility into execution behavior, enabling debugging and performance analysis. The system is designed to be operated through observable state and events without direct infrastructure access.

8.7 Security & Data Protection

Pixop FILE applies encryption at all stages of the data lifecycle, including in transit and at rest. Media assets are transferred directly to encrypted object storage, minimizing intermediary data handling and reducing the attack surface.

Sensitive values are protected using application-layer encryption, and all data is isolated at the team level to ensure strict access control. Asset access is governed through time-limited URLs, enabling secure sharing without persistent exposure.

The platform is built on AWS managed services, inheriting a security model that provides encryption, isolation, and operational resilience by design.

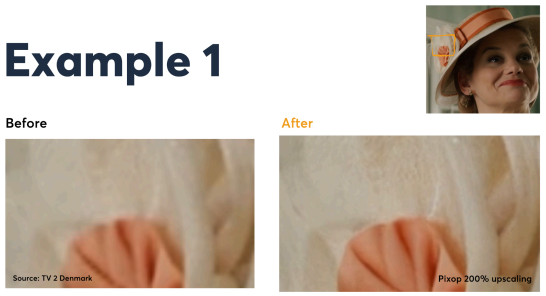
9. Use Cases

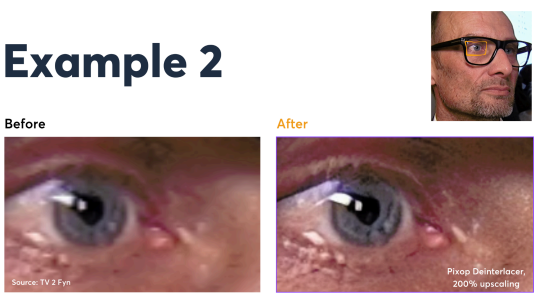
9.1 Archive Restoration and Remastering

Pixop FILE is used to restore and enhance legacy video archives at scale.

Archived content often contains compression artifacts, noise, interlacing, and inconsistent formats. Pixop FILE applies restoration, deinterlacing, and super-resolution within a unified pipeline to improve visual quality and standardize outputs (see examples 1 and 2).

Because processing is deterministic and repeatable, large archives can be processed consistently using predefined configurations. This enables organizations to upgrade legacy content libraries without manual intervention or per-asset tuning.





9.2 UHD and HDR Upconversion

Pixop FILE is used to convert existing HD and SDR content into UHD (4K) and HDR formats for modern distribution.

This includes pipelines that combine restoration, super-resolution, and dynamic range conversion to transform legacy or mezzanine content into formats aligned with current delivery standards.

Processing typically involves:

- restoration of compression artifacts and noise
- super-resolution to increase spatial resolution to 4K UHD
- SDR to HDR conversion (e.g., PQ or HLG)
- color space alignment to Rec. 2020

These transformations are applied within a unified pipeline, ensuring that each stage operates on an optimized signal. Restoration improves the input prior to scaling, and super-resolution provides a higher-quality foundation for tone mapping and color processing.

This approach produces more consistent results than applying these operations independently, where each stage may operate on suboptimal input.

UHD and HDR upconversion enables:

- reuse of existing HD content for UHD channels and platforms
- alignment with modern display capabilities
- improved visual quality in OTT and broadcast delivery

9.3 Large-Scale Batch Processing

Pixop FILE supports large-scale batch processing workflows where thousands or millions of assets must be processed reliably.

The event-driven architecture allows jobs to be queued and executed asynchronously, scaling processing capacity based on demand. Pipelines can be applied consistently across entire datasets, ensuring uniform output without manual coordination.

This is particularly relevant for backlog processing, content migration, and library-wide upgrades.

9.4 Automated MAM-Driven Workflows

Pixop FILE integrates directly into media asset management systems, enabling fully automated processing workflows.

Through Mimir integration, processing can be triggered by asset metadata, lifecycle events, or workflow rules. Assets can be automatically processed upon ingestion, update, or classification.

Processing results are returned to the MAM system and associated with the original asset, enabling end-to-end automation without external orchestration.

This allows video transformation to be embedded directly into content workflows rather than handled as a separate operational step.

9.5 Content Standardization and Format Alignment

Pixop FILE is used to standardize content across diverse sources and formats.

In workflows where content originates from multiple sources, such as archives, user-generated content, or distributed production, Pixop FILE can normalize resolution, frame rate, color space, and dynamic range into a consistent output format.

This ensures compatibility across downstream systems and improves consistency in presentation and delivery.

10. Technical Specifications

Summary of key technical capabilities for integration and evaluation. Supported configurations may vary depending on deployment and hardware.

10.1 Media Formats & Interfaces

	Input	Output
Interfaces	S3, HTTPS	S3, HTTPS
Containers	MP4, MOV, MXF, MPEG-TS, and other FFmpeg-supported formats	MP4, MOV, MXF, MPEG-TS
Video Codecs	H.264, HEVC, MPEG-2, ProRes, DNxHD, XDCAM, FFV1, and other FFmpeg-supported codecs	H.264, HEVC, MPEG-2, ProRes, DNxHD, XDCAM, FFV1
Audio	Supported (container/codec dependent)	Passthrough (container/codec dependent)
Data / subtitles	Preserved where supported	Passthrough

10.2 Video Format Support

	Supported
Input Resolutions	SD, HD, Full HD, UHD (4K), UHD (8K)
Output Resolutions	Up to 8K (7680×4320)
Frame Rates	Up to 240 fps
Standard frame rates	23.976, 24, 25, 29.97, 30, 50, 59.94, 60, 100, 119.88, 120
Scan Types	Interlaced, progressive
Bit Depth	8-bit, 10-bit
Chroma	4:4:4 / 4:2:2 / 4:2:0
Color Range	Limited, full
Rate Control	Constant Rate Factor (CRF), average video bitrate
Aspect ratio handling	Display, storage, pixel-aspect preserved

	Supported
Framing	Crop, pad, fixed-output alignment

10.3 HDR & Colorimetry

	Supported
Color Spaces	Rec. 601, Rec. 709, Rec. 2020
Dynamic range	Rec. 709, Rec. 2020-10, PQ (HDR), HLG (HDR)
HDR formats	HDR10 (PQ), HLG
SDR → HDR	Inverse tone mapping up to 1000 nits (ML-based and analytical)
HDR → SDR	Analytical tone mapping
Tone-mapping operators	Linear, Hable, Reinhard, Mobius, Clip
HDR metadata	Static HDR10 metadata (SMPTE ST.2086, MaxCLL, MaxFALL)
HDR control	Output luminance (nits), saturation control (model-dependent)
Metadata handling	Conversion or tagging-only modes

10.4 Processing Model

	Specification
Pipeline model	Configurable filter-based pipelines
Execution	Unified multi-pass pipeline
Processing stages	Decoding → restoration → enhancement → transformation → encoding
Determinism	Repeatable outputs for identical inputs
Processing types	Restoration, super-resolution, format conversion, color and HDR processing
Compute model	Hybrid CPU (FFmpeg) and GPU (Pixop ML filters)

10.5 Filter Categories

	Functions
Restoration	Artifact removal, denoising, dejitter, detail recovery
Deinterlacing	Interlaced to progressive conversion
Super-resolution	ML-based upscaling (up to 6×)
Frame-rate conversion	Interpolation and frame-rate normalization
Spatial processing	Scaling, cropping, aspect ratio handling
Color processing	Color space conversion and signal alignment
Dynamic range	SDR ↔ HDR conversion

	Functions
Perceptual processing	Film grain synthesis, face enhancement

10.6 Model Variants

	Description
Deep Restoration 2 (Generic)	General-purpose restoration and scaling
Deep Restoration 2 (Fine-Tuning)	Conservative artifact removal without aggressive denoising
Deep Restoration 2 (Selfie-Style)	Optimized for webcam and human-centric recordings

10.7 Automation and Control

	Supported
Control model	Resource-oriented REST API
Job execution	Asynchronous, split-and-merge
Job tracking	Lifecycle-based state model
Event handling	Webhook-based notifications
Integration model	Event-driven orchestration
Authentication	API key (team-scoped)

10.8 Performance Characteristics

	FHD	4K
Pixop Deep Restoration 2	~48 FPS	~13 FPS
Pixop Dejitterer	~72 FPS	~14 FPS
Pixop Deinterlacer	~47 FPS	~11 FPS
Pixop Denoiser	~11.2 FPS	~2.4 FPS

11. Conclusion & Next Steps

Pixop FILE provides a structured approach to file-based video processing by combining restoration, enhancement, and format conversion into unified, event-driven pipelines.

By consolidating processing into a single, deterministic system, it ensures consistent results across large-scale workloads while eliminating the need to manage GPU infrastructure or fragmented toolchains. The platform integrates directly into media workflows through its API and native Mimir support, enabling automation and metadata-driven processing.

Pixop FILE is designed for production environments where scalability, reproducibility, and operational simplicity are required. Its cloud-native architecture enables efficient processing of large content volumes while maintaining full control over workflows and integration.

The platform can be evaluated using existing media assets and integrated into current workflows without infrastructure changes. Typical adoption paths include:

- batch processing of archive content
- mezzanine preparation for distribution
- automated MAM-driven workflows
- format standardization across content libraries

Pixop FILE is available as a fully managed service and can be accessed through the web application or REST API. Organizations can begin by testing representative workflows and scaling as needed.