

Dock as First Responder

Enterprise Deployment & Operations Guide

Contents

- Preface** 002
 - About This Document 002
 - Who Should Read This 002
 - What This Document Will Help You Do 003
 - Chapter Overview 003
 - Recommended Reading Paths 004
 - Core Terminology 004
- Chapter 1: Industry Context and DFR Value Analysis** 006
 - 1.0 DFR Vision: Four Layers of Aerial Response Infrastructure 006
 - 1.1 Why DFR: A Faster and Lower-Cost Response Resource 007
 - 1.2 From Patrol-Based Deployment to Dock as First Responder 008
 - 1.3 Why Dock DFR Is Ready for Wider Deployment 011
 - 1.4 Common Use Cases Across Public Safety and Private Security 013
 - 1.5 DFR Project Application Stages 015
 - 1.6 Preconditions and Risk Controls: Gantt-Style Stage Matrix 016
- Chapter 2: DFR System Architecture** 018
 - 2.0 Chapter Overview 018
 - 2.1 FlightHub 2 System Overview 019
 - 2.1.1 Product Introduction 019
 - 2.2 FlightHub 2 Logical Architecture 024
 - 2.2.1 Architecture Breakdown 024
 - 2.2.2 From Architecture to Integration 027

2.3 FlightHub 2 Open Interfaces	029
2.3.1 FlightHub Sync	029
2.3.2 Open API	034
2.3.3 Frontend Independent Components	038
2.4 Integration Interface Selection Guide	040
2.4.1 Live Streaming	040
2.4.2 Telemetry	051
2.4.3 Media	060
2.4.4 Event API	074
2.5 Deployment Architecture Options	080
2.5.1 Public Cloud	080
2.5.2 All-in-One (AIO)	081
2.5.3 On-Premises (Private Deployment)	083
2.5.4 Deployment Form Selection Guide	085
2.6 Product Reference Links	086
Chapter 3: Application Scenarios & Workflow Analysis	088
3.0 Chapter Overview	088
3.1 Workflow Analysis	089
3.1.1 Traditional Workflow	089
3.1.2 DFR-Enhanced Workflow	091
3.1.3 Role Breakdown & System Interaction	099
3.1.4 System Integration - From Requirements to Architecture	104

Chapter 4: Data Security and Compliance	107
4.0 Chapter Overview	107
4.1 Data Types and Sensitivity Analysis	108
4.2 DFR Data Flow	110
4.3 Deployment Types and Data Control	112
4.4 Security Control Framework	113
4.4.1 Link and Transmission Security	113
4.4.2 Identity and Access Control	114
4.4.3 Storage and Data Protection	115
4.4.4 Network and Infrastructure Protection	116
4.4.5 Secure Development, Vulnerabilities, and Incident Response	116
4.5 What ISO 27001 Represents	117
4.6 Audit Logs and Accountability	118
4.7 Third-Party Integration Security	118
4.7.1 CAD / Alarm / PSIM: Alarm Trigger and Auto Dispatch	118
4.7.2 VMS / Command Center: Livestream Forwarding	119
4.7.3 MQTT / GIS / PSIM: Telemetry Synchronization	120
4.7.4 Evidence / S3 / Archive: File Sync and Archiving	121
4.7.5 EventAPI: Business Event Push	121
4.7.6 OpenAPI and Frontend Components: Deep Integration	122
4.8 Public Trust and Privacy Governance	122
4.9 IT Review Delivery Package	123
4.10 Shared Responsibility Model	124
4.11 Conclusion	124

Appendix A: Internal Data Flow Reference	126
Data Flow 1: Device Registration & Command Delivery	127
Data Flow 2: From Lens to Screen	132
Data Flow 3: From Plan to Propeller	137
Data Flow 4: From Flight to Archive	141
Appendix B: Network Port Configuration	145

Preface

About This Document

This whitepaper is issued by DJI Enterprise, drawing on Dock as First Responder (DFR) deployment experience across public safety and private security programs in multiple regions. It is not a product manual. It is a practitioner's reference — structured around the decisions, trade-offs, and sequencing that determine whether a DFR integration succeeds in a real operational environment.

The content reflects engagements that have moved beyond proof-of-concept into enterprise IT review, production deployment, and operational service delivery.

Who Should Read This

System integrators and solution architects — the integration interface guides, architecture decision frameworks, and phased deployment models in Chapters 2 and 3 are written for teams designing and delivering DFR systems against real customer requirements.

Enterprise IT and security teams — the security control framework, shared responsibility model, and network port reference in Chapter 4 and Appendix B are structured to support formal IT review processes.

Operations leads and program managers — the five-stage project evolution model, role breakdown, and integration sequencing plan in Chapters 1 and 3 provide the structure to scope delivery commitments and build a repeatable program methodology.

This document assumes technical competence. It does not assume prior DJI product knowledge.

What This Document Will Help You Do

- Establish the business case for DFR with a client or internal stakeholder — grounded in operational cost comparisons and documented market cases
- Select the right integration interfaces for each data domain — live streaming, telemetry, media, and event notification — with protocol-level detail and decision logic
- Design a phased integration architecture that validates at each stage before the next layer of complexity is added
- Prepare the documentation package required to pass enterprise IT security review
- Staff and operate a production DFR program with a defined team structure and escalation paths

Chapter Overview

	Chapter / Appendix	Core Question	Reader Value
Ch.1	Industry Context & DFR Value Analysis	Why is DFR deployable at scale now? How should a project evolve?	Commercial framing, five-stage project roadmap, readiness matrix
Ch.2	DFR System Architecture	What interfaces does FH2 expose? Which fits each integration scenario?	Interface selection guide, protocol comparisons, architecture patterns, deployment sizing
Ch.3	Application Scenarios & Workflow	What does a live DFR operation look like? Who does what, and in what sequence?	End-to-end workflow, six-role staffing model, three-phase integration sequencing plan
Ch.4	Data Security & Compliance	Where does data go? Who can access it? How do you pass IT review?	Security control framework, ISO 27001 evidence, IT review delivery package
App. A	Internal Data Flow Reference	How does FH2 process data internally? Where do failures originate?	Component-level flow diagrams for production troubleshooting and support escalation
App.B	Network Port Configuration	Which ports and subnets does On-Premises require?	Actionable port list and subnet reference for firewall and network configuration

Chapters 2 and 3 are designed to be read together. Chapter 2 defines what is technically possible; Chapter 3 shows what that looks like in operation. Reading only one gives an incomplete picture.

Recommended Reading Paths

Reader Priority	Recommended Path
Evaluating DFR feasibility	Ch.1 → Ch.3 §3.1.1–3.1.2
Designing integration architecture	Ch.2 §2.3–2.4 → App.A
Selecting deployment form and sizing infrastructure	Ch.2 §2.5 → App.B
Preparing for IT security review	Ch.4 → App.B
Staffing the operations team	Ch.3 §3.1.3–3.1.4
First full read	Ch.1 → Ch.2 → Ch.3 → Ch.4 → Appendices

For system integrators moving from reference reading to solution design, this white paper is accompanied by the [Awesome DFR Integration Skill](#), an AI-ready integration resource that organizes DFR architecture, workflow, and FlightHub 2 interface guidance into structured context for middleware planning and implementation support.

Core Terminology

Term	Definition
DFR — Drone as First Responder	A response model in which an autonomous drone launches from a docked station ahead of ground units to provide aerial scene confirmation
FH2 — FlightHub 2	DJI’s enterprise drone fleet management platform; the central coordination layer in a DFR deployment
Dock / DJI Dock 3	DJI’s autonomous docking station; enables automated launch, landing, and recharging without on-site personnel

Term	Definition
Auto Dispatch Workflow	FH2's workflow engine connecting external alarm triggers to drone response via a workflow POST endpoint or inbound trigger endpoint
CAD	Computer-Aided Dispatch — the system used by public safety agencies to log, classify, and assign incident responses
VMS	Video Management System — centralized management and display of video feeds
GIS	Geographic Information System — geospatial data visualization, used in DFR for drone position display and map annotation
PSIM	Physical Security Information Management — integrates cameras, access control, and alarms into a unified operator interface
FlightHub Sync	FH2's configuration-based data bridge for livestream forwarding, file synchronization, and telemetry forwarding without custom development
On-Premises	A deployment model in which FH2 runs on customer-managed infrastructure within the organization's own network boundary

01 | Industry Context and DFR Value Analysis

1.0 DFR Vision: Four Layers of Aerial Response Infrastructure

Public safety agencies and private security operators already have dispatchers, patrol officers, vehicles, fixed cameras, VMS platforms, radios, and evidence systems. The gap sits earlier in the response chain: verifying what is happening before committing high-cost ground resources.

Dock as First Responder (DFR) fills that gap by turning the drone into part of the response infrastructure. Aerial confirmation becomes a routinely available response resource, similar to patrol cars, cameras, and radio systems.

From an industry perspective, the DFR value chain has four layers:

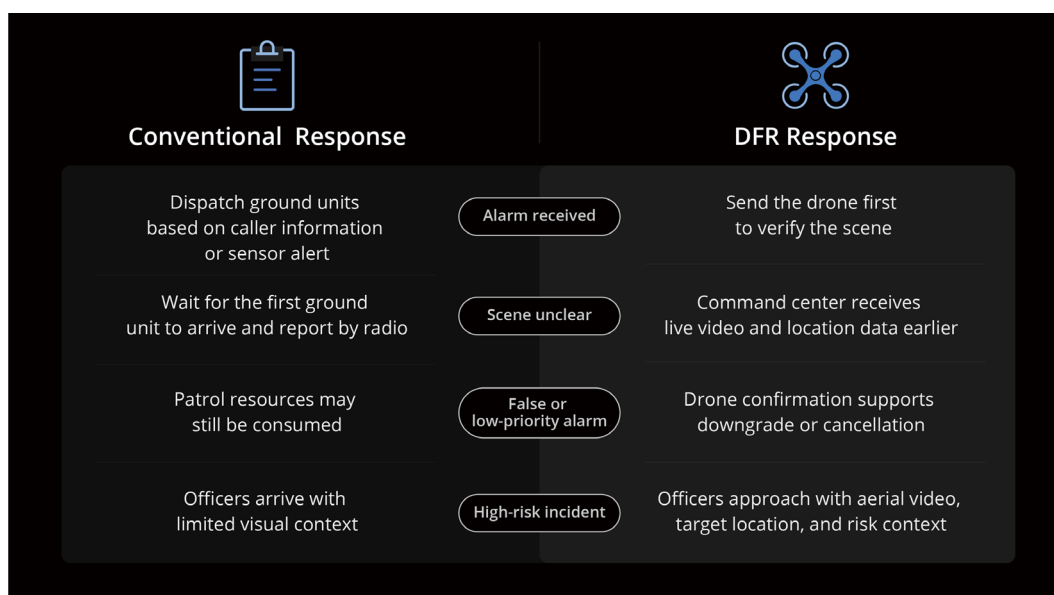
Layer	Role	Opportunity	Challenge
Top layer	Data users:public safety agencies and private security operators	Use aerial data to support response decisions and reduce unnecessary dispatch	Connect drone data into existing command workflows and establish response SOPs
Third layer	Dock as a Service (DaaS) operators	Provide drone response as a subscription service and move from project margin to service margin	Build a remote operations center (ROC) and SLA assurance capability
Second layer	System integrators	Design and implement DFR systems and turn delivery into a repeatable methodology	Handle multiple interfaces, customer-specific environments, and delivery tooling
Base layer	Solution providers/ technology platforms	Provide docks, aircraft, flight management platforms, and open interfaces	Ensure reliability, multi-network support, and long-term availability across markets

This whitepaper is issued from the DJI perspective and focuses on the system integrator layer. Its purpose is to provide a practical best-practice reference for selecting, deploying, integrating, and operating DFR systems in a repeatable, auditable, and compliance-aware way.

1.1 Why DFR: A Faster and Lower-Cost Response Resource

Ground response resources are expensive and limited. A patrol vehicle carries trained personnel, equipment, communications systems, fuel, insurance cost, and opportunity cost. Sending that resource to every alarm creates visible waste when the event turns out to be false, low priority, or already resolved.

DFR changes the sequence. The drone goes first. Ground units follow after the aerial view confirms that a ground response is needed.



The operational value is direct:

Value	Meaning
Lower-cost confirmation	A drone response usually costs far less than a full patrol vehicle and officer dispatch
Faster scene awareness	A docked drone can launch quickly and stream live video before ground arrival
Better resource allocation	Dispatchers can reserve officers, patrol cars, medical teams, and security teams for confirmed incidents

DFR should be planned as a decision-support layer. It helps the dispatch center qualify an incident before committing ground response resources.

1.2 From Patrol-Based Deployment to Dock as First Responder

Early DFR programs used patrol-based deployment. A trained pilot received a call, carried or prepared a drone, launched it near the operating area, flew to the scene, and streamed the view back to the command team. This model proved the value of aerial response, yet it scales poorly. Every new coverage area creates more demand for pilot scheduling, standby coverage, equipment readiness, and launch preparation.



Patrol-based deployment still has operational value for SWAT overwatch, indoor operations, dynamic scenes, or cases where a trained pilot is already on site. Dock DFR adds a scalable response infrastructure layer for routine, repeatable, and time-critical coverage.



Dock as First Responder changes the cost structure of the operating model. The table below compares the same response scenario under a patrol-based deployment model and a docked automation model.

Cost Dimension	Patrol-Based Deployment	Dock Automation	Quantified Impact
Single-dispatch travel cost	A pilot drives to the scene with the aircraft, typically adding 10–30 minutes	A nearby dock launches and arrives in about 3–5 minutes	Response travel time reduced by 70–80%
Pilot labor	Each response requires one pilot to remain engaged throughout the mission	One pilot can supervise up to four active dock operations in mature procedures	Pilot productivity can increase by up to 4x
Field safety exposure	The pilot enters an uncertain operating environment	The pilot supervises remotely and does not need to enter the scene	Worker exposure and third-party liability risk are reduced
Multi-shift night coverage	24/7 coverage requires multiple pilot shifts	Automated tasks plus one remote duty pilot can cover more sites	Night standby labor can be reduced by 60–70%
Equipment loss and crash exposure	Manual stick input creates more pilot-error exposure	Standard routes and boundary rules shift routine execution to the system	Internal data indicates Dock DFR crash rates are significantly lower than manual flight
Training and qualification	Every pilot must maintain active control proficiency	Training shifts toward remote supervision and exception handling	Training cost decreases and pilot value increases

Beyond cost, Dock as First Responder changes the capability boundary of a DFR project in four ways:

Upgrade Area	Patrol-Based Deployment	Dock as First Responder	Capability Change
Launch model	Pilot prepares and launches the drone	Dock stays ready and launches when triggered	Shorter launch time
Staffing model	Often one pilot per drone	Mature operations can support up to a 1:4 pilot-to-active-dock oversight ratio	Scaling becomes less dependent on pilot headcount
Safety model	Manual stick input creates human-error exposure	Standard missions run through automation, task boundaries, and remote oversight	Manual-control risk decreases
Pilot role	Low-level flight control	Mission supervision, risk judgment, and exception handling	Trained pilots perform higher-value work

Safety is a key part of this upgrade. Internal crash reviews show that more than 90% of drone crash incidents are linked to pilot error. Dock DFR reduces that exposure through standardized mission execution: planned routes, task boundaries, geofencing, return-to-home logic, and remote supervision.

The pilot role changes as well. The industry is moving from stick-level flight control toward supervision of automated aviation tasks. A mature DFR team should use pilots for judgment, mission supervision, airspace awareness, exception handling, and operational accountability.

1.3 Why Dock DFR Is Ready for Wider Deployment

DFR deployment conditions are maturing at the same time.

Condition	What Has Changed	Why It Matters for Integrators
UAS safety management	More markets now have procedures for flight approval, operating areas, pilot qualification, risk assessment, and emergency response	DFR projects can be delivered through more repeatable compliance paths
Hardware and software compliance capability	Docks, aircraft, geofencing, mission logs, remote oversight, lost-link handling, and return procedures are more mature	Integrators can document safety controls in customer and regulator materials
System integration capability	FlightHub 2 can provide live video, telemetry, media, events, and mission data	DFR can connect to CAD, VMS, PSIM, GIS, and evidence systems
Proven projects	Public safety, city management, private security, and critical infrastructure projects have moved beyond demos	Customer education is easier and procurement risk is lower

Approval structures differ by country and customer type. Private security projects usually need to work with civil aviation authorities and local airspace rules. Public safety projects may involve police aviation authorities, civil aviation regulators, public-aircraft frameworks, or multi-agency review.

The practical work is similar across markets: define the flight area, write the operating procedure, assign responsible personnel, prepare emergency actions, explain data processing, and prove that the system can operate safely in the customer environment.

Formal cases now exist in several regions, each representing a different market entry point:

- **Fidelity Security Group, South Africa:** one of South Africa's largest private security companies, with operations across mining areas, industrial parks, and commercial

zones. By deploying DJI Dock 3, Fidelity can move from a patrol-based deployment model toward 24/7 automated perimeter response. Once docks are pre-positioned at high-risk sites, the dock responds first and the pilot verifies remotely, shortening response time and allowing pilots to supervise more locations.



Figure 1.3-A DFR Case Study — Fidelity Security Group (South Africa)

- **Mondiapol, Italy:** an Italian security group serving residential, commercial, and critical infrastructure customers. Mondiapol integrates DJI Dock 3 and FlightHub 2 into its existing security service workflow. OpenAPI and Auto Dispatch Workflow triggers can push drone data into CAD and incident management processes, making DFR part of a repeatable service package rather than a standalone equipment delivery.

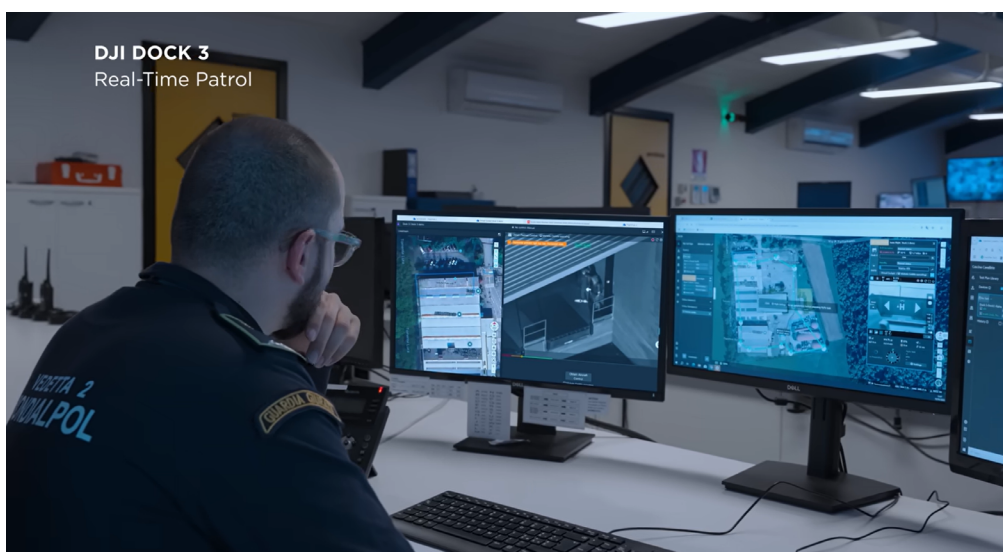


Figure 1.3-B DFR Case Study — Mondiapol (Italy)

- **El Paso, United States:** a municipal public safety project emphasizing compliance, auditability, and data sovereignty. DJI Dock 3 with FlightHub 2 On-Premises supports local data control and law-enforcement audit requirements, enabling the command center to dispatch drones directly and shift from waiting for a patrol vehicle to obtaining aerial confirmation within minutes.



Figure 1.3-C DFR Case Study — El Paso Police Department (United States)

The shared signal is clear: Dock DFR has moved beyond demonstration. For integrators, the core work is to extract repeatable methodology and tooling instead of starting each project from zero.

1.4 Common Use Cases Across Public Safety and Private Security

Public safety and private security operate under different mandates. The first operational question often matches: does this alarm deserve more ground resources?

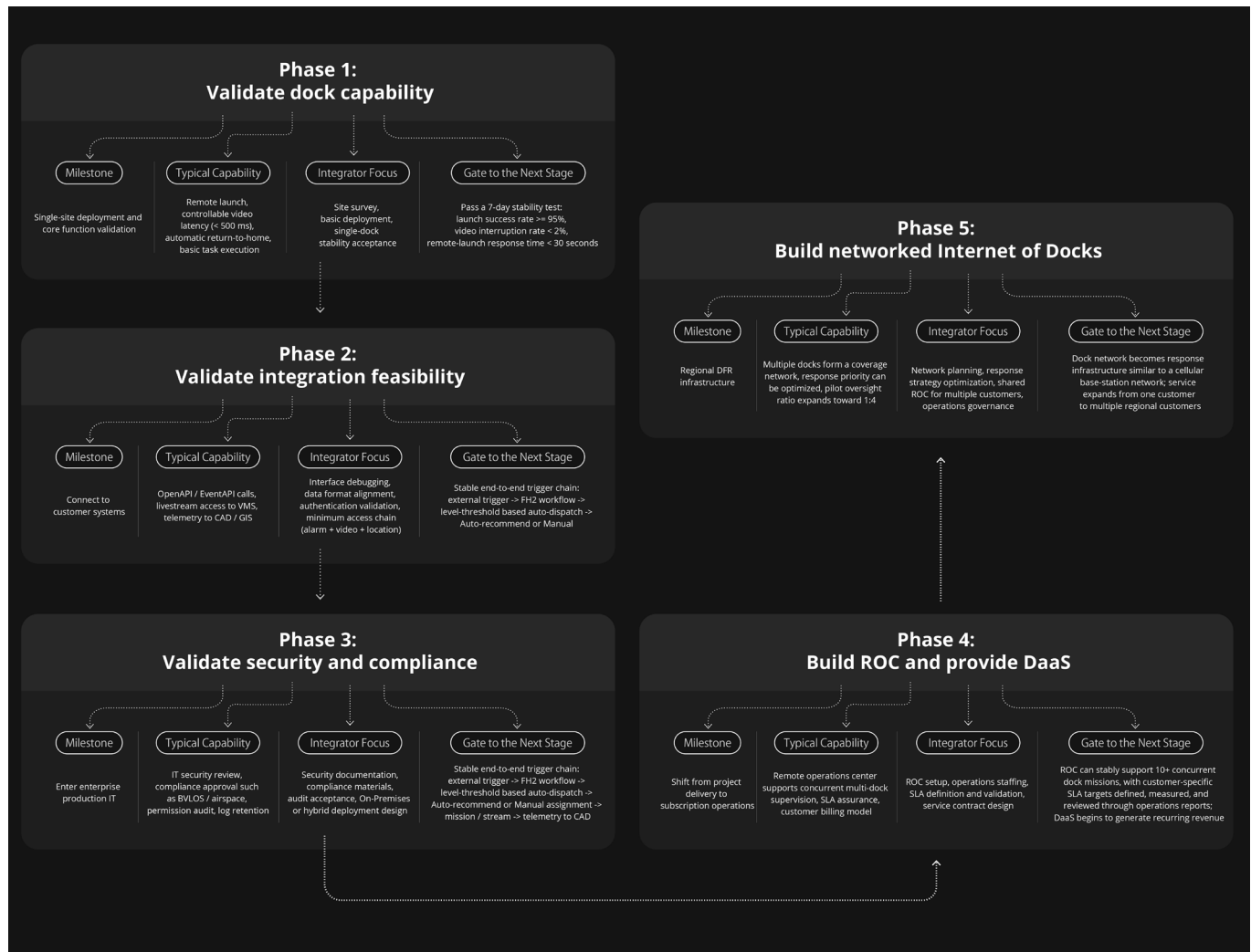
User Type	Scenario	Drone First Action	Follow-On Decision
Public safety	Traffic accident	Check vehicle position, congestion, injuries, and secondary risk	Decide whether to escalate police, fire, or medical response
Public safety	Intrusion or theft	Confirm whether the suspect remains on site	Route patrol units or adjust containment points
Public safety	Violent or high-risk call	Identify people, vehicles, approach routes, and danger zones	Reduce blind entry risk for officers
Public safety	Search and rescue	Expand the search area quickly from above	Guide ground teams to the right zone
Private security	Industrial perimeter alarm	Fly to the trigger point and confirm people, vehicles, or intrusion status	Dispatch guards, call police, or downgrade the alarm
Private security	Logistics park or port	Cover large areas at night	Reduce unnecessary patrol movement
Private security	Energy or oil and gas site	Inspect high-risk areas before sending staff	Decide whether human entry is required
Private security	Major event or exhibition	Check crowd movement and perimeter conditions	Reallocate security staff in real time

These scenarios share three capabilities:

Capability	Operational Meaning
Fast verification	The drone fills the information gap left by unclear alarms
Live situational awareness	Video, location, mission status, and alarm context enter the command workflow
Resource qualification	Ground response starts with better information and clearer priority

1.5 DFR Project Application Stages

DFR should not be designed as a one-time delivery. It is an evolution path from pilot validation to response infrastructure. Each stage has acceptance criteria and an upgrade gate. For integrators, this becomes a roadmap for project margin, customer commitment boundaries, and internal capability building.



The transition between stages is a change in operating object and business model:

- **Phase 1 to Phase 2:** from standalone dock operation to the customer's system ecosystem.
- **Phase 2 to Phase 3:** from interface connectivity to enterprise-grade security and compliance acceptance.
- **Phase 3 to Phase 4:** from project delivery to service operations.
- **Phase 4 to Phase 5:** from serving one customer to becoming regional infrastructure.

The ROI of a DFR operator rises step by step as the service model matures:

- **Security services (Phase 1–3):** lower input, lower output. The entry point is response efficiency, and the value is reducing unnecessary dispatch.
- **Inspection services (Phase 3–4):** input expands. Scheduled automated missions plus AI detection create high-frequency, standardized inspection data.
- **Mapping and asset digitization (Phase 4–5):** input expands into modeling, AI training, and data operations. Output becomes a long-term data asset with higher reuse value.

The DaaS operator begins to move toward platform profitability.

1.6 Preconditions and Risk Controls: Gantt-Style Stage Matrix

DFR engineering readiness does not happen once. Each domain deepens as the project moves from validation to production operation. The chart below shows readiness investment by domain across five project phases — darker shading indicates heavier investment.

Readiness Domain	Phase 1 Validate Capability	Phase 2 Validate Integration	Phase 3 Validate Compliance
Site Deployment	Light site survey	Formal installation	Multi-position fixed installation
Power / Network	Temporary power + 4G	Dedicated power + network	UPS + private line + redundancy
Flight Approval	VLOS or pre-approval	Remote operation approval	Formal BVLOS / airspace approval
Airspace Safety	Basic airspace data	Geofence and airspace planning	Complete airspace management
System Integration	Not connected	Minimum stack: alarm + video + location	CAD + VMS + evidence
Data Security	Public Cloud baseline	Basic permissions	Enterprise IT review
Roles and SOP	Simple role split	Interface and operations roles	Complete SOP and qualifications
Maintenance and SLA	Manual checks	Regular checks	Formal maintenance plan

Readiness Domain	Phase 4 DaaS Operations	Phase 5 Internet of Docks
Site Deployment	Multi-customer sites	Regional network
Power / Network	Production monitoring	High-availability wide-area network
Flight Approval	Cross-customer approval reuse	Routine approval framework
Airspace Safety	Cross-area coordination	Dynamic airspace management
System Integration	Multi-customer interface standards	Platformized access
Data Security	Multi-tenant permission and audit model	Governance and retention system
Roles and SOP	ROC + SLA team	Governance and training system
Maintenance and SLA	SLA-driven maintenance	Predictive maintenance

The matrix sends three practical signals:

- **Phase 1** has the lowest input. The purpose is fast validation of dock value.
- **Phase 3** usually carries the highest project friction because enterprise security and regulatory acceptance must be completed.
- **Phase 4 and Phase 5** convert investment into output because ROC and networked operations allow integrators to earn operational service revenue instead of one-time engineering revenue.

High-quality DFR projects validate four things early: aerial confirmation reduces unnecessary ground response, dispatch decisions become more accurate, dock coverage can scale, and the installation can operate reliably over time.

02 | DFR System Architecture — A Technical Deep Dive

2.0 Chapter Overview

This chapter is the technical foundation for any FlightHub 2 integration project. It is written for system integrators and enterprise technical teams who need to understand not just what FH2 can do, but how it is structured, what interfaces it exposes, and how to make informed decisions at every stage of an integration engagement.

By the end of this chapter, you will be able to:

- **Understand what FlightHub 2 is and how it is organized** — its core functional modules, logical service architecture, and the role it plays in a DFR deployment
- **Select the right integration interface for your scenario** — FlightHub Sync, Open API, Frontend Components, and Event API each serve a different integration need; this chapter maps each one to the scenarios where it applies
- **Make informed technical decisions across four integration domains** — live streaming, telemetry, media, and event notification; each domain includes interface selection guidance, focused scenarios, protocol primers, and integration best practices
- **Choose the right deployment form for your project phase** — Public Cloud for validation, FlightHub 2 All-in-One (AIO) for pre-production confirmation, and On-Premises for scaled production delivery
- **Access the full reference set** — documentation links, hardware sizing, port configuration, and deployment manuals consolidated in one place

This chapter does not assume prior DJI product knowledge. It does assume technical competence — the content is written for engineers and architects who will be making integration decisions.

2.1 FlightHub 2 System Overview

FlightHub 2 is the operational backbone of a DFR deployment — managing devices, missions, data, and third-party integration from a single platform. Understanding what FH2 does and how it is structured gives you the foundation to make informed decisions throughout the integration process.

This section introduces FH2's core functional modules, its role in a DFR architecture, and the deployment forms available. By the end, you will have a clear picture of what FH2 brings to your solution and where it sits relative to the systems you are integrating with.

2.1.1 Product Introduction

FlightHub 2 Brief Introduction

DJI FlightHub 2 (FH2) is an enterprise-grade drone flight management platform for unmanned systems operations and integration. As the central coordination layer in a DFR deployment, FH2 manages mission scheduling, device lifecycle, real-time situational awareness, data aggregation, and third-party system integration — across cloud-hosted and on-premises environments.

Three deployment forms are available, each targeting different infrastructure and compliance requirements:

Public Cloud — hosted service, minimal setup overhead

On-Premises — private server deployment, full data sovereignty

AIO — pre-configured hardware unit, optimized for rapid deployment

Integration interfaces remain consistent across all three forms.

Core Functional Modules

FH2's capabilities are organized into four integration-relevant groups:

Operations & Control

Remote management of aircraft, docks, and missions through a unified interface.

Command and control via Virtual Cockpit — browser-based situational display and flight management interface

Multi-type mission planning and route scheduling

Automated dock operations — DJI Dock

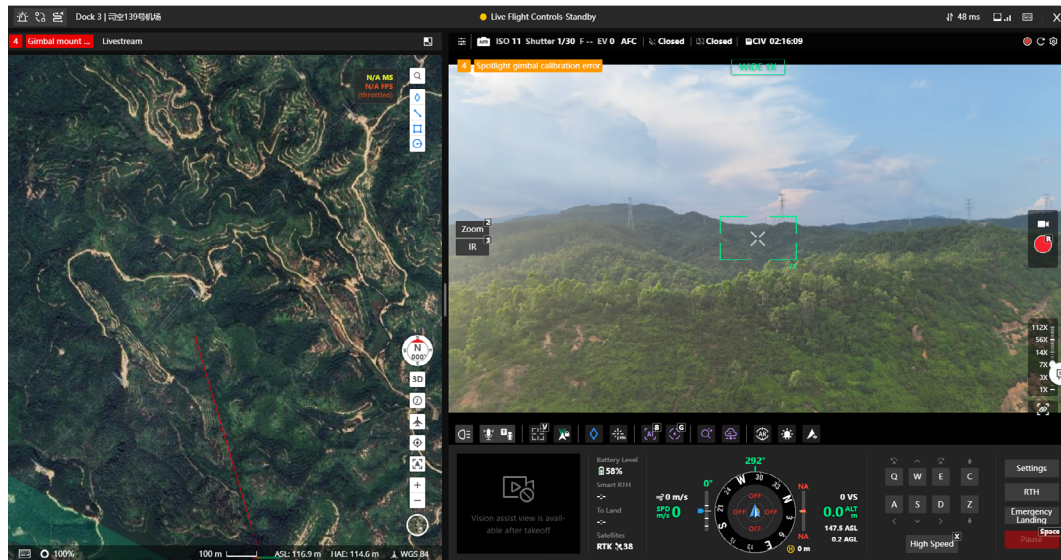


Figure 2.1.1-A Operations & Control — Virtual Cockpit & Flight Planning Interface

Automated Response

Event-driven workflow trigger with assisted or configured automatic dispatch.

Auto Dispatch Workflow — external systems call the FH2 workflow POST endpoint or inbound trigger endpoint. The request uses X-User-Token and lowercase x-project-uuid; this lowercase project header differs from the PascalCase X-Project-Uuid convention used in OpenAPI v2.0 references. Dispatch is level-threshold based: event levels are 1–5, the default threshold is 3, and auto-dispatch fires only when event level $\geq$ threshold. Two modes are available: Auto-recommend (FH2 selects the most suitable idle device; the recommended-device list is used as fallback) and Manual assignment (fixed device or per-trigger manual pick).

Pre-configured on-scene action sequences (e.g., AI detection, loudspeaker, spotlight)

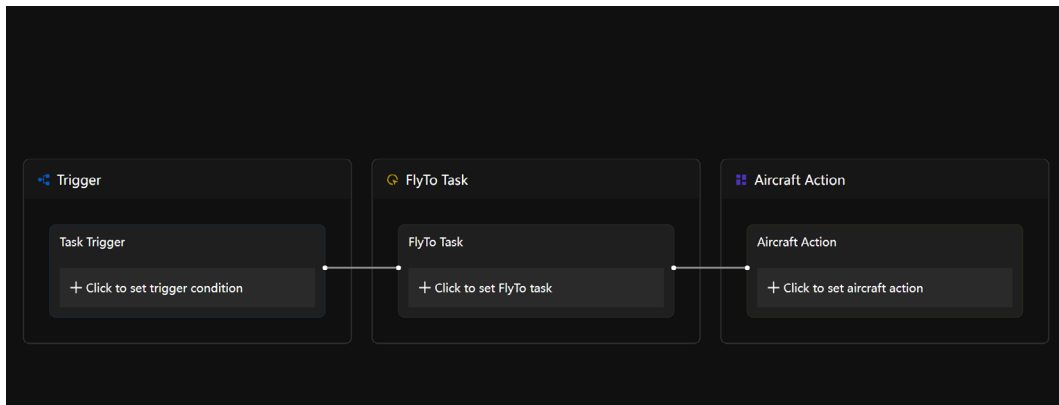


Figure 2.1.1-B Automated Response — Auto Dispatch Workflow & Remote Inspection

Data & Media

Centralized collection, processing, and archiving of mission output.

Mission-linked media management and file archiving

Cloud Modeling — automated 3D reconstruction from mission imagery; requires GPU

End-to-end mission traceability and task log access



Figure 2.1.1-C Data & Media — Media Library & Cloud Modeling

Open Integration

Standardized interfaces for connecting FH2 to existing enterprise systems.

Key external integration interfaces include FlightHub Sync, Public Cloud Telemetry Data

Forwarding, Open API, Frontend Independent Components, On-Premises MQTT Bridge, and Event API, with availability and configuration paths depending on deployment form and integration scenario.

On-premises deployment with configurable network isolation and access control

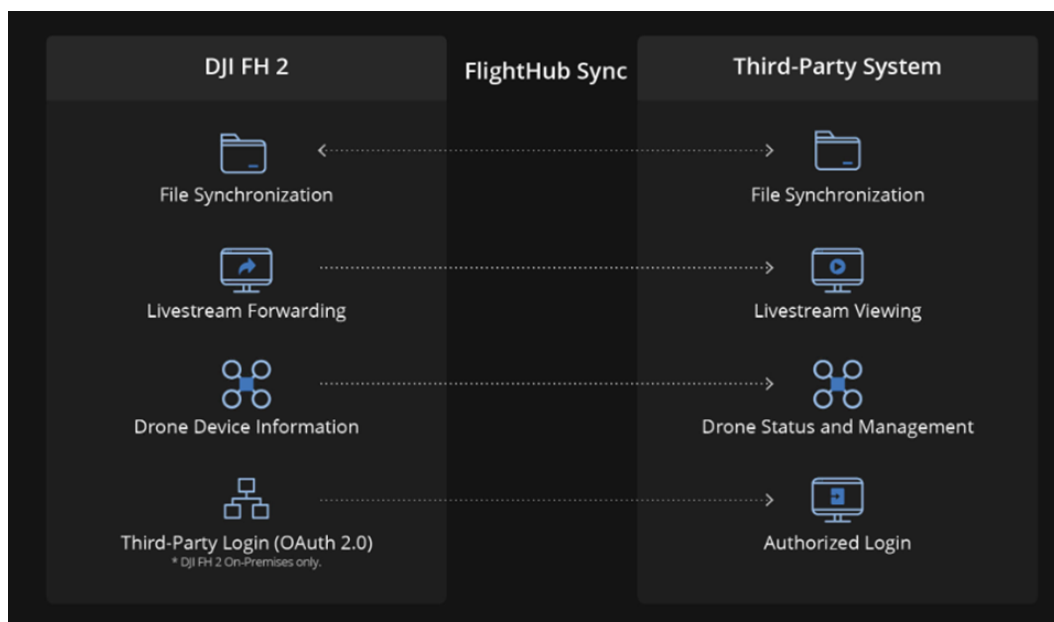


Figure 2.1.1-D Open Integration — FlightHub Sync & Open API Interface

DFR Integration Role

Auto Dispatch Workflow connects external alarm sources to drone response workflows through the workflow POST endpoint or inbound trigger endpoint — no custom hardware integration required.

When triggered, FH2 applies level-threshold based auto-dispatch (event levels 1–5, default threshold 3; auto-dispatch fires when event level $\geq$ threshold). In Auto-recommend mode, FH2 selects the most suitable idle device, using the recommended-device list as fallback. In Manual assignment mode, a fixed device or per-trigger manual pick is used. The platform then executes the configured mission sequence, and streams live video, telemetry, and mission results to third-party platforms via FlightHub 2's open integration interfaces.

For system integrators, the implication is direct: drone response capability can be embedded into an existing command platform or PSIM through a single workflow POST endpoint endpoint, with no dependency on DJI-specific SDKs.

Workflow configuration and prerequisites are covered in Chapter 3.

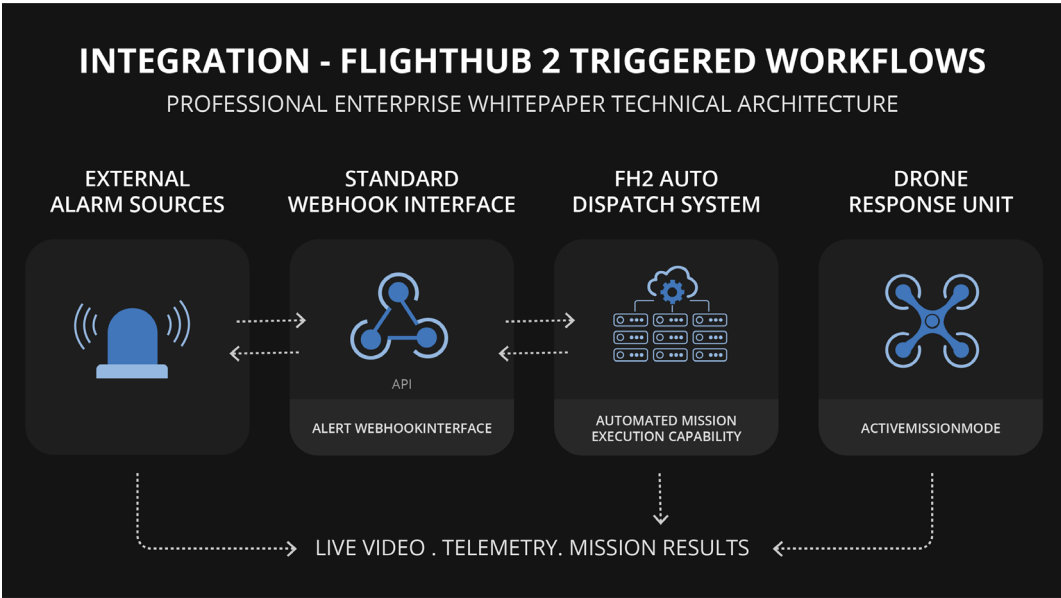


Figure 2.1.1-E Auto Dispatch Workflow — DFR Alert-to-Dispatch Sequence

FlightHub 2 as PaaS: Plug Drone Intelligence into Your Platform

FH2 exposes drone operations as a structured data service — four standardized interfaces connect directly to existing security platforms.

Interface	Data	Protocol	Integration Point
Live Video	Real-time aerial video	RTSP / RTMP	Standard RTSP/RTMP endpoint consumable by VMS, NVR, or any RTSP-capable display client
Telemetry	Drone position & device status	MQTT	MQTT topic stream subscribable by GIS, dashboard, or event-processing systems
Mission Media & AI Results	Footage + AI detection outputs	S3 / File Sync	S3-compatible object storage; files addressable via standard S3 API or pre-signed URL
Event Notifications	Mission lifecycle events	Webhook / EventAPI	Structured JSON callbacks via HTTP Webhook; EventAPI supports configurable event rule filters

Integrated platforms gain access to aerial intelligence through standardized interfaces — without changes to the host system architecture.

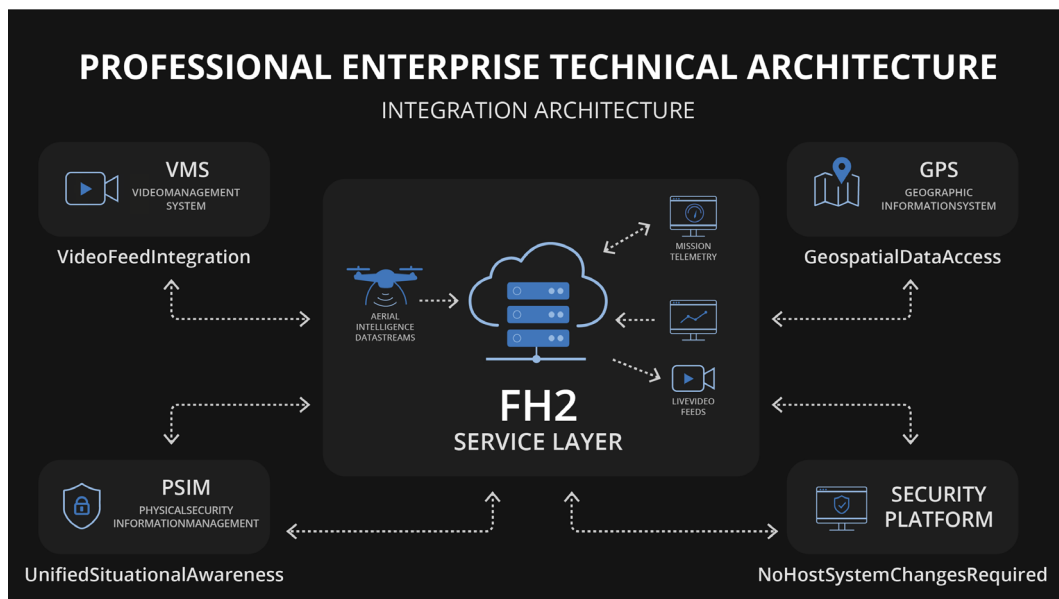


Figure 2.1.1-F FlightHub 2 PaaS Integration Topology

Terminology Reference

Terminology Reference: VMS (Video Management System), GIS (Geographic Information System), and PSIM (Physical Security Information Management).

2.2 FlightHub 2 Logical Architecture

Knowing what FH2 can do is not enough — knowing how it is organized internally tells you where integration points sit, what infrastructure decisions each domain drives, and where to look when something goes wrong.

This section breaks down FH2's five service domains and explains how third-party systems connect to them. By the end, you will be able to map your integration requirements to the right layer of the platform — and scope your project with confidence.

2.2.1 FlightHub 2 Architecture Breakdown

Understanding FH2's service architecture helps solution teams scope integrations accurately and identify the right touchpoints early.

Service Stack Overview

FH2 architecture is organized across five logical service domains, each with a defined scope and distinct role in the platform operation. For solution teams, understanding these domains clarifies where integration points sit, what deployment decisions each domain drives, and how to isolate issues when they arise -- much like understanding the floor plan of a building before deciding where to run the cables.

1. Authorization Management

Manages platform licensing, user identity, and access control across all system actors -- covering console users, connected devices, and third-party API clients alike. This domain operates as a prerequisite layer: authenticated access, device onboarding, and API authorization all depend on it being correctly configured and reachable. Think of it as the access control system at the building entrance -- no actor enters the platform without passing through it first.

Common scenarios:

- User login to the FH2 console
- Device registration and cloud onboarding (Dock / RC)
- Third-party platform access via Open API

2. Business Services

The operational core of FH2 -- where platform actions are initiated, coordinated, and tracked. This domain covers:

- Device management and fleet status monitoring
- Mission planning, scheduling, and dispatch
- Live streaming session control
- Media management and file archiving

For integrators, this is the primary point of interaction with FH2 operational logic -- via Open API, workflow POST / inbound trigger calls, or FlightHub Sync. Think of it as the operations room: every instruction issued by an operator or an external system passes through here before reaching the field.

3. Business Middleware

The internal communication layer between FH2 service domains. Responsible for real-time device messaging, event routing, and service coordination -- ensuring that state changes in one domain are reliably propagated to others.

Think of it as the switching network behind the scenes: messages go in, get routed to the right destination, and the rest of the platform stays in sync.

4. Post-processing Scheduler

Handles post-mission tasks as a separate asynchronous queue, fully decoupled from the flight control chain. Delays or failures here -- such as a slow media transfer or a modeling backlog -- do not affect active flight operations.

Common scenarios:

- Media file transfer from device storage to FH2
- Triggering Cloud Modeling once imagery is received
- Forwarding mission completion events to integrated third-party systems

Think of it as a task queue that picks up where the flight left off -- running independently in the background.

5. Modeling Services & Algorithms

Provides 3D reconstruction and AI algorithm processing for mission imagery -- the domain that turns raw aerial data into actionable outputs such as Cloud Modeling results and AI detection records.

Deployment considerations for on-premises installations:

- Cloud Modeling and compute-intensive algorithm tasks require a GPU-capable host. Where the primary server lacks GPU resources, this domain can be deployed on a dedicated secondary server.
- In a split-node configuration, reliable network connectivity between the two servers is a necessary condition -- connectivity gaps will affect processing throughput and job completion stability.

Think of it as a processing lab that can run alongside the main server or on its own machine -- as long as the two can communicate consistently.

What this means for integration planning

- **Interface exposure** -- which domain handles the capability being integrated (e.g., mission dispatch via Business Services, media access via Post-processing Scheduler)
- **Deployment dependencies** -- which domains require specific infrastructure (e.g., GPU for Modeling Services in on-premises deployments)
- **Fault boundary** -- where to look when something goes wrong (e.g., a failed media upload traces to Post-processing Scheduler, not Business Services)

2.2.2 From Architecture to Integration

For readers interested in FH2 internal data flows -- device registration, command routing, and media upload sequences -- these are covered in Appendix A. This section focuses on a more practical question: how do FH2 service domains connect to third-party systems, and through what interfaces?

Where FH2 services sit in an integrated architecture

The five service domains described in 2.2.1 all run within FH2 platform layer. External systems interact through two outward-facing interfaces:

- **Open API** -- the primary integration surface, backed by Business Services. Covers device management, mission dispatch, live streaming, flight route management, and media access.
- **Telemetry Data Forwarding / MQTT Bridge** — forwards real-time telemetry and device events. On Public Cloud, this is implemented as Telemetry Data Forwarding: FH2 acts as an MQTT client and pushes data to a customer-managed broker. On On-Premises deployments, this is implemented as MQTT Bridge: a broker-to-broker connection between the FH2 internal EMQX and the customer broker, configured via REST API POST /bridges/v1/resources. For systems that need live device state without polling.

Outputs from the Post-processing Scheduler and Modeling Services domains -- completed media files and Cloud Modeling results -- are accessible through the Open API surface once processing is complete.

How third-party systems connect

Two patterns cover most integration scenarios:

- **Request-response pattern** — your platform calls FH2 via Open API to query state, dispatch missions, or retrieve media on demand
- **Event-driven pattern** — FH2 pushes data to your platform via Public Cloud Telemetry Data Forwarding or On-Premises MQTT Bridge for telemetry, and via Event API for notifications as events occur

Pattern	Interface	Typical use
Request / response	Open API	Mission control, data retrieval
Continuous push	MQTT Bridge	Live telemetry, event monitoring

Third-party platforms can also embed FH2 operational UI directly -- Flight Route Editor, Virtual Cockpit, and Map are available as Frontend Components that integrate into a host platform without rebuilding those capabilities from scratch.

Behind these integration patterns sit four dedicated interfaces -- each designed for a different integration scenario. Section 2.3 walks through how each one works and when to use it.

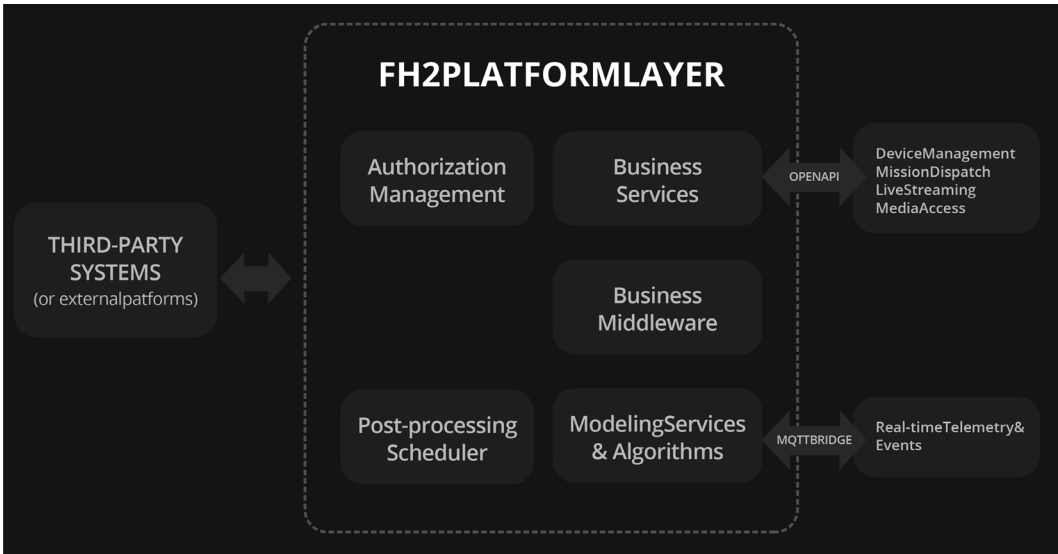


Figure 2.2.2-A FlightHub 2 System Integration Architecture

2.3 FlightHub 2 Open Interfaces

FlightHub 2 exposes three primary integration interfaces — FlightHub Sync, Open API, and Frontend Independent Components — giving your platform direct access to drone operational capability at the level that fits your project. A fourth integration capability, Event API, is part of the FlightHub Sync layer and is covered in depth in Section 2.4.4.

This section maps out what each interface connects to, what it unlocks, and when to use it. By the end, you will know exactly which interface fits your requirements and what to prepare before integration begins.

2.3.1 FlightHub Sync

FlightHub Sync is FlightHub 2's configuration-based data bridge -- connecting FH2's live streams, device telemetry, and mission media to third-party systems without custom development. Where Open API requires active development effort, FlightHub Sync is designed for integrators who need to move data between FH2 and existing platforms through GUI-based configuration alone.

Core Capabilities

Livestream Forwarding

Forwards real-time drone video from FH2 to external platforms via RTMP or RTSP -- no media server development required on the FH2 side. If your VMS or CAD system accepts an inbound stream address, this is the fastest path to getting aerial video into your existing display. Here is how to set it up.

- Configurable per device and per camera source
- One-click start / stop from the FH2 dashboard
- Typical targets: VMS platforms, CAD systems, and command center displays that accept inbound RTMP or RTSP streams

New Forwarding Channel

Forwarding Channel Name

Live-Streaming Test

Forwarding Channel Type

☒ RTMP
☐ RTSP

Server Address ⓘ

rtmp://a.rtmp.stream.com/live2/StreamKey

Device Name

Stream Source ⓘ

Video Quality

Auto

☐ Create Next

Cancel

Confirm

Figure 2.3.1-A FlightHub Sync — Livestream Forwarding Configuration

File Synchronization

Automatically syncs mission output from FH2 to an S3-compatible storage endpoint -- covering media files, flight routes, 3D models, flight logs, and map annotations. Sync direction varies by data type, and each serves a different operational purpose. The table below breaks it down.

Data Type	Sync Direction	Notes
Media Files (Photos / Videos)	FH2 → Third-party storage	One-way
Flight Routes	FH2 ↔ Third-party storage	Two-way
3D Models	FH2 → Third-party storage	One-way
Flight Logs	FH2 → Third-party storage	One-way
Custom Zones & Annotations	FH2 ↔ Third-party storage	Two-way

Data Type	Direction	Integration Scenario
Media files (photos / videos)	FH2 to Third-party	Post-mission evidence archiving; feeding imagery into existing operational databases
Flight routes / 3D models	Bidirectional	Sharing pre-planned routes across platforms; syncing post-incident Cloud Modeling outputs -- such as 3D scene reconstructions captured during DFR overflights -- to GIS or documentation systems
Flight logs	FH2 to Third-party	Compliance archiving; operational audit trails
Custom zones / annotations	Bidirectional	Keeping geofence definitions and map markups in sync between FH2 and command platforms

Add Storage Configuration

Bucket Type

AWS S3

Bucket Name

eg: ***-name

Used to identify this bucket in rule configuration.

Bucket AK

eg: AKIAQCU5PAC4***

Bucket SK

eg: ad7ojxmbn***

Region

us-east-1

Preset Path

Optional Configuration

[Permission Configuration Code Sample](#)

Connectivity Check

Cancel

Save Configuration

Figure 2.3.1-B FlightHub Sync — File Synchronization Flow

Configure Sync Rule

Rule Name

Destination Storage Configuration

▼

+ Create Storage

Select Source Project

Q

☐ Sefateng

☐ Limberg Mining Company

☐ Rose Road

☐ Langebaan

☐ KEW Action Group

☐ Stellenbosch

☐ Two Rivers Platinum Mine

Sync Content and Direction Control

Configure transfer direction for each data type

Media Files

FlightHub → Third-Party

Third-Party → FlightHub

Syncing third-party media files to FlightHub not supported

Flight Route File

FlightHub → Third-Party

Third-Party → FlightHub

Model Data

FlightHub → Third-Party

Third-Party → FlightHub

Flight Logs

FlightHub → Third-Party

Third-Party → FlightHub

Log import not supported

Cancel

Save Rule

Compatible storage targets are configured as third-party S3-compatible storage buckets directly in the FH2 web interface, including managed object-storage services and self-hosted S3-compatible endpoints.

Up to 20 storage configurations per organization can be active simultaneously.

Telemetry Data Forwarding / MQTT Bridge

Enables third-party systems to subscribe to live device telemetry -- position, flight status, battery, and mission events -- directly from FH2 via a standard MQTT broker. Rather than polling for updates, your system receives a continuous push stream as soon as the bridge is active. The three steps below walk through the setup.

1. Broker connection -- Input the third-party broker address, port, and authentication credentials in FH2 settings. FH2 connects as an MQTT client.

Configure MQTT Server

Transfer Protocol

TCP

Host & Port

Enter host

1883

Enter host

Compliance Info

Enter username

Enter password
👁

Data Forwarding Description >>

Test Connection

Cancel

Save and Connect

Figure 2.3.1-C MQTT Bridge — Topic Subscription Configuration

2. Topic mapping -- Define which FH2 internal topics map to which topics on the external broker, using wildcard patterns (+/#).

Representative topics available for subscription:

Target Data	Topic
Aircraft real-time position & attitude (OSD, 0.5 Hz)	thing/product/{sn}/osd
Device property state changes (flight mode, storage status)	thing/product/{sn}/state
Mission execution events & progress updates	thing/product/{sn}/events
Service command acknowledgment (downlink reply)	thing/product/{sn}/services_reply

Figure 2.3.1-D MQTT Bridge — Payload Structure

3. Enable and verify -- Activate the bridge. Subscribed systems on the third-party broker begin receiving telemetry. Connection status is visible in the FH2 dashboard.

```
thing/product/8UUXN3D00A02MC/osd 2026/01/21 22:40:11
{ "OrganizationUUID": "34b8f9da-44ae-417c-80a4-1cdeaf70f09f", "biz_code": "device_osd", "version": "1.0", "data": { "host": { "acc_time": 3012858, "acdc_power_input": 463.3333435058594, "activation_time": 1764074727, "backup_battery": { "switch": 1, "temperature": 33.9, "voltage": 13224 }, "deployment_mode": 1, "drone_battery_maintenance_info": { "batteries": [ { "capacity_percent": 94, "index": 0, "temperature": 24.6, "voltage": 25327 } ], "heat_state": 0, "maintenance_state": 0, "maintenance_time_left": 0 }, "electric_supply_voltage": 220, "gimbal_holder_state": 0, "job_number": 101, "maintain_status": { "maintain_status_array": [ { "last_maintain_time": 0, "last_maintain_type": 17, "last_maintain_work_sorties": 0, "state": 0 }, { "last_maintain_time": 0, "last_maintain_type": 18, "last_maintain_work_sorties": 0, "state": 0 } ] }, "poe_link_status": 0, "poe_power_output": 0, "relative_alternate_land_point": { "latitude": 0, "longitude": 0, "safe_land_height": 70, "status": 1 }, "self_converge_coordinate": { "height": 115.85935549926758, "latitude": -8.89513493729708, "longitude": 13.435068850789996 }, "temp_mode_state": false, "working_current": 8870, "working_voltage": 47103 }, "sn": "8UUXN3D00A02MC" }, "timestamp": 1769006407158 }
```

Figure 2.3.2-A Open API Integration Model

Deployment

FlightHub Sync is available across both public cloud and on-premises deployments. OAuth 2.0-based third-party login -- allowing users to authenticate into FH2 using an existing enterprise identity provider -- is supported on on-premises deployments only.

2.3.2 Open API

For deployment authentication, Public Cloud uses organization token headers such as X-User-Token, while On-Premises uses OAuth 2.0.

Open API is FlightHub 2's primary programmatic interface for enterprise system integration — a RESTful web API that exposes FH2 operational resources to third-party platforms through standard HTTP requests.

Where FlightHub Sync handles data bridging through GUI configuration, Open API enables active, bidirectional integration: external platforms can query FH2 state, dispatch missions,

manage resources, and retrieve outputs as part of their own application logic.

Functional Coverage

Open API 2.0 exposes eight capability domains via standard HTTPS methods. All endpoints share the base path `/openapi/v2.0/`. Authentication requires X-User-Token in the request header (Public Cloud) or OAuth 2.0 (On-Premises), plus x-project-uuid for all project-scoped operations.

Domain	Methods	Scope	Key Endpoints	Reference
Organization & Project	GET / POST	Project list, member roles, organization structure	GET <code>/openapi/v2.0/project</code> POST <code>/openapi/v2.0/project/{project_uuid}/member</code>	Tutorial
Device Management	GET / POST	Device list (org & project scope), online status, camera info, HMS health data, thing model	GET <code>/openapi/v2.0/device</code> GET <code>/openapi/v2.0/project/device</code> GET <code>/openapi/v2.0/device/hms</code>	Tutorial
Real-time Control	GET / POST	Control permission query; camera lens switch; in-flight commands (RTH, pause, resume); TCA status	POST <code>/openapi/v2.0/devices/{sn}/cameras/{camera_index}/lens</code> POST <code>/openapi/v2.0/devices/commands</code> GET <code>/openapi/v2.0/devices/commands</code>	Tutorial
Livestream	GET / POST / DELETE	Initiate livestream sessions; retrieve pull address (url + url_type); stream converter management; recording task query	POST <code>/openapi/v2.0/live-stream/start</code> POST <code>/openapi/v2.0/live-stream/converter</code> GET <code>/openapi/v2.0/live-stream/converter</code>	Tutorial
Flight Task	GET / POST / PATCH	Wayline task creation, dispatch, status query, track retrieval, media access (image / video / ppk); flight records	POST <code>/openapi/v2.0/flight-task</code> GET <code>/openapi/v2.0/flight-task</code> GET <code>/openapi/v2.0/flight-task/{task_uuid}/media</code>	Tutorial

Domain	Methods	Scope	Key Endpoints	Reference
Wayline	GET / POST	Wayline list query, details, download URL; upload completion notification	GET /openapi/v2.0/wayline GET /openapi/v2.0/wayline/ {wayline_id} POST /openapi/v2.0/wayline/ upload-callback	Tutorial
Map	GET / POST / DELETE	Map annotation CRUD (point / line / polygon); custom flight area (project & workspace scope); airspace warning	POST /openapi/v2.0/map/ element GET /openapi/v2.0/map/ element DELETE /openapi/v2.0/map/ element/{element_id}	Tutorial
Model Reconstruction	GET / POST	Cloud 3D/2D model list, reconstruction, status, download URL; open modeling upload & task management	POST /openapi/v2.0/map/ element GET /openapi/v2.0/map/ element DELETE /openapi/v2.0/map/ element/{element_id}	Tutorial

Note: Organization creation, project creation, and device binding are not currently supported via Open API. These operations must be completed through the FH2 web console before API-based integration begins. For full API reference, see the Open API Documentation.

For the complete endpoint reference, request/response schemas, and authentication flow, see the FlightHub 2 Open API v2.0 Reference and the Open API User Manual.

/live-stream/start returns a livestream pull address. When the response contains url_type = srs, the pull URL is a WebRTC-WHEP endpoint served by SRS. Clients should use a WHEP-compatible player or the official SDK.

Target Users

Open API is designed for **enterprise system integration developers and solution architects** — teams responsible for embedding drone operational capability into existing platforms such as VMS, CAD, GIS, or PSIM systems.

The case for reusing FH2's operational layer

Enterprise integrations require more than device connectivity. Production-grade fleet management demands:

- **Stable flight control** — consistent command execution across varied field conditions
Reliable device state handling — accurate real-time status across mixed fleets
- **Mission scheduling** — coordinated task dispatch with operator-assisted or configured automatic execution paths
- **Flight safety enforcement** — platform-level policies governing pre-flight conditions, in-flight anomaly response, and return-to-home triggers; logic that must be comprehensive, field-validated, and consistently enforced across every mission

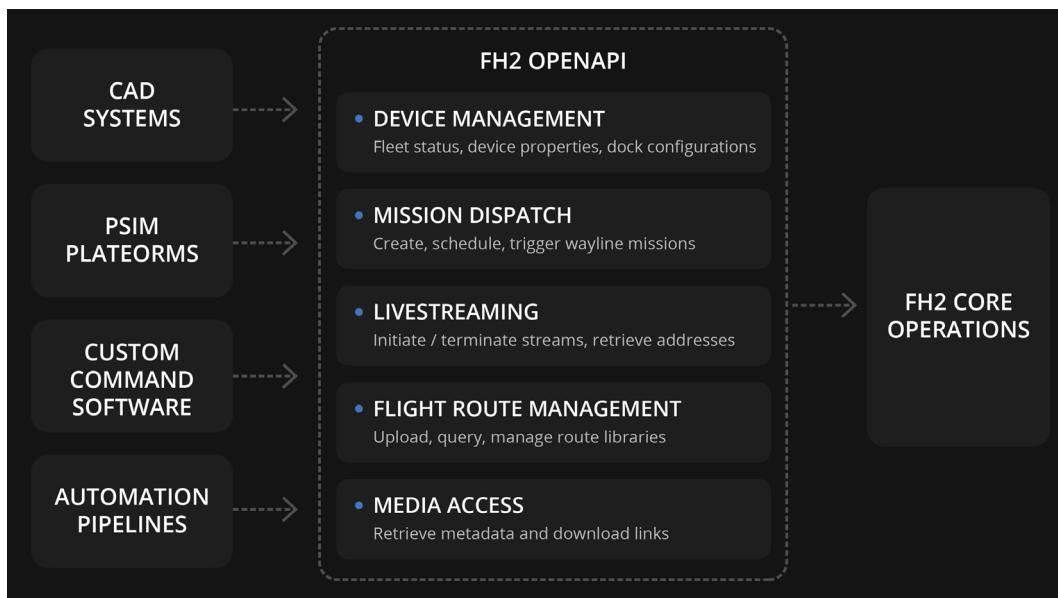


Figure 2.3.2-B Open API Architecture — Enterprise Platform Integration

Building these modules independently carries substantial development cost. Core stability — particularly in flight management, device handling, and safety enforcement — is difficult to achieve without extensive field validation.

Open API exposes FH2's production-tested operational modules as callable services.

Host platforms inherit that maturity directly — the enterprise system issues instructions via standard HTTP; FH2 handles the drone operational logic.

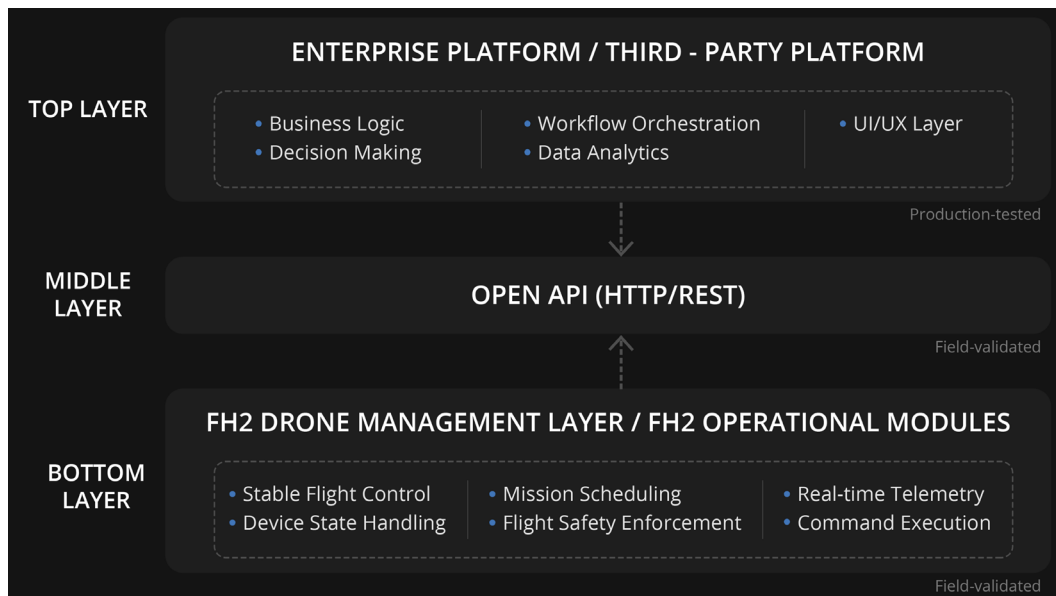


Figure 2.3.2-C Open API Data Model

2.3.3 Frontend Independent Components

Frontend Independent Components allow third-party platforms to embed FlightHub 2's operational UI directly — without rebuilding flight route creation, mission monitoring, or map management from scratch.

Third-party platforms can pull FH2's ready-built interface modules directly into their own system. The host platform keeps its existing look and feel; FH2 components slot into a defined area of the page and can be styled to match.

FlightHub 2 Frontend Components

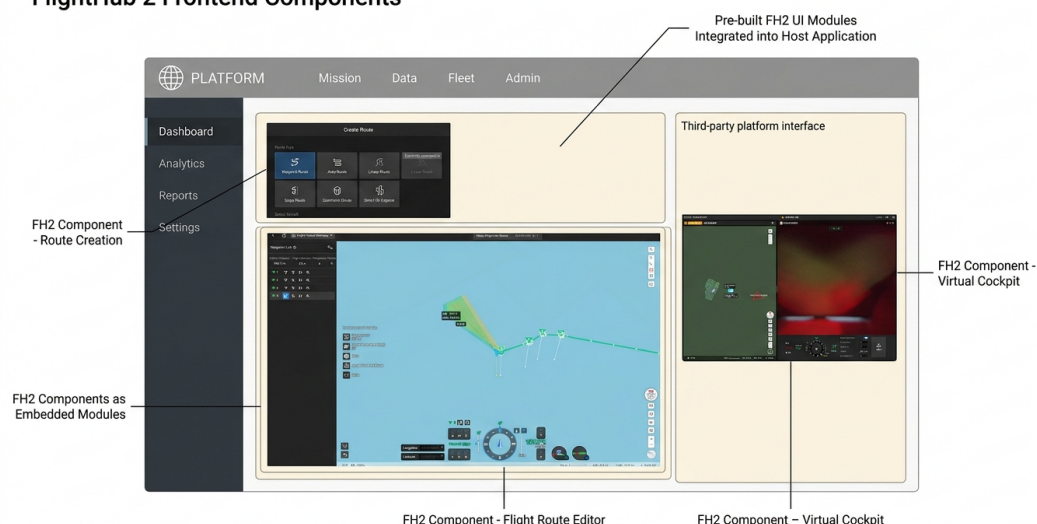


Figure 2.3.3-A Frontend Component Embedding — FH2 UI Modules Rendered Within a Third-Party Platform

Available Components

Four component modules are currently available for integration:

Component	Function
Virtual Cockpit	Real-time drone monitoring and remote control interface
Route Editor	Interactive flight route creation and waypoint configuration
Flight Records	Mission history, data playback, and operational analytics
Project / Map	Geospatial project management and map annotation display

Why Embed Rather Than Build

Drone operational interfaces — particularly Virtual Cockpit and Route Editor — require deep integration with real-time telemetry, device state, and mission control logic. Building these independently carries substantial cost with limited differentiation.

Embedding FH2’s components delivers three concrete advantages:

- **Operational maturity out of the box** — production-tested across live deployments; edge cases in telemetry rendering and route validation are already handled
- **Reduced integration surface** — drone UI complexity is fully encapsulated within FH2; the host platform connects once and inherits the capability
- **Consistent update cycle** — embedded components inherit FH2 capability improvements without additional development effort on the host side

Drone operational UI that behaves like a native part of the host platform — built and maintained by DJI.

Deployment Note

Frontend Components are available on on-premises deployments only. Public cloud deployments do not currently support this integration method. For full implementation

guidance, see the Frontend Components Documentation.

`paas.js` is hosted on the FlightHub 2 On-Premises server and accessed via the server's IP address or domain. No separate download or installation is required — the host application loads it remotely at runtime. For the complete resource loading and initialization guide, refer to the Frontend Components Documentation.

Frontend Components Documentation: <https://fh.dji.com/user-manual/en/custom-development/frontend-components/component-introduction.html>

2.4 Integration Interface Selection Guide

Selecting the right interface is the most consequential early decision in any FlightHub 2 integration project. The four sections below cover the integration scenarios most commonly encountered in DFR deployments — live streaming, telemetry, media archiving, and event notification — each with interface selection guidance, protocol selection, architecture patterns, and a phased implementation path.

Each section is self-contained and can be read independently based on your current integration priority. For teams starting from scratch, the recommended reading sequence follows the section order: establish the video feed first, then add telemetry, then automate media archiving, then close the loop with event-driven workflows.

If you are unsure which section applies to your current requirement, refer to the interface comparison in Section 2.3 before proceeding.

2.4.1 Live Streaming

The most common integrator requirement is to deliver one DJI-side live video stream into an existing VMS.

The central question for live streaming integration: your target platform needs real-time aerial video — which path gets it there most effectively?

Option 1 — FlightHub Sync: Live Stream Forwarding

If your target platform already accepts inbound video streams, configuration is all you need — read on to confirm this is the right fit.

Focused Scenarios

- An existing VMS or CAD system is deployed and needs to receive live drone footage. FlightHub Sync pushes the stream directly to the target platform via a configured RTMP or RTSP address — the receiving system requires no modification.
- During routine security patrols, operators initiate live streams from the FH2 dashboard in a single action; video appears on the VMS display wall in real time.

Best Fit Conditions

- **Multi-drone coordinated operations** — multiple drones airborne on coordinated patrol missions; the CAD system receives concurrent feeds from each unit to maintain a unified operational picture across the scene
- **Requirements still being defined** — the team is in the process of evaluating what drone video integration should look like in practice; Sync forwarding provides the lowest-cost path to validate integration value in real operational conditions before scope and development commitment are determined

When the requirement moves from aerial awareness to active event response — alarm-triggered dispatch, automatic stream initiation, coordinate push — Option 2 is the natural next step.

Option 2 — Open API: Live Streaming

When your platform needs to trigger, manage, and close video streams as part of its own event logic — rather than relying on manual initiation — read on to confirm Open API is the right fit.

Focused Scenarios

Event-driven response

Alarm received → CAD calls Open API → stream starts → playback address delivered to dispatcher. No manual steps. Stream closes when the mission ends. The same integration layer can grow into gimbal control and flight commands over time.

Deep workflow integration

Live video sits alongside alarm coordinates, mission status, and AI results in a single operator view. Stream addresses are generated per event and embedded in the dispatch interface — video as part of the workflow, not a tab opened separately.

Best Fit Conditions

- **Workflow is defined** — the team knows when video is needed, in what format, and who receives it
- **Development resource is available** — RESTful HTTP integration experience required
- **Video is the starting point, not the end goal** — the integration is expected to grow into broader on-scene interaction over time

Open API positions live streaming not as a standalone feed, but as one layer of a deeper operational integration — where drone capability becomes a native part of the host platform's event response chain.

RTMP vs RTSP vs WebRTC — Protocol Selection

Choosing the right streaming protocol is as important as choosing the right interface. The three protocols supported by FlightHub 2 serve different delivery models, target different platform types, and carry different latency profiles. Match the protocol to your target platform and operational requirement — not the other way around.

RTMP — Real-Time Messaging Protocol

RTMP is an **active push protocol** — the video source pushes the stream to a target server on demand, making it the natural fit for event-driven scenarios: the stream starts when the event triggers, and stops when it ends.

- **Focused scenarios:** CAD alarm-linked stream push, event-triggered DFR response
- **Select RTMP when:** the stream needs to start and stop in response to business events

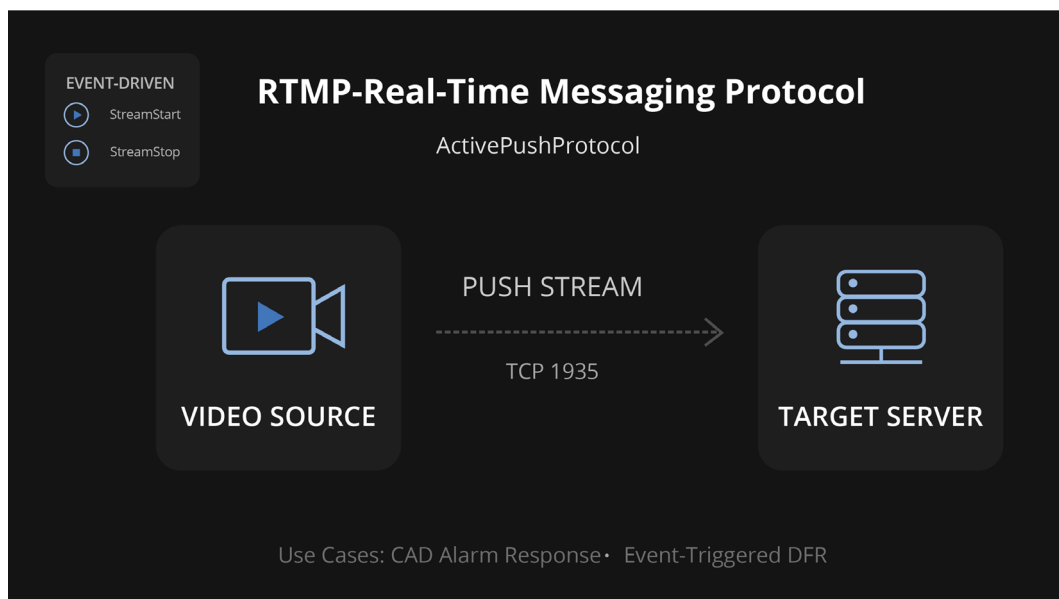


Figure 2.4.1-C RTMP Stream Push Topology

RTSP — Real-Time Streaming Protocol

RTSP operates on a **server-holds, client-pulls** model — the stream is held at the server and clients request it on demand. Multiple clients can pull the same stream concurrently, making it well-suited for scenarios where the same drone feed needs to reach several terminals simultaneously.

- **Focused scenarios:** environments with an existing VMS deployed, multi-terminal viewing of the same drone feed across a command center
- **Select RTSP when:** the target platform is a traditional VMS or IP camera management system, and concurrent multi-client access to the same stream is required

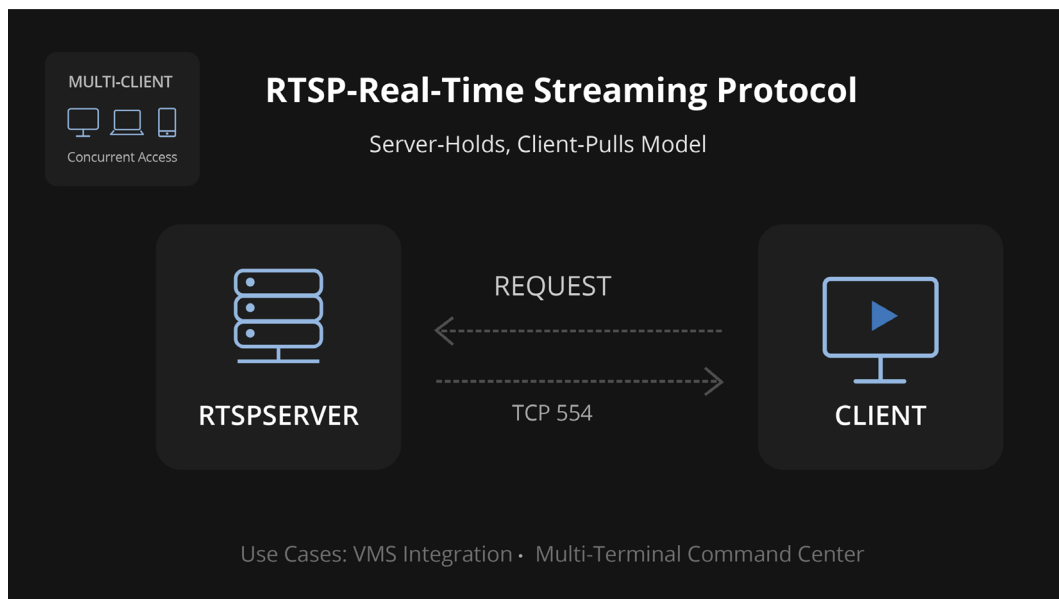


Figure 2.4.1-D RTSP Pull Stream Topology — Multi-Client Access

WebRTC — Web Real-Time Communication

WebRTC delivers drone live video directly to a browser with sub-500ms latency — no client installation, no plugin. Purpose-built for cloud environments, it is the protocol of choice when operators need near-real-time aerial video to support active on-scene decisions.

- **Focused scenarios:** web-based command platforms, cloud-deployed environments, latency-sensitive command and control interfaces
- **Select WebRTC when:** the target platform is a web application or cloud-deployed environment, and latency is a primary selection criterion

Protocol	Model	Latency	Primary Target Platform	Traditional VMS, IP camera systems
RTMP	Push	1–3s	CAD, security platforms	Web platforms, cloud environments
RTSP	Pull	1–3s	Traditional VMS, IP camera systems	FlightHub Sync
WebRTC	Push / P2P	<500ms	Web platforms, cloud environments	Open API

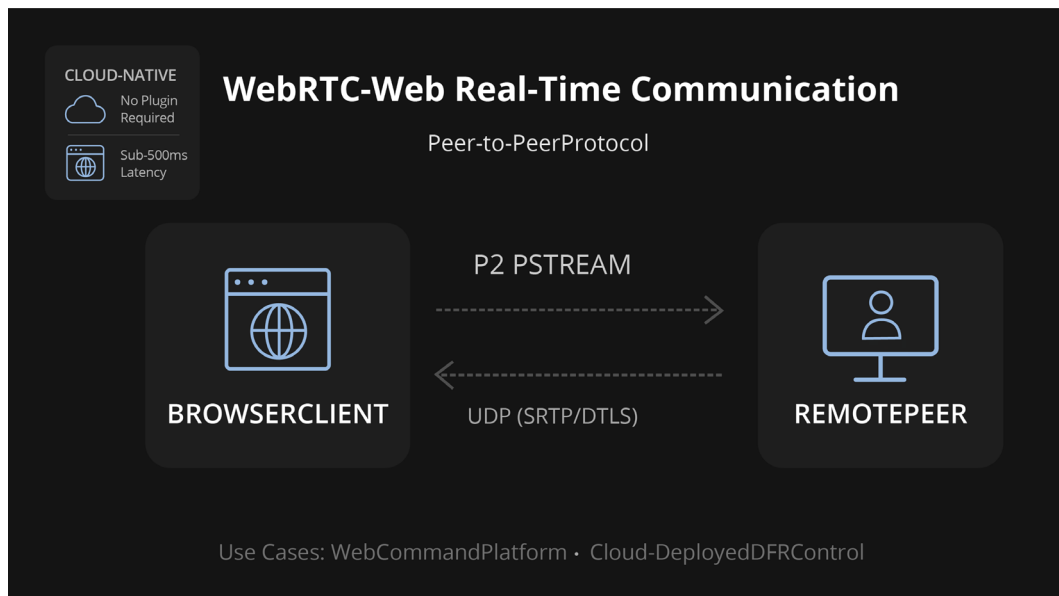


Figure 2.4.1-E WebRTC Stream Topology — Browser-Native Cloud Delivery

Live Streaming Integration Best Practice

Getting the interface and protocol right is necessary — but not sufficient. Before any development begins, the variables that drive architecture decisions need to be confirmed. Ambiguity at this stage is the most common source of integration rework.

Requirements Definition

Six variables determine how your live streaming integration should be architected. Confirming each one before selecting an approach reduces the risk of building toward the wrong target.

#	Dimension	Key Question	Architecture Impact
1	Concurrency	How many users or systems need to pull the stream simultaneously? What is the peak concurrent count?	Determines whether a media relay layer is required
2	Latency tolerance	What delay is acceptable — under 1s, 1–3s, or above 3s?	Drives protocol selection: WebRTC vs RTSP vs RTMP
3	Control requirements	Is video-only sufficient, or does the platform also need gimbal, camera, or flight control?	Significantly increases system complexity; control and video should be decoupled by design
4	Target platform capability	What video protocols does the receiving system support? Is custom development possible?	Determines whether a protocol conversion layer is needed
5	Deployment environment	Cloud, on-premises, or hybrid? Does the network cross public internet or NAT boundaries?	Affects stream path design, reachability, and stability planning
6	Security and compliance	Is encrypted transport required? Are authentication mechanisms or industry compliance standards in scope?	Shapes security architecture and access control design

Each of these variables has a direct downstream effect on architecture selection — treating them as optional pre-work is the fastest path to a redesign mid-project.



Figure 2.4.1-D WebRTC Stream Topology

Architecture Selection

Two architecture patterns cover the majority of live streaming integration scenarios.

The right choice follows directly from the requirements confirmed in the previous step.

Option A — Direct Streaming

Drone → FlightHub 2 → Target Platform

The target platform pulls the video stream directly from FlightHub 2. No intermediate media layer is required.

Conditions where this applies — all must be met:

- Concurrent stream count is low (≤ 3 simultaneous pulls)
- A single system is consuming the stream
- The target platform supports the protocol FlightHub 2 outputs natively
- The network environment is stable with no cross-public-internet or NAT constraints

Typical scenarios:

- A single operator workstation pulling a live drone feed for real-time situational monitoring
- An AI analytics engine consuming a single drone stream for on-the-fly object detection or behavior analysis
- A command center screen displaying one active drone feed during a DFR response, with no secondary consumer
- Early-stage integration validation — confirming stream reachability and playback stability before committing to a full architecture

Characteristics:

- Simple architecture, low deployment overhead, fast time to integration
- No scalability headroom for large-scale concurrent distribution — RTSP pull allows multiple clients to access the same stream without a relay layer in small-scale or demo scenarios, but is not recommended where concurrent consumer count is high or

stream delivery needs to reach multiple downstream systems reliably

Option B — Media Relay

Drone → FlightHub 2 → Media Server → Multiple Target Platforms

A media relay server sits between FlightHub 2 and the consuming platforms, handling stream distribution, replication, and protocol conversion.

Conditions — adopt if any one applies:

- Multiple systems need concurrent access to the same stream at scale — for small-scale validation or limited RTSP pull scenarios, FlightHub Sync can serve a limited number of clients directly; where live stream sharing spans multiple platforms, sites, or high concurrency requirements, a relay layer is required
- Protocol conversion is required between FlightHub 2 output and the target platform's supported format
- The network environment involves public internet traversal, NAT, or firewall constraints

Typical scenarios:

- A VMS display wall, a CAD operator interface, and an AI detection engine all consuming the same drone feed simultaneously at production scale — where stream delivery must be reliable across multiple systems or sites, a relay layer handles distribution and protocol conversion
- A target platform that requires a protocol different from the FH2-provided pull or forwarding address — requiring protocol conversion at the relay layer
- A deployment where the drone network and the command center are separated by public internet or firewall boundaries
- A multi-site security operation where the same aerial feed needs to reach command centers at different locations concurrently

Characteristics:

- Supports high-concurrency stream distribution across multiple consumers
- Handles protocol conversion and unified stream ingestion
- Architecture scales from single-node to cluster as demand grows

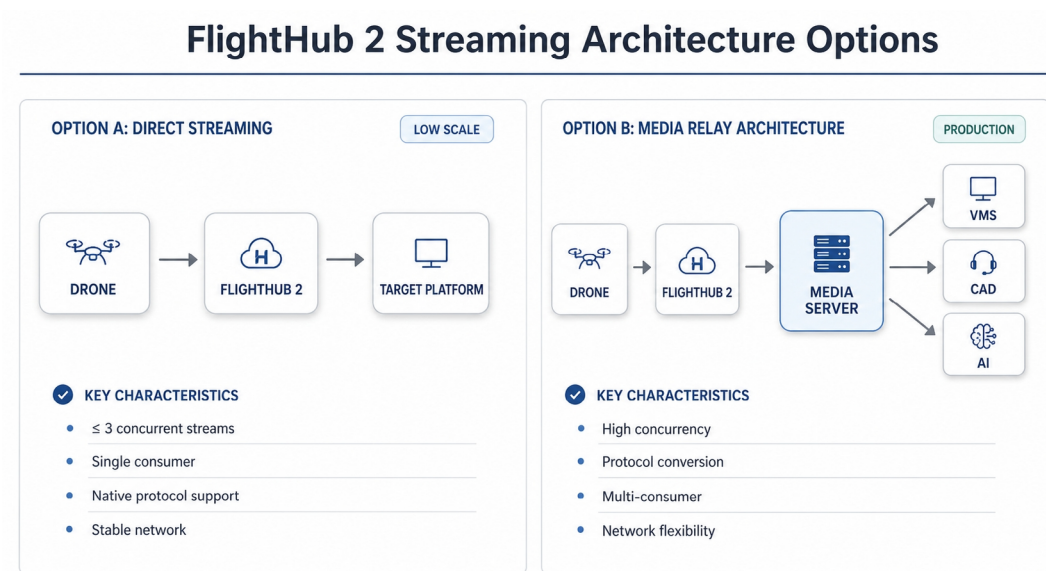


Figure 2.4.1-E Live Streaming Architecture Comparison — Direct vs Media Relay

Implementation Tip — Validate Before You Scale

The most common cause of live streaming integration failure is committing to a full architecture before protocol compatibility and stream stability have been confirmed. A two-phase approach reduces that risk without slowing the project down.

Phase 1 — Lightweight Validation

Before designing for scale, validate that the fundamental integration works.

3. Deploy a lightweight media relay service and complete protocol adaptation
4. Validate stream pull stability and playback continuity under realistic conditions
5. Confirm the target platform receives and renders the stream without modification

Open-Source Middleware Reference

The projects listed below have been referenced in FlightHub 2 live streaming integration contexts by the developer community. They are independent open-source contributions — not developed or maintained by DJI — and are shared here as a starting point for integrators exploring this path.

Phase 2 — Production Architecture

Once Phase 1 is complete and the integration path is validated, scale with confidence.

6. Assess concurrency and access patterns — how many concurrent streams are required, how many systems will consume them, and whether peak load follows predictable patterns (e.g., shift changes, incident spikes) or is continuous

7. Design the formal streaming architecture — single-node for contained deployments; cluster configuration for high-concurrency or multi-site scenarios where a single point of failure is not acceptable

8. Introduce development and operations resources — stream session management, health monitoring, automated restart on failure, and log retention for post-incident review

9. Apply security controls — encrypted transport (TLS/SRTP), token-based or IP-allowlist authentication, access scoping per system or user role, and any industry-specific compliance requirements identified during the requirements phase

Why this sequence matters

Approach	Risk
Build full architecture first, validate later	Protocol incompatibility discovered late; significant rework cost
Validate first, scale after confirmation	Incompatibility caught early; architecture decisions informed by real data

A lightweight validation stage costs a fraction of a redesign. The phase boundary is not a delay — it is a risk control point.

Decision Logic — Which Architecture Fits

Work through the checklist below. Check all conditions that apply to your project — the column with the most checks indicates the recommended architecture.

Condition	Direct Streaming	Media Relay
Single system consuming the stream	☐	
Multiple systems or users accessing concurrently		☐
Target platform supports FH2 output protocol natively	☐	
Protocol conversion required between FH2 and target platform		☐
Stable network, no public internet or NAT boundaries	☐	
Stream crosses public internet, NAT, or firewall		☐
Low concurrency (≤ 3 simultaneous pulls)	☐	
High concurrency or multi-site stream distribution		☐
Recommended Architecture	Direct Streaming	Media Relay

2.4.2 Telemetry

For most security integrators, live video is the first capability that comes to mind when evaluating drone integration. Telemetry is what makes that video operationally meaningful — and beyond situational awareness, real-time drone state data is a baseline safety requirement for autonomous aerial operations.

Why Telemetry Matters in DFR

Real-time drone telemetry is what transforms a drone from a remote camera into a

coordinated operational asset. Without a continuous data feed into the command platform, the drone operates as an isolated device — visible on screen, but invisible to the systems that manage the response.

Four operational capabilities depend directly on telemetry integration:

- **Unified resource dispatch** — when three drones are airborne simultaneously, the command center needs to know which unit is at 20% battery, which is two minutes from the scene, and which has just completed its mission. Without that data, dispatch decisions are guesswork
- **Airspace safety** — a drone that deviates from its route or loses signal needs to be caught by the platform, not noticed by an operator. Continuous telemetry is what enables the system to detect and respond to anomalies before they escalate
- **End-to-end response visibility** — from takeoff to return, ground commanders need live drone state to decide whether to hold position, redirect, or recall the unit
- **Cross-system coordination** — when drone position and status flow into CAD, GIS, or PSIM platforms, it stops being a separate device and becomes part of the command picture alongside officers and vehicles

For system integrators, telemetry integration is not an optional enhancement. It is the data layer that makes drone capability visible and actionable within existing command systems.

How MQTT Telemetry Works

MQTT (Message Queuing Telemetry Transport) is the protocol FlightHub 2 uses to deliver real-time drone telemetry to third-party systems. On Public Cloud, this is implemented as Telemetry Data Forwarding: FH2 acts as an MQTT client and pushes to a customer-managed broker. On On-Premises, this is implemented as MQTT Bridge: a broker-to-broker connection configured via POST /bridges/v1/resources. Both share the same thing/product/{sn}/... topic structure.

MQTT operates on a **publish-subscribe model** — the drone continuously reports its state, and any subscribed system receives updates the moment they arrive. No polling, no

repeated requests.

Think of it as a live broadcast: the station transmits continuously; any receiver tuned to that channel gets the feed in real time.

This makes MQTT well-suited for drone telemetry:

- Position data pushed at 0.5 Hz
- State changes delivered the moment they occur
- Receiving systems stay current with zero polling overhead

The Publish / Subscribe Model

- **Publisher** — FlightHub 2, pushing device telemetry to the Broker in real time
- **Broker** — the message routing hub; decouples the publisher from subscribers and manages topic distribution
- **Subscriber** — the third-party platform, receiving data on subscribed topics without polling

Topic Structure

FlightHub 2 publishes all device data to the **MQTT Broker** — the central router that forwards each message to every subscribed system. Multiple platforms can consume the same data stream simultaneously without any awareness of each other.

Each **Topic** defines what a subscriber receives. Topics follow a structured path with the device serial number as a variable — subscribe to a single device, or use wildcards to cover the entire fleet.

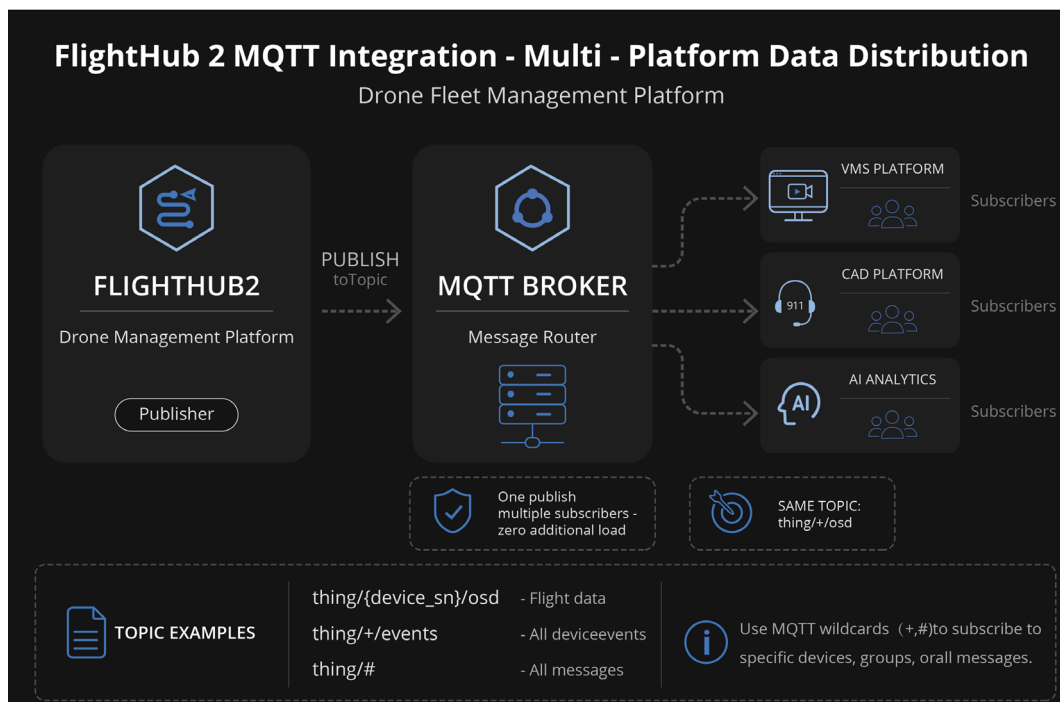


Figure 2.4.2-A MQTT Telemetry Architecture — FlightHub 2 to Third-Party Platform

Topic	Direction	Frequency	Content
thing/product/{sn}/osd	Device → Cloud	0.5 Hz	Position, attitude, velocity, battery
thing/product/{sn}/osd	Device → Cloud	0.5 Hz	Aircraft-level telemetry
thing/product/{sn}/state	Device → Cloud	Low	Flight mode, storage status, device properties
thing/product/{sn}/events	Device → Cloud	Event-driven	Device online/offline, mission complete, alerts

For the full topic reference, see the DJI Cloud API — MQTT Topic Definition.

Integration Scenarios

Understanding how MQTT works is the foundation — knowing which data to subscribe to for your specific use case is what makes the integration operationally useful. The three scenarios below represent the most common telemetry integration requirements in a DFR deployment. Each has a distinct data need, a corresponding set of topics, and a clear operational justification.

Scenario 1 — Real-Time Drone Coordinate Push During DFR Response

During an active DFR response, the command platform needs continuous drone position data to:

- Correlate aerial coverage with ground unit positions
- Assess arrival time and scene coverage in real time
- Maintain a unified operational picture across all responding units

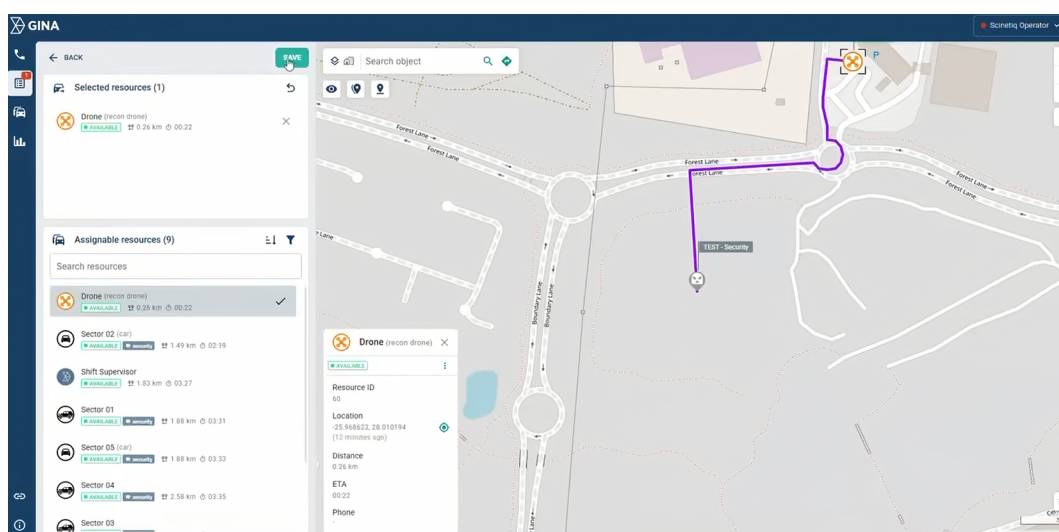


Figure 2.4.2-B DFR Real-Time Coordinate Push — Drone Position on CAD/GIS Map

Subscribe to:

- thing/product/{sn}/osd — position, altitude, velocity, heading at 0.5 Hz
- thing/product/{sn}/osd — aircraft-level telemetry

Scenario 2 — Fleet-wide Situational Awareness at the Command Center

When multiple drones are operating concurrently, the command center needs a live view of every unit's position, flight mode, battery status, and mission state — the data layer that makes dispatch decisions possible:

- Identify available units and current task assignments
- Flag units approaching battery limits before they become a constraint
- Detect deviations from assigned coverage areas in real time

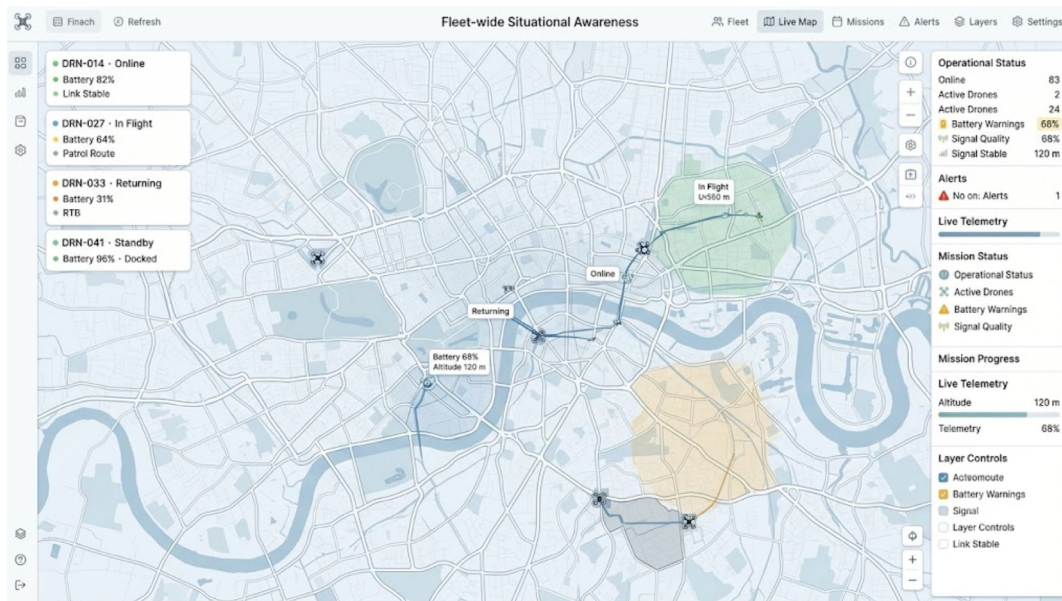


Figure 2.4.2-C Fleet-wide Situational Awareness — Multi-Drone Real-Time Status Display

Subscribe to:

- `thing/product/{sn}/osd` — real-time position and battery across all active units
- `thing/product/{sn}/state` — flight mode, storage status, connection state
- `thing/product/{sn}/events` — mission complete, device online/offline, return-to-home triggered

Scenario 3 — Device Health Management and Alerting

Proactive device management requires continuous state monitoring — battery degradation, signal quality, and firmware compliance are all detectable from the telemetry stream before they become operational failures.

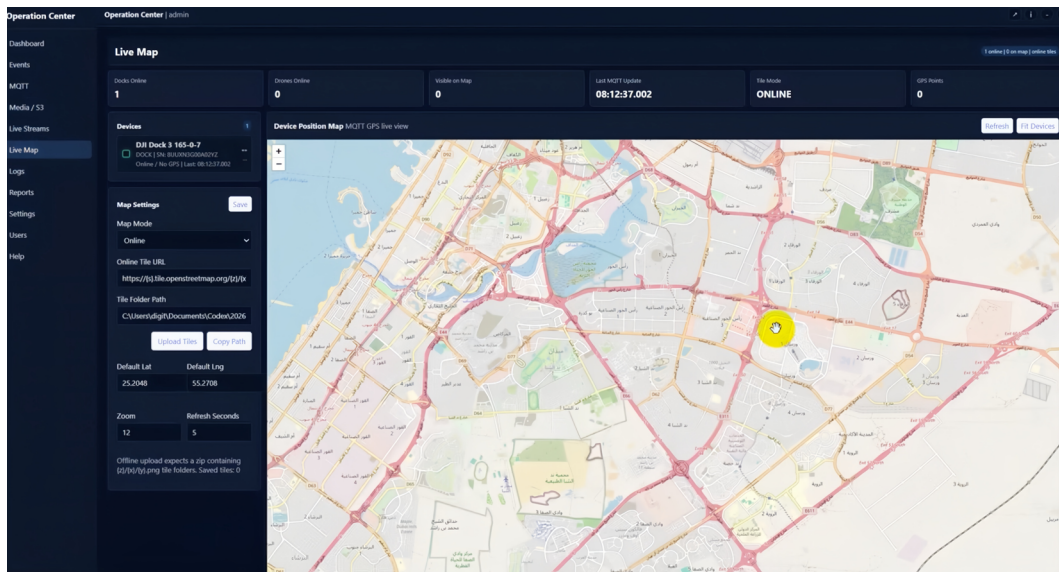


Figure 2.4.2-D Device Health Management — Proactive Monitoring via Telemetry Stream

Subscribe to:

- thing/product/{sn}/state — battery level, signal strength, firmware version, storage status
- thing/product/{sn}/events — device anomalies, low battery alerts, signal loss, device offline

Full MQTT Topic Reference

The three scenarios above cover the most common integration requirements. For a complete listing of all available topics — including payload structures, field definitions, and data direction — refer to the official topic reference: [DJI Cloud API — MQTT Topic Definition](#).

Telemetry Integration Best Practice — When Your Platform Does Not Support MQTT in 24/7 Operations

The three scenarios above assume the target platform can subscribe to MQTT directly. In practice, many enterprise systems — particularly legacy CAD, VMS, and device management platforms — do not have native MQTT client capability. The recommended path in this case is a Middleware Bridge.

Middleware Bridge

Deploy a lightweight middleware component that subscribes to FlightHub 2 MQTT topics and converts the data stream into a format the target platform supports. Select the conversion path based on the target platform type and the scenario being served:

Solution	Focused Scenarios	Implementation	Recommendation
MQTT → WebSocket	DFR real-time coordinate push; fleet-wide position display on GIS map or command screen	Middleware subscribes to OSD topics and forwards via WebSocket long connection to frontend; no polling required	★★★★★
MQTT → Redis / Database	Device health trend analysis; mission data archiving; firmware compliance reporting	High-frequency OSD written to Redis for real-time reads; low-frequency state and events persisted to database for reporting and audit	★★★★
MQTT → Kafka	City-scale multi-system distribution; large fleet concurrent operations requiring high-throughput fan-out to CAD, GIS, AI platforms simultaneously	Middleware writes to Kafka topics; each downstream system consumes independently as a Kafka consumer	★★★★ (large-scale)

Requirement Discovery Template

Use this template during the pre-integration discovery phase to confirm the variables that drive architecture and protocol selection.

#	Dimension	Question	Client Response	Decision Impact
A1	Data consumption model	Is continuous real-time telemetry required, or periodic device status queries?		MQTT subscription vs middleware path
A2	Latency tolerance	What data delay is acceptable? (<1s / 1-5s / >5s)		Protocol and architecture selection
A3	Platform MQTT capability	Does the target platform support native MQTT subscription?		Whether middleware conversion is required
A4	Data type requirements	Which data is needed: position / flight state / battery / mission progress / event notifications?		Topic subscription scope
A5	Concurrent device scale	How many devices will be online simultaneously?		Architecture capacity and design
A6	Target platform type	CAD / GIS / VMS / AI platform / database / other?		Middleware conversion path selection
A7	Network environment	Does the stream cross public internet or NAT? Are there firewall restrictions?		Broker reachability and link design
A8	Security and compliance	CAD / GIS / VMS / AI platform / database / other? Does the stream cross public internet or NAT? Are there firewall restrictions?		Security architecture design

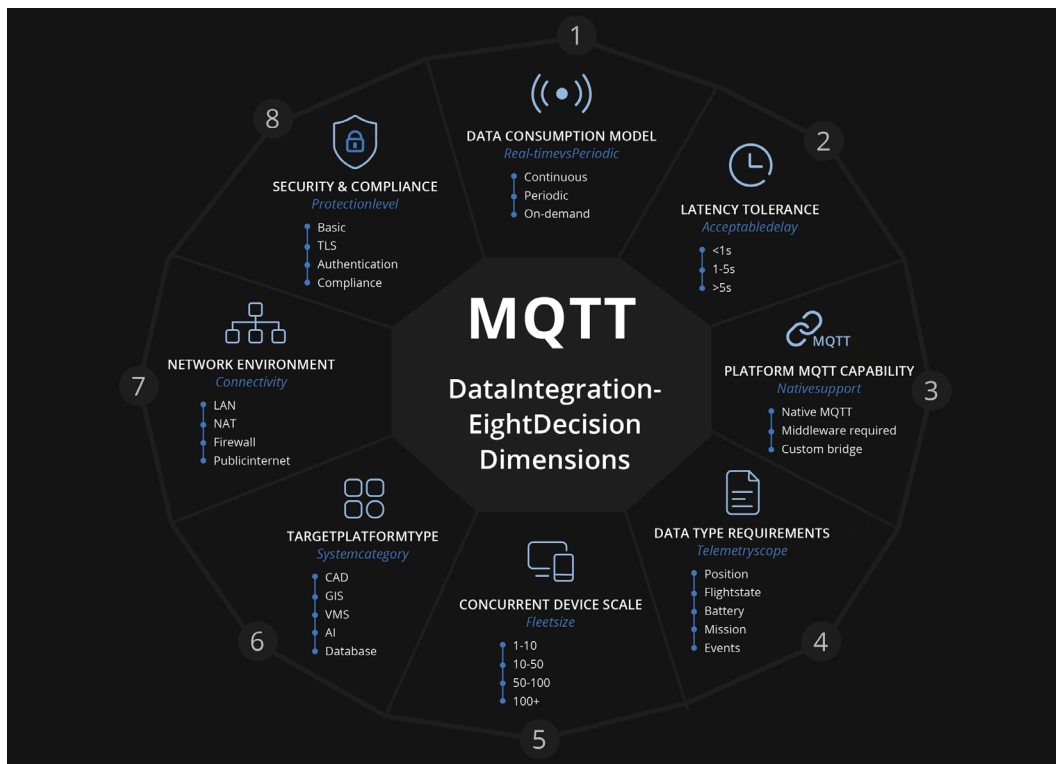


Figure 2.4.2-E File Sync Flow — Automated Media Transfer to S3 Storage

2.4.3 Media

A DFR response generates more than situational awareness — every mission produces imagery, video, and spatial data that carry operational and evidentiary value long after the drone has returned to its dock. How that data moves from FlightHub 2 into existing enterprise systems determines whether it becomes a usable asset or an archived file no one can find.

From the Field to the System — Making Mission Media Work

Mission media is the evidentiary and analytical record of every DFR response — but its value is only realized when it reaches the right systems, automatically and on time.

- **Evidence preservation** — mission footage is the primary record for incident reconstruction and enforcement support; it needs to reach the evidence management system intact, without manual handling
- **Operational debrief** — photos and video provide an objective record of scene conditions, supporting command-level review of response quality and coverage
- **3D model and orthomap integration** — Cloud Modeling outputs feed directly into GIS platforms for spatial analysis and site assessment

- **Scale** — in multi-mission operations, manual media handling is not a viable workflow; automated sync is the foundation of a scalable media pipeline

Media integration is the difference between a drone that generates footage and one that generates organizational intelligence.

FlightHub Sync: File Synchronization

If your target storage supports S3-compatible ingest, automated media sync requires no development — read on to confirm this is the right fit.

Focused Scenarios

- Mission completes; photos, videos, and flight logs are automatically pushed to the configured S3 storage endpoint without any operator action. Files are available in the target system within minutes of the drone returning to the dock.
- Cloud Modeling outputs — orthomosaics and 3D models — sync automatically to a designated storage path, ready for import into the GIS platform without manual transfer.

Data Type	Sync Direction	Notes
Media Files (Photos / Videos)	FH2 → Third-party storage	One-way
Flight Routes	FH2 ↔ Third-party storage	Two-way
3D Models	FH2 → Third-party storage	One-way
Flight Logs	FH2 → Third-party storage	One-way
Custom Zones & Annotations	FH2 ↔ Third-party storage	Two-way

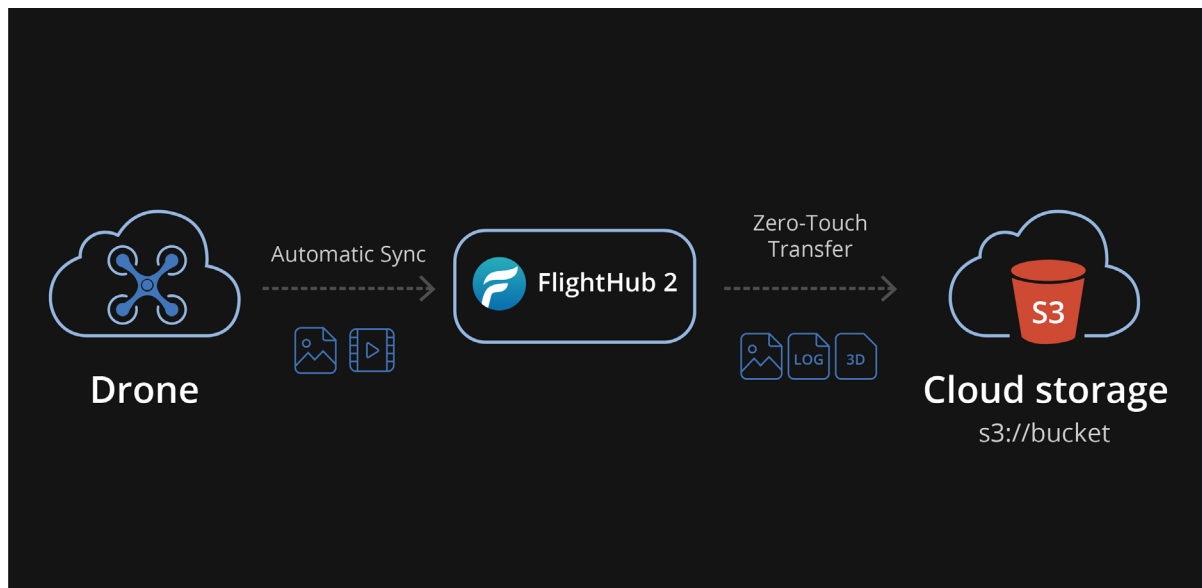


Figure 2.4.3-A FlightHub Sync — File Synchronization to S3 Storage

Best Fit Conditions

- **Storage is S3-compatible** — the target endpoint is AWS S3, Alibaba Cloud OSS, or a self-hosted S3-compatible service such as MinIO
- **No post-processing required** — files are archived as-is; the integration requirement is transfer and storage, not transformation
- **Requirements still being validated** — the team needs to confirm that automated media flow works end-to-end before committing to a deeper integration architecture

When the requirement moves beyond storage — filtering by mission, binding media to business events, or triggering downstream workflows on file arrival — a middleware-assisted architecture is the natural next step.

S3 Protocol Primer

S3 (Simple Storage Service) is the object storage protocol that underpins FlightHub Sync's file synchronization capability. Originally introduced by AWS, S3 has become the de facto standard for cloud object storage — supported natively by major cloud providers and self-hosted storage platforms alike.

Core Concepts

Concept	Definition
Bucket	The top-level container for stored objects — equivalent to a root directory. Each sync target is configured as a bucket endpoint
Object	Any file stored in the bucket — photo, video, flight log, or 3D model. Each object is addressed by a unique key within the bucket
Prefix	A path segment used to organize objects within a bucket — functions like a folder structure (e.g., /missions/2026-05/dock-01/)

Compatible Storage Targets

FlightHub Sync File Synchronization supports any S3-compatible endpoint. Confirmed compatible targets include:

- **AWS S3** — public cloud, global availability
- **Alibaba Cloud OSS** — public cloud, China and international regions
- **Self-hosted S3-compatible endpoints** — MinIO and other S3-compatible solutions for on-premises or air-gapped deployments

If your storage platform exposes an S3-compatible API, it can serve as a sync target — no additional configuration on the FH2 side is required beyond the endpoint address and credentials.

Media Integration Best Practice — When Your System Does Not Support S3

FlightHub Sync File Synchronization is built on S3 protocol. When the target system does not expose an S3-compatible endpoint, direct sync is not available — but the media data pipeline can still be established through alternative paths.

Understanding why this gap exists helps scope the integration correctly. Most enterprise VMS and CAD platforms were designed before S3 became the dominant storage standard, and their media ingest models reflect that:

- **VMS platforms** — predominantly store video via proprietary database formats or local/NAS-based filesystem paths. Media ingest is typically handled through folder-

watch mechanisms or camera driver interfaces, not object storage APIs

- **CAD platforms** — are primarily workflow and dispatch systems, not media storage platforms. Media attachments are typically linked via URL reference or uploaded through a proprietary document management interface; native S3 ingest is uncommon
- **Evidence management systems** — increasingly adopting cloud-native architectures with S3-compatible endpoints, but many on-premises deployments still rely on local file ingest or proprietary upload APIs

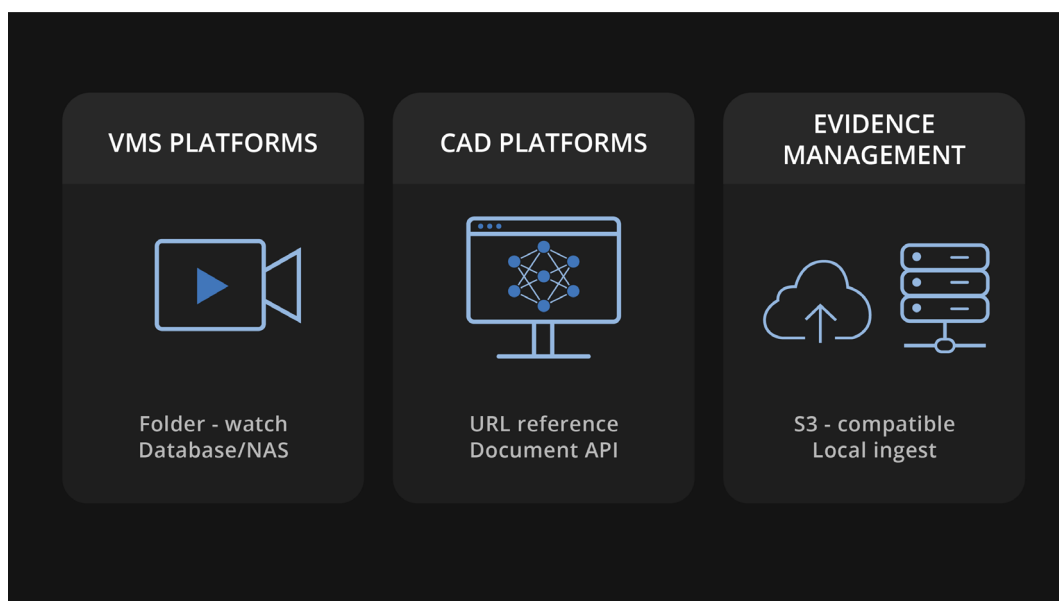


Figure 2.4.3-B FlightHub Sync — S3 Integration Overview

FlightHub 2 Sync to AWS S3 — Cloud-to-Cloud Integration Best Practice

1. AWS S3 & IAM — Component Overview

Amazon S3 (Simple Storage Service) is an object storage service that organizes files as objects within named buckets, each identified by a unique key and accessible via a standard S3-compatible API. In a FlightHub 2 Sync deployment, the S3 bucket serves as the landing target for all mission media — photos, videos, and AI result files — written automatically upon mission completion.

AWS IAM (Identity and Access Management) controls access to AWS resources. An IAM user carries a credential pair — Access Key ID and Secret Access Key — used to authenticate

API requests. Permissions are defined in an IAM Policy and attached to the user, governing exactly which operations are permitted on which resources.

The relationship in this integration is straightforward:

The IAM user holds the credentials that FlightHub 2 Sync uses to authenticate against AWS
The IAM policy attached to that user defines exactly which S3 operations are permitted — and on which bucket

FlightHub 2 Sync uses those credentials to write mission media directly into the configured S3 bucket via the S3 API

No other AWS services are required for this integration. The IAM user needs write access to one bucket; nothing beyond that scope should be granted.

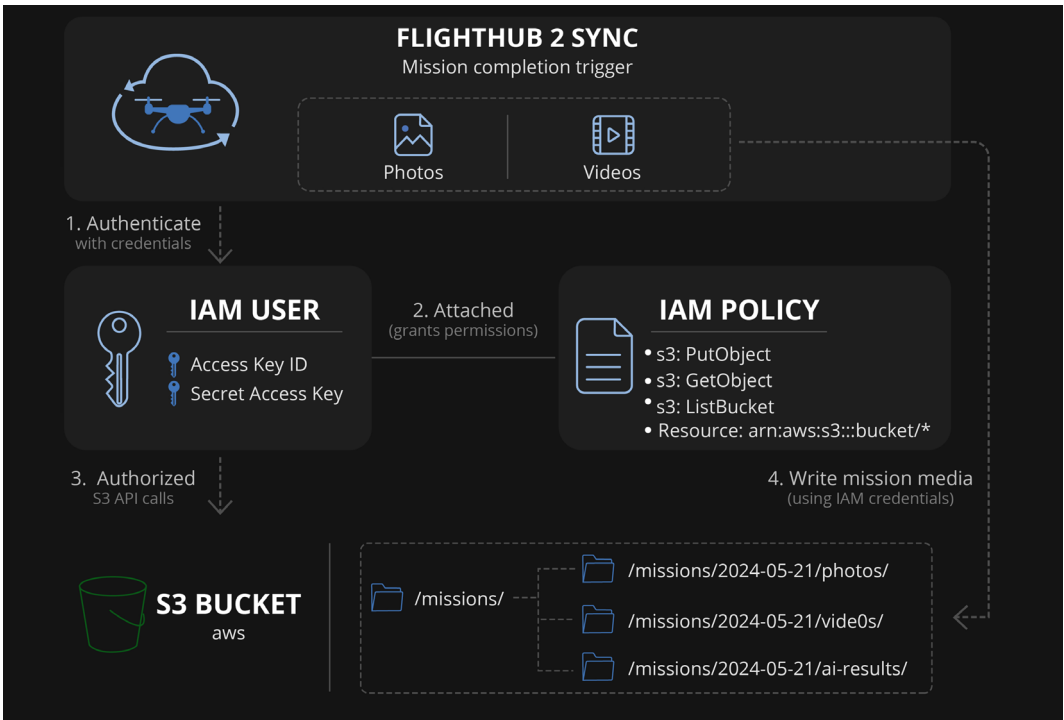


Figure 2.4.3-C FlightHub 2 Sync — S3 & IAM Relationship Diagram

Step 1 — Create an S3 Bucket

In the AWS Console, navigate to S3 → Buckets → Create bucket. Select General purpose as the bucket type and enter a unique bucket name. Select the AWS Region closest to your FlightHub 2 deployment, then click Create bucket.

Create bucket [Info](#)

Buckets are containers for data stored in S3.

General configuration

AWS Region
Asia Pacific (Tokyo) ap-northeast-1

Bucket type [Info](#)

☒ **General purpose**
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

☐ **Directory**
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket namespace
Choose the namespace where you want to create your bucket. [Learn more](#)

☒ **Global namespace**
By default, S3 creates general purpose buckets in the global namespace.

☐ **Account Regional namespace (recommended)**
General purpose buckets created in your account Regional namespace are unique to your account. These buckets can never be created by another AWS account.

Bucket name [Info](#)

DFR-test

Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn more](#)

Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.

Figure 2.4.3-D AWS S3 — Create Bucket Configuration

Step 2 — Create an IAM Policy

Navigate to IAM → Policies → Create policy. In the Policy editor, select JSON and paste the permission set from the Permission Configuration Code Sample link in the FlightHub 2 Sync storage panel. Click Next.

Specify permissions [Info](#)

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

Visual **JSON** Actions [Visual](#) [JSON](#) [Actions](#)

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "Statement1",
6       "Effect": "Allow",
7       "Action": [],
8       "Resource": []
9     }
10  ]
11 }

```

Edit statement Statement1 [Remove](#)

Add actions

Choose a service

Available

- AI Operations
- AMP
- API Gateway
- API Gateway V2

Figure 2.4.3-E AWS IAM — Create Policy (JSON Editor)

Add Storage Configuration

Bucket Type: AWSS3

Bucket Name: eg: ***-name
Used to identify this bucket in rule configuration

Bucket AK: eg: AKIAQCU5PAC4***

Region: us-east-1

Permission Configuration Code Sample

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Statement1",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3::your-bucket-name"
    },
    {
      "Sid": "Statement2",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectTagging",
        "s3:PutObjectTagging",
        "s3:PutObject",
        "s3:AbortMultipartUpload",
        "s3:ListMultipartUploadParts"
      ],
      "Resource": "arn:aws:s3::your-bucket-name/*"
    }
  ]
}
```

Buttons: Cancel, Save Configuration

Figure 2.4.3-F FlightHub 2 Sync — Permission Configuration Code Sample

On the Review and create page, enter a policy name and click Create policy.

Review and create Info

Review the permissions, specify details, and tags.

Policy details

Policy name
Enter a meaningful name to identify this policy.
DFR@
Maximum 128 characters. Use alphanumeric and '+', '@', '-' characters.

Description - optional
Add a short explanation for this policy.
goodday
Maximum 1,000 characters. Use alphanumeric and '+', '@', '-' characters.

Permissions defined in this policy Info
Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, group, or role), use the IAM console or the AWS CLI.

Buttons: Edit

Figure 2.4.3-G AWS IAM — Policy Review and Create

Step 3 — Create an IAM User and Attach the Policy

Navigate to IAM → Users → Create user. Enter a user name and leave console access disabled. Click Next.

Step 1
● Specify user details
○ Step 2
○ Step 3
○ Review and create

Specify user details

User details

User name
DFR Test

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

☐ Provide user access to the AWS Management Console - optional
In addition to console access, users with `SignInLocalDevelopmentAccess` permissions can use the same console credentials for programmatic access without the need for access keys.

① If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

Cancel Next

Figure 2.4.3-H AWS IAM — Create User: Specify User Details

Select Attach policies directly, find and select the policy created in Step 2, then click Next.

Step 1
○ Specify user details
● Set permissions
○ Step 3
○ Review and create

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

- ☐ Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
- ☐ Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.
- ☒ Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1495)
Choose one or more policies to attach to your new user.

Search Filter by Type

Policy name	Type	Attached entities
fh2-sync-test	Customer managed	Permissions policy

Create policy

Figure 2.4.3-I AWS IAM — Create User: Set Permissions

Confirm the policy appears under Permissions summary and click Create user.

Step 1
○ Specify user details
○ Step 2
● Review and create

Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

User name DFR Test	Console password type None	Require password reset No
-----------------------	-------------------------------	------------------------------

Permissions summary

Name	Type	Used as
fh2-sync-test	Customer managed	Permissions policy

Tags - optional
Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

CloudShell Feedback Console Mobile App © 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Figure 2.4.3-J AWS IAM — Create User: Review and Create

Step 4 — Generate an Access Key

Open the IAM user and click Create access key. Select Third-party service as the use case.

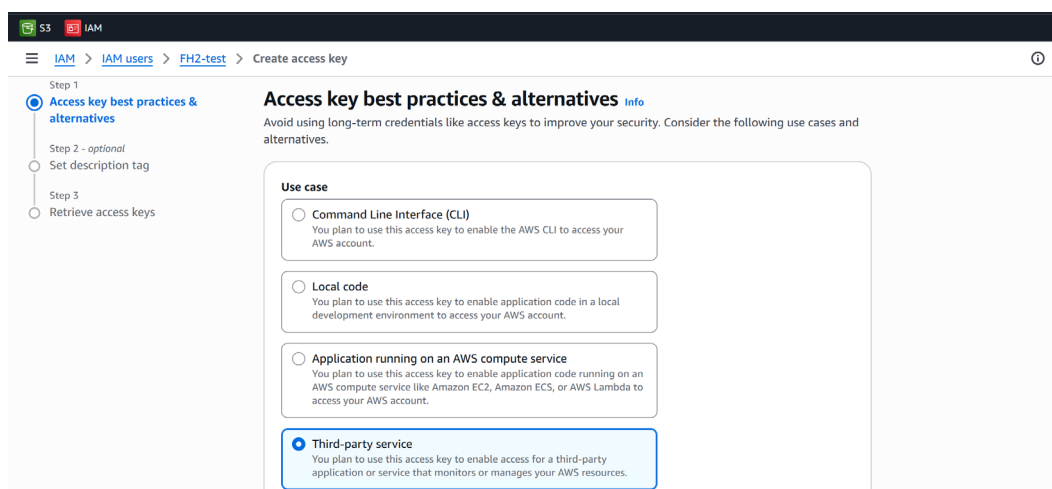


Figure 2.4.3-K AWS IAM — Create Access Key: Third-Party Service

Copy both the Access key and Secret access key immediately — the secret is shown only once.

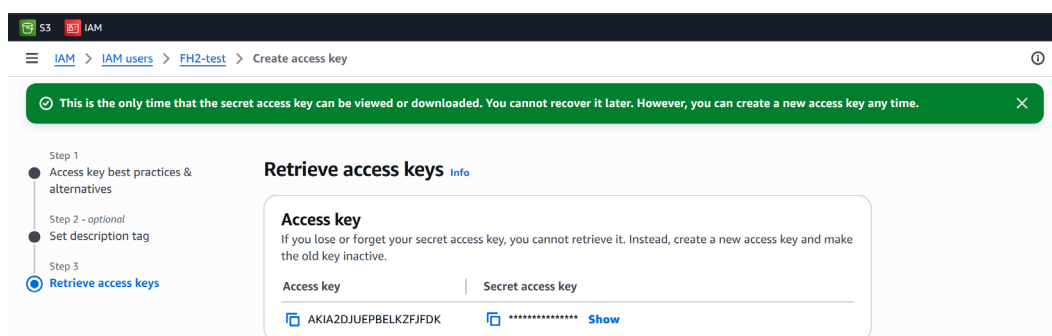


Figure 2.4.3-L AWS IAM — Retrieve Access Keys

In FlightHub 2, go to FlightHub Sync → Storage Configuration → Add Storage Configuration. Set Bucket Type to AWSS3 and fill in the bucket name, Access Key (AK), Secret Key (SK), and Region. Optionally set a Preset Path to organize synced files within the bucket.

Edit Storage Configuration

Bucket Type

AWSS3

Bucket Name

rectron-training

Used to identify this bucket in rule configuration.

Bucket AK

AKIA2DJUEPBENEBAHNXB

Bucket SK

eeQU+KhAnbsMnSpp235kWAojAL80WqZHgFWvCMzV

Region

ap-northeast-1

Preset Path

Optional Configuration

[Permission Configuration Code Sample](#)

Connectivity Check

Connected

Cancel

Delete Configuration

Save Configuration

Figure 2.4.3-M FlightHub 2 Sync — Add Storage Configuration

Click Connectivity Check. A Connected status confirms successful authentication. Click Save Configuration.

Edit Storage Configuration

Bucket Type

AWSS3

Bucket Name

rectron-training

Used to identify this bucket in rule configuration.

Bucket AK

AKIA2DJUEPBENEBAHNXB

Bucket SK

eeQU+KhAnbsMnSpp235kWAojAL80WqZHgFWvCMzV

Region

ap-northeast-1

Preset Path

Optional Configuration

[Permission Configuration Code Sample](#)

Connectivity Check

Connected

Cancel

Delete Configuration

Save Configuration

Figure 2.4.3-N FlightHub 2 Sync — Storage Configuration: Connected

In FlightHub 2, create and execute a test mission to generate media output.

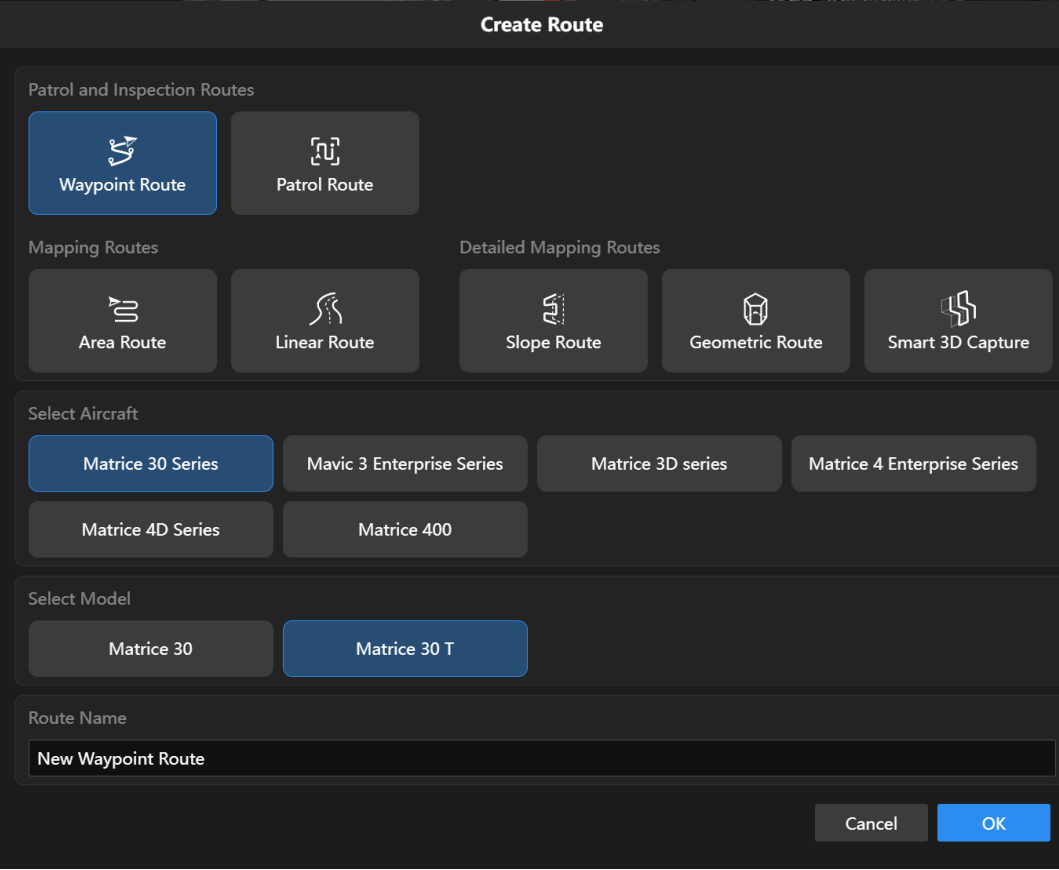


Figure 2.4.3-O FlightHub 2 — Create Route for Verification Mission

After mission completion, open the S3 bucket in the AWS Console and navigate to the sync path. Confirm that mission files have landed with timestamps matching the mission completion time.

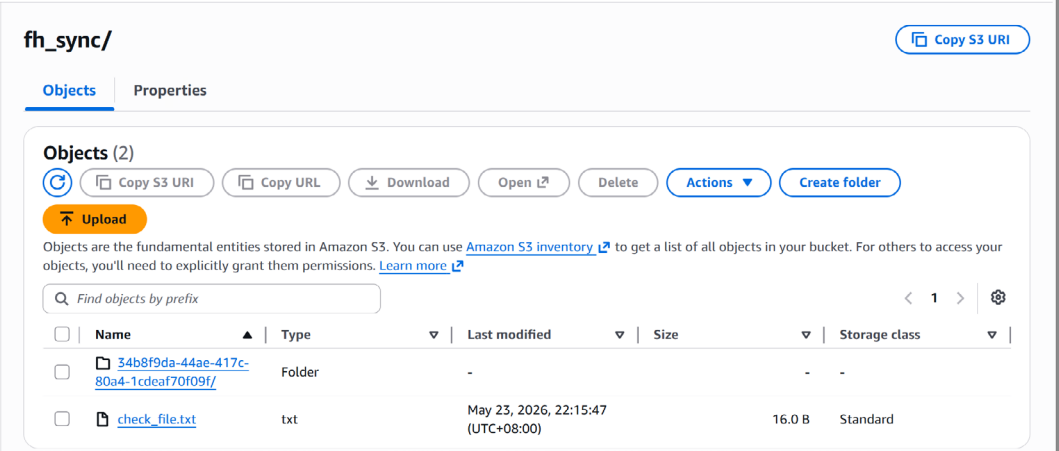


Figure 2.4.3-P AWS S3 — Sync Files Landed in Target Bucket

The practical implication: for most existing VMS and CAD deployments, a direct S3 sync from FlightHub 2 will not reach the target system without an intermediate step. The paths below address that gap.

Path 1 — S3-to-Local Filesystem Mount

Mount the S3 bucket as a local filesystem directory on the target server. The target system reads files as if they were stored locally — no S3 client integration required on the application side.

- **Suited for:** on-premises systems with local file access capability but no S3 API support
- **Typical targets:** legacy evidence management systems, local NAS-based archives, on-premises VMS with folder-based media ingest
- **Consideration:** performance depends on network bandwidth; not recommended for high-frequency, large-volume transfers without adequate bandwidth planning

```
# Create local mount directory
```

```
# This directory will be used as the local filesystem entry point for the S3 bucket  
mkdir -p /mnt/s3bucket
```

```
# Mount S3 bucket to local filesystem
```

```
# Applications can access S3 objects transparently via standard file I/O operations
```

```
s3fs your-bucket-name /mnt/s3bucket \
```

```
-o passwd_file=/etc/passwd-s3fs \
```

```
-o url=https://s3.amazonaws.com \
```

```
-o use_path_request_style \
```

```
-o allow_other \
```

```
-o nonempty
```

```
# Verify that the filesystem is successfully mounted
```

```
# Confirms the S3 bucket is correctly mapped to the local mount point
```

```
df -h | grep s3bucket
```

```
# Validate object visibility through filesystem interface
# Ensures basic read access to the mounted S3 bucket
ls /mnt/s3bucket
```

Path 2 — Middleware-Assisted Transfer

Deploy a lightweight middleware component that connects to the S3 bucket, retrieves files on a schedule or trigger, and delivers them to the target system via its native interface.

- **Suited for:** systems with specific ingest interfaces that cannot be adapted to S3 directly
- **Typical targets:** platforms with proprietary ingest APIs
- **Consideration:** middleware introduces an additional component to maintain; scope the transfer logic carefully to avoid duplicating files or missing new uploads

Selection Guide

Condition	Path 1 S3 Mount	Path 2 Middleware
Target system has local filesystem access	☐	
Target system has proprietary ingest interface		☐
Production-grade automation required	☐	☐
Minimal development resource available	☐	

Requirement Discovery Template

#	Dimension	Question	Client Response	Decision Impact
A1	Data type	Which file types are required — photos / videos / flight logs / 3D models / orthomosaics?		Sync scope and storage planning
A2	Target storage	Does the destination support S3-compatible ingest?		Direct Sync vs alternative path

#	Dimension	Question	Client Response	Decision Impact
A3	Target system interface	How does the target system ingest files — S3 / local folder / API / other?		Path selection
A4	File structure	Are there naming conventions or directory structure requirements?		Prefix, path, and middleware logic
A5	Processing requirement	Do files need transformation or metadata tagging before delivery?		Whether middleware logic is required
A6	Volume and retention	Expected daily data volume? Retention requirements?		Storage capacity and lifecycle policy
A7	Security and compliance	Encrypted transfer required? Access controls or retention policies applicable?		Security architecture and compliance design

2.4.4 Event API

Telemetry tells you what the drone is doing. Media captures what it saw. Event API closes the loop — pushing structured notifications to third-party systems the moment something significant happens inside FlightHub 2, so downstream platforms can act without polling and with reduced manual handoff.

Why Event API Matters in DFR

In a DFR deployment, the most time-critical actions are triggered by events, not people. A mission completes, a media file becomes available, a device exits return-to-home unexpectedly — each moment has a downstream consequence that the command platform, evidence system, or operations team needs to act on without delay.

Without event notification, third-party systems must either poll repeatedly and hope they catch the state change in time, or wait for manual intervention. Neither is viable in a live response environment.

Event API eliminates that gap:

- **From awareness to action** — telemetry and media data tell you what happened; Event API tells downstream systems what to do next. It is the trigger layer that connects FH2 operational events to enterprise business logic
- **No polling overhead** — rather than querying FH2 at fixed intervals, subscribed systems receive a push notification the moment the event occurs; system load drops, response latency drops with it
- **Complete DFR workflow closure** — mission complete, media ready, device anomaly, stream address changed: every operational node that requires a downstream action can be driven by an event notification rather than a human decision

Event API is what makes a DFR deployment feel automated rather than monitored.

How Event API Works

Event API operates on an HTTP push model — FlightHub 2 sends a POST request to a configured endpoint the moment a subscribed event occurs. No persistent connection is required; the third-party platform simply exposes a receivable URL and defines which events to subscribe to.

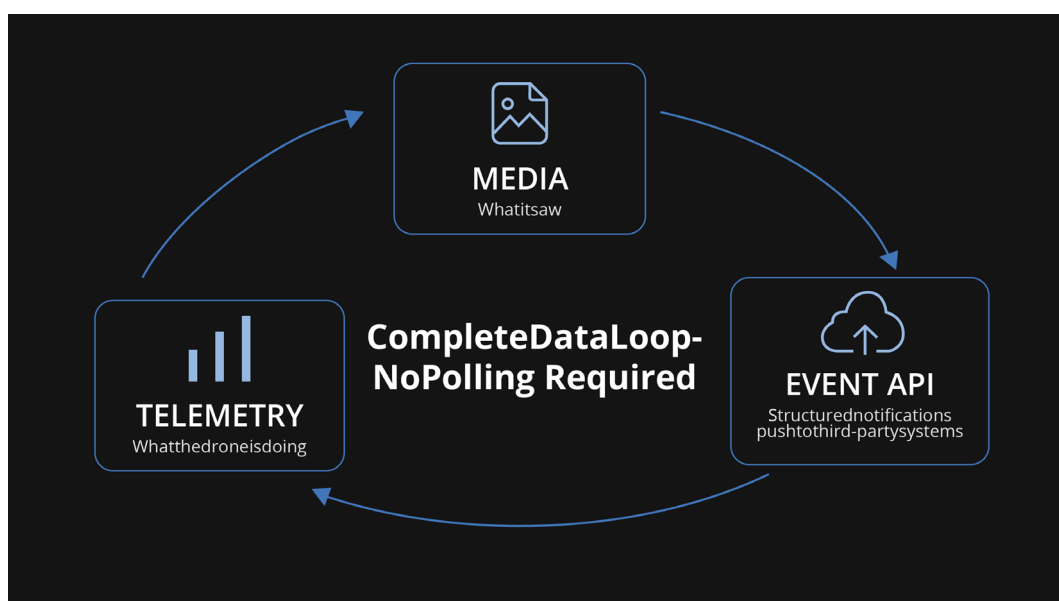


Figure 2.4.4-A Event API — Complete Data Loop (Telemetry / Media / Event)

Push Mechanism

- FlightHub 2 pushes event notifications via HTTP POST to the configured Webhook URL
- Both HTTP and HTTPS endpoints are supported
- Pushes originate from a fixed set of IP addresses — allowlist configuration on the receiving server is recommended for security

Region	Fixed Push IPs
EU Server / Global Server	18.198.174.64, 3.64.50.113, 3.71.214.14, 63.177.169.61, 52.3.45.199, 52.55.215.248, 52.73.49.179, 34.237.190.206, 34.237.182.49, 34.228.131.169
On-Premises	Depends on server configuration

Signature Verification

Every push request carries an **x-dji-signature** field in the HTTP header. The signature is calculated using HMAC-SHA256 based on the organization token. Receiving systems should validate this signature on every inbound request to confirm the notification originated from FlightHub 2.

Subscription Configuration

- Select event types to subscribe to from the available event list
- Define data scope: all projects in the organization, or specific projects
- Each project can only be referenced by one custom rule; projects already referenced by a rule are unavailable for additional rules
- Rule status (enabled / disabled) can be toggled without deleting the configuration

Configure Notification Rules

Rule Name*

Example: Flight Task Sync System

Receiving Address*

https://xxxx or http://xxxx

Subscription Data Range*

Risk Warning of Duplicate Subscriptions:

Rule for subscribing to all projects (All alerts) already exists in current organization
 If rule is created, selected projects receive duplicate event pushes from both global rule and this rule

☐ All Projects Within Organization
☒ Specify Project

Q

Figure 2.4.4-B Event API — Configure Notification Rules

Subscription Event Type*

- ☐ File Sync Notification
- ☐ Obtain AI Warning List & Snapshot
- ☐ Notification of model reconstruction completion
- ☐ New Wayline File Notification
- ☐ Exit RTH Notification
- ☐ FlightHub Mapping API - Model Reconstruction Progress
- ☐ Organization Annotation Add/Modify Notification
- ☐ RTSP Streaming Link Update Notification
- ☐ Custom Flight Zone Data Sync Notification

Figure 2.4.4-C Event API — Subscription Event Type List

Delivery Log Auditing

FlightHub 2 retains 7 days of event delivery statistics, including success rate, total events generated, and failed delivery count. Logs are downloadable in CSV format from the FH2 console — providing an audit trail for integration health monitoring and support escalation.

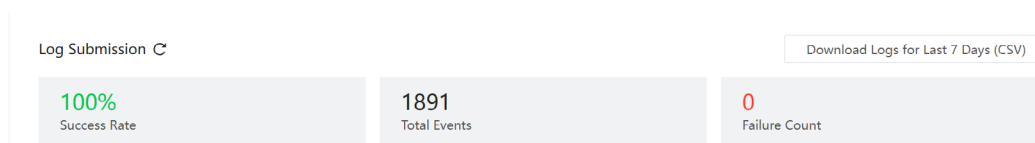


Figure 2.4.4-D Event API — Delivery Log: Success Rate & Event Count

Event Types — Official Reference

FlightHub 2 Event API supports nine event types across four operational domains:

Event Type	Domain	Trigger
file_sync_success	File Sync	File successfully synced to third-party storage
media_file_uploaded	Media	Media file uploaded to FlightHub 2 from device
wayline_file_uploaded	Flight Route	Flight route file uploaded or updated
model_post_reconstruction	Cloud Modeling	3D or 2D model reconstruction completed
model_reconstruction_percent_change	Cloud Modeling	Reconstruction or compression progress updated
ai_alert_record	AI Detection	AI detection alert triggered during mission
device_exit_return_home	Device Safety	Drone abnormally exited return-to-home sequence
map_geo_elements_update	Map Annotations	Map annotation added or edited
RTSP Address Update (no `type` field — payload-identified by `converter_id` and `sn`)	Live Streaming	Pushes refreshed `url`, `username`, `password`, `sn`, and `camera_index` when stream session is refreshed or device reconnects; payload identified by `converter_id` field

For complete payload schemas and field definitions, see the FlightHub 2 Event API Documentation.

Integration Scenarios

Understanding which events to subscribe to is as important as understanding how the push mechanism works. The scenarios below map specific event types to the operational requirements they serve in a DFR deployment.

Scenario 1 — Closing the DFR Response Loop

Live streaming provides eyes on the scene; telemetry tracks the drone through the response. Event API closes what neither can — notifying the command platform the moment a mission produces a result or encounters a safety event.

Subscribe to: ``media_file_uploaded``, ``device_exit_return_home``

- ``media_file_uploaded`` — footage is available for immediate review; surfaces in the incident record without manual checking
- ``device_exit_return_home`` — immediate notification on abnormal RTH exit with reason code; operations team can respond before the situation escalates

Scenario 2 — Automating the Evidence and Media Pipeline

File Sync moves media from FH2 to target storage. Event API tells the evidence management system when to act on it — no polling required.

Subscribe to: ``media_file_uploaded``, ``file_sync_success``

- ``media_file_uploaded`` — triggers evidence system to initiate retrieval
- ``file_sync_success`` — confirms file has reached third-party storage; ``file_external_key`` provides the exact storage path

Scenario 3 — Keeping Live Streams Connected

RTSP addresses change on device reconnection or stream session refresh. Platforms holding a stale address lose the feed silently — with no way to know until playback fails.

Subscribe to: RTSP Address Update

- Delivers refreshed ``url``, ``username``, ``password``, ``sn``, and ``camera_index`` the moment the address changes — stream re-establishes before any interruption reaches the operator



Figure 2.4.4-E Event API — DFR Response Closed-Loop Scenario

2.5 Deployment Architecture Options

FlightHub 2 is available in three deployment forms. Each serves a distinct phase of the integration journey — from early-stage validation through to production-scale delivery. Selecting the right form at each phase determines the speed of iteration, the scope of integration interfaces available, and the infrastructure investment required.

2.5.1 FlightHub 2 Public Cloud

FlightHub 2 Public Cloud is a hosted SaaS service managed by DJI. The platform is accessible immediately upon registration — no server configuration, no deployment lead time, and no infrastructure overhead. DJI handles all provisioning, maintenance, and updates on the backend.

For teams evaluating DFR solution feasibility or testing integration interfaces before committing to a private deployment, Public Cloud removes the infrastructure barrier entirely and puts a fully functional FH2 environment in reach within minutes.

Target Users: System integrators and enterprise teams in the requirements validation and solution design phase.

When to Use

- Validating FH2 core functionality and DFR scenario feasibility
- Building a rapid proof-of-concept before committing to a private deployment
- Exploring and refining use cases before on-premises infrastructure decisions are made
- Validating new feature releases — Public Cloud receives the latest FH2 updates first, making it the fastest environment to test and confirm new capabilities before they reach on-premises deployments

Integration Interface Availability

Interface	Available
FlightHub Sync	✓
Open API	✓
Event API	✓
Frontend Components	X On-premises only

Public Cloud is the right starting point for validation — not for production delivery or data-sovereign deployments.

2.5.2 FlightHub 2 All-in-One (AIO)

FlightHub 2 AIO is a pre-configured hardware unit with the complete FH2 software stack built in — delivering a private deployment environment out of the box, without server provisioning or infrastructure preparation.

For teams bridging the gap between Public Cloud validation and full On-Premises commitment, the Appliance mirrors the On-Premises feature set in a self-contained form factor — making it the natural next step when private deployment capability and full-interface integration need to be confirmed before production rollout.

Target Users: System integrators and enterprise IT teams validating private deployment requirements and full-interface integration scope prior to production deployment.

When to Use

- Validating the complete integration interface set — including Frontend Components, which are not available on Public Cloud
- Testing data sovereignty requirements and network isolation configurations in a controlled environment
- Confirming integration architecture and system compatibility before the production On-Premises deployment is scoped and resourced

Integration Interface Availability: All interfaces supported — FlightHub Sync, Open API, Event API, Frontend Components.

The Appliance bridges the gap between Public Cloud validation and production On-Premises delivery — use it to confirm the integration works before scaling it.

Hardware Specifications

The AIO ships as a pre-configured unit with the following hardware specifications:

Component	Specification
Processor	Intel® Core™ Ultra 7 Processor 265
Memory	64 GB
Storage	3 × 2 TB SSD — 1 for system, 2 for data (RAID 1 mirroring)
Graphics	NVIDIA RTX™ 2000 Ada

Component	Specification
Network (Onboard)	1 × 1 GbE RJ45 (Intel® I219-LM); 10/100/1000 Mbps adaptive
Network (Expansion)	4 × 10/100/1000 Mbps adaptive Ethernet (RJ45); Wi-Fi 7 (IEEE 802.11be) + 802.11a/b/g/n/ac/ax
Front I/O	1 × USB-A 3.2 Gen 2; 2 × USB-C 3.2 Gen 2 (DisplayPort capable); 1 × 3.5mm audio
Rear I/O	4 × USB-A 3.2 Gen 2; 3 × DisplayPort 1.4; 1 × GbE RJ45
Rated Power	330 W
Operating Voltage	100–240 V AC (90–264 V rated range)

2.5.3 FlightHub 2 On-Premises (Private Deployment)

FlightHub 2 On-Premises is the production-grade deployment form for organizations with data sovereignty requirements, large-scale fleet operations, or deep enterprise system integration needs. The platform runs on customer-managed infrastructure as a containerized application stack (Docker + K3S) on Linux Ubuntu.

For system integrators delivering scaled DFR programs, On-Premises is the endpoint of the deployment journey — full interface availability, full data control, and the infrastructure flexibility to grow with the operation.

Target Users: Enterprise customers and system integrators delivering production DFR deployments with compliance, data residency, or multi-system integration requirements.

When to Use

- Scaled DFR deployment with multiple Docks operating concurrently
- Data residency or air-gapped network requirements where data cannot leave the organization's infrastructure

- Deep integration with existing enterprise systems — CAD, GIS, VMS, PSIM — requiring full interface access and on-premises network reachability

Integration Interface Availability: All interfaces supported — FlightHub Sync, Open API, Event API, Frontend Components.

Hardware Sizing Reference

Infrastructure sizing scales with fleet size. The table below provides reference specifications for three fleet scales:

Fleet Scale	vCPU	Memory	Memory	Data Disk	Outbound Bandwidth
~20 Docks	8	32 GB	32 GB	1 TB SSD	≥ 100 Mbps
~50 Docks	16	64 GB	64 GB	3–4 TB SSD	≥ 250 Mbps
~100 Docks	32	128 GB	128 GB	5 TB SSD + cold storage	≥ 500 Mbps

Separate the OS disk from the data disk. Use SSD for primary data storage. For large-scale media archiving, consider a dedicated cold storage tier.

Network Port Overview

The following ports must be exposed for device connectivity and platform access. Source IP ranges should be restricted to known device and user address ranges wherever possible.

Category	Ports	Protocol	Purpose
Web Access	80 / 443	TCP	FH2 console and API entry point
Device Onboarding	1883 / 8883 / 20000 / 20001	TCP	MQTT device registration and command relay
Live Streaming	1935 / 8000	TCP	RTMP push / WebRTC pull
Media Upload	30802	TCP	MinIO storage ingest
4G Transmission	8090–8096	TCP + UDP	Enhanced transmission and heartbeat

Transmission protection is link-specific: OcuSync and 4G enhanced transmission use AES-256-based protections, while cloud, web, MQTT, and application links rely on HTTPS / TLS / MQTTS / WSS as applicable. For complete firewall configuration rules, source IP restriction recommendations, and internal management port reference, see **Appendix B: FlightHub 2 On-Premises Network Port Configuration Reference**.

Deployment Manual Reference

The complete FlightHub 2 On-Premises Deployment Manual (V1.5) — covering system requirements, installation steps, container configuration, license activation, and data directory setup — is available at:

FlightHub 2 On-Premises Version Deployment Manual (V1.5)

For DJI security architecture, encryption details, and cybersecurity compliance reference:
DJI Drone Security White Paper V3.1 (2025)

2.5.4 Deployment Form Selection Guide

The three deployment forms are not alternatives — they are stages. Most integration projects will move through Public Cloud and AIO before committing to On-Premises production delivery. The table below maps each form to the decision criteria that determine when it applies.

Dimension	Public Cloud	All-in-One Appliance	On-Premises
Recommended Phase	Solution validation / PoC	Integration validation / Pre-deployment	Production deployment / Scale
Data Sovereignty	Data hosted on DJI servers	Local storage, organization-controlled	Local storage, organization-controlled
Deployment Lead Time	Immediate — register and start	Out of the box	Requires server infrastructure preparation
Frontend Components	X	✓	✓
Full Interface Access	Partial	✓	✓

Dimension	Public Cloud	All-in-One Appliance	On-Premises
Scalability	Limited (suitable for evaluation and PoC; not recommended for fleet-scale operations)	Constrained by hardware spec	Scales with infrastructure
Recommended For	Feature validation, new release testing, use case refinement	Integration architecture confirmation	Scaled DFR delivery, enterprise integration

Start with the deployment form that matches your current project phase. Moving to the next form should be driven by confirmed requirements — not anticipated ones.

2.6 Product Reference Links

The following references consolidate all documentation and development resources cited in this chapter. Links are organized by category for quick access.

Platform & Product

- FlightHub 2 Product Page — Official product landing page — deployment options, feature overview, and licensing information
- FlightHub 2 On-Premises Deployment Manual (V1.5) — Complete installation and configuration guide for On-Premises deployments — system requirements, container setup, license activation
- FlightHub 2 On-Premises Operation and Maintenance Manual — Operational reference for On-Premises deployments — monitoring, troubleshooting, backup, and routine maintenance procedures
- DJI Dock 3 Deployment Guide — Site survey, installation, network configuration, and commissioning guide for DJI Dock 3

Development Interfaces

- [Open API Documentation](#) — Full API reference for FlightHub 2 Open API — endpoint definitions, authentication, and request/response schemas
- [FlightHub Sync & Event API Documentation](#) — Configuration guide and event type reference for FlightHub Sync and Event API integration
- [Frontend Components Documentation](#) — Integration guide for embedding FH2 frontend modules — Virtual Cockpit, Route Editor, Map, and Flight Records
- [DJI Cloud API — MQTT Topic Definition](#) — Complete MQTT topic reference — topic paths, payload structures, data directions, and field definitions

Security

- [DJI Drone Security White Paper V3.1 \(2025\)](#) — DJI's encryption architecture, cybersecurity design principles, and compliance reference for enterprise and government deployments

03 | Application Scenarios & Workflow Analysis

3.0 Chapter Overview

Chapter 2 established the technical architecture of a FlightHub 2 integration — the interfaces, the protocols, and the deployment options. This chapter answers the question that follows: what does that architecture look like when it is actually put to work?

This chapter is written for solution architects, system integrators, and operations leads who need to understand not just how the technology connects, but how an organization operates around it — the workflows, the roles, and the integration sequence that turn a DFR platform into a functioning response capability.

By the end of this chapter, you will be able to:

- **Understand why DFR changes the response model** — not as a technology add-on, but as a structural change to how incidents are detected, assessed, and closed; and where the traditional workflow fails under operational pressure
- **Map the end-to-end DFR workflow** — from Dock pre-deployment and alarm trigger through live intelligence delivery, command assessment, and post-incident evidence archiving
- **Define the staffing configuration a mature DFR program requires** — six operational roles, what each one does, which systems it operates, and how they coordinate across the response cycle
- **Translate the DFR workflow into an integration roadmap** — connecting the four system integration requirements to the right interfaces, with a phased sequencing plan that builds from Public Cloud validation through to On-Premises production delivery

This chapter bridges the technical content of Chapter 2 and the operational reality of deploying DFR at scale. It is most useful when read alongside the interface and deployment guidance in Chapter 2.

3.1 Workflow Analysis

This section traces the operational change DFR introduces — from the structural limitations of traditional dispatch, through the end-to-end DFR workflow, to the team configuration and integration architecture that make it function reliably at scale.

3.1.1 Traditional Workflow

Before DFR, public safety response operated on a linear dispatch model — each step dependent on the one before it, with the command center relying on voice communication and ground unit reports for situational awareness throughout.

The six-step traditional dispatch sequence:

- 1. Alarm received** — a 911 call or sensor trigger reaches the CAD (Computer-Aided Dispatch) system; the operator logs the incident type, caller-reported location, and initial severity assessment
- 2. CAD allocation** — the dispatcher identifies the nearest available unit and assigns the call; unit availability, current workload, and proximity determine the assignment
- 3. Ground unit en route** — the assigned patrol unit responds; travel time is determined by distance, traffic conditions, and road access — factors outside the dispatcher's control
- 4. On-scene arrival** — the first officer arrives and conducts an initial visual assessment; this is the first moment the command layer receives ground-truth information about the scene
- 5. Situation report** — the officer communicates scene conditions via radio; the dispatcher and command center build a picture of the incident from verbal descriptions, with no visual confirmation
- 6. Reinforcement decision** — based on the radio report, the dispatcher decides whether to escalate — requesting additional units, specialist resources, or emergency services; the decision is made without visual confirmation of the scene

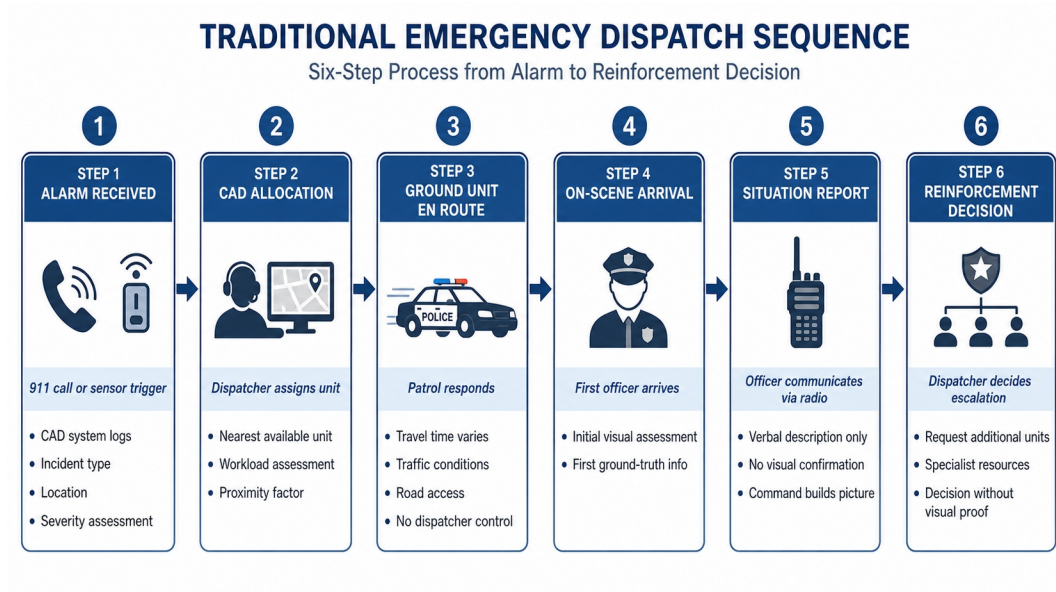


Figure 3.1.1-A Traditional Emergency Dispatch Workflow — Six-Step Linear Response Model

Structural limitations of this model:

- **Information lag** — the command center has no visual access to the scene until a ground unit arrives and reports; the interval between alarm and first situational awareness is determined entirely by ground travel time
- **Decision quality** — reinforcement decisions are made on the basis of verbal reports from an officer who has just arrived; assessment accuracy is constrained by individual judgment under pressure
- **Resource efficiency** — units are dispatched based on availability and proximity, not scene-specific requirements; over- or under-resourcing is a common outcome
- **Coordination overhead** — multi-unit operations rely on radio coordination; shared situational awareness degrades as the number of responding units increases

In a dense urban environment, the gap between alarm and actionable situational awareness can span several minutes — a window in which incident conditions continue to evolve without command visibility.



Figure 3.1.1-B The Command Visibility Gap — Traditional Dispatch vs DFR Response Time

3.1.2 DFR-Enhanced Workflow

With DFR integrated into the dispatch system, the response chain is no longer linear. The drone reaches the scene before the ground unit, the command center gains visual situational awareness before the first officer arrives, and reinforcement decisions are made on the basis of confirmed aerial intelligence rather than radio descriptions.

The five-stage DFR workflow below reflects the full operational sequence — from pre-deployment preparation through incident closure.

Stage 1 — Dock Deployment & Mission Safety



Figure 3.1.2-B NFZ Configuration — Custom Flight Boundaries & Restricted Zones in FlightHub 2

7. Custom flight area configuration — high-risk structures are designated as NFZ (No-Fly Zones) within FlightHub 2; the drone will not enter these boundaries regardless of mission routing



Figure 3.1.2-C 3D Cloud Modeling — Jurisdiction Aerial Reconstruction in FlightHub 2

8. Supplementary detail capture — key structures and complex urban geometries receive additional photography passes to ensure model accuracy at decision-critical locations

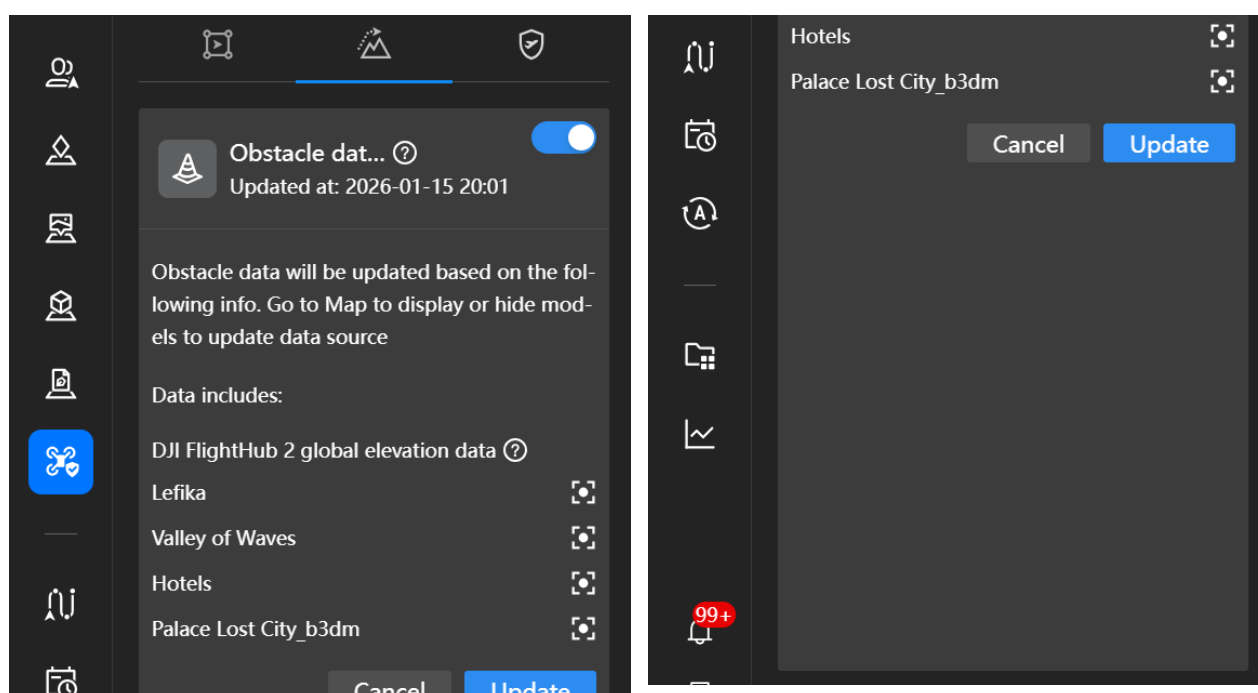


Figure 3.1.2-D Obstacle Data Configuration — Building Models as Automatic Avoidance Zones

9. Obstacle data configuration — building models are set as obstacle flight zones; FH2 applies automatic avoidance buffers (typically 30m) during route calculation

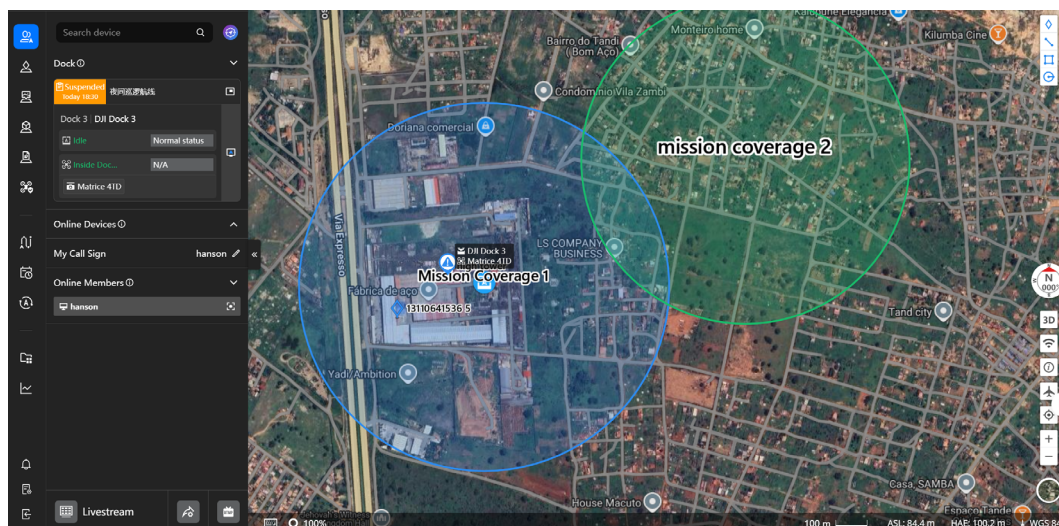


Figure 3.1.2-E Operational Overlay Deployment — Dock Coverage Areas & Patrol Routes in FlightHub 2

10. Operational overlay deployment — mission zones, patrol routes, and Dock coverage areas are mapped as overlays within FH2, establishing the pre-configured response grid

Team & System Interaction:

- **Platform configuration** — NFZ boundaries, 3D model uploads, obstacle data, and coverage overlays are configured in FH2; this setup must be re-validated whenever the operational environment changes (new construction, event-specific restrictions, airspace changes)
- **Dock pilot assignment** — each Dock is assigned a designated pilot with a defined coverage zone, on-call schedule, incident type responsibility, and primary aircraft (model, payload, battery readiness)

A well-configured response grid with clearly assigned pilots and equipment is the foundation every automated response depends on.

Stage 2 — Alarm Trigger, Task Generation & Scene Arrival

A 911 call-taker receives the report — logging the incident location, nature of the event, and initial caller assessment. The completed incident record is dispatched to the CAD system.

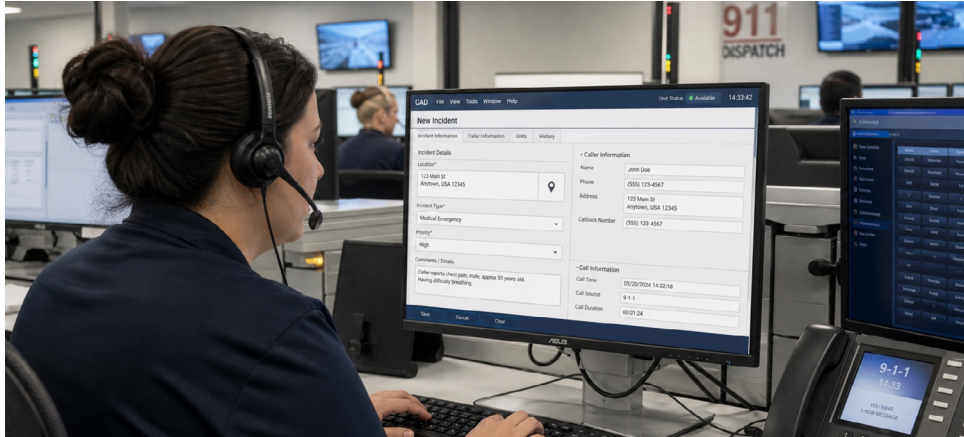


Figure 3.1.2-F Alarm Trigger to Scene Arrival — From 911 Call to Aerial Presence

CAD pushes the alarm coordinates to FlightHub 2 through the Auto Dispatch Workflow inbound trigger endpoint, triggering the configured FH2 Workflow. FH2 applies level-threshold based auto-dispatch: event levels are 1–5, the default threshold is 3, and auto-dispatch fires only when event level $\geq$ threshold. In Auto-recommend mode, FH2 selects the most suitable idle device and uses the device list as fallback if no suitable device is available. In Manual assignment mode, the project can use a fixed device assignment or require a per-trigger manual pick.



Figure 3.1.2-G Real-Time Flight Monitoring in FlightHub 2 Virtual Cockpit

Team & System Interaction:

- **911 Call Center** captures and qualifies the incident; the completed record is the trigger for all downstream DFR response
- **Dispatch Center (CAD)** pushes the incident coordinates to FH2 via the **Auto Dispatch Workflow** inbound trigger endpoint; dispatch proceeds through **Auto-recommend** or **Manual assignment**, and auto-dispatch fires only when event level $\geq$ the configured threshold
- **Drone Emergency Response Team** maintains fleet oversight across all active Docks — confirming availability, monitoring concurrent missions, and managing battery readiness for follow-on deployments

From 911 call to aerial presence: the DFR sequence can run through Manual assignment or Auto-recommend under the configured level-threshold based auto-dispatch policy, depending on project policy.

Stage 3 — Live Video & Location Feed

With the drone on scene, the command center gains real-time visual access before any ground unit has arrived. Two parallel data streams activate simultaneously:

- **Video feed** — live stream pushed via RTMP/RTSP to the VMS display wall and CAD operator interface; the command center sees what the drone sees in real time
- **Position feed** — drone coordinates update continuously via MQTT to the CAD/GIS map; the command center tracks the drone's position and coverage area on the operational map

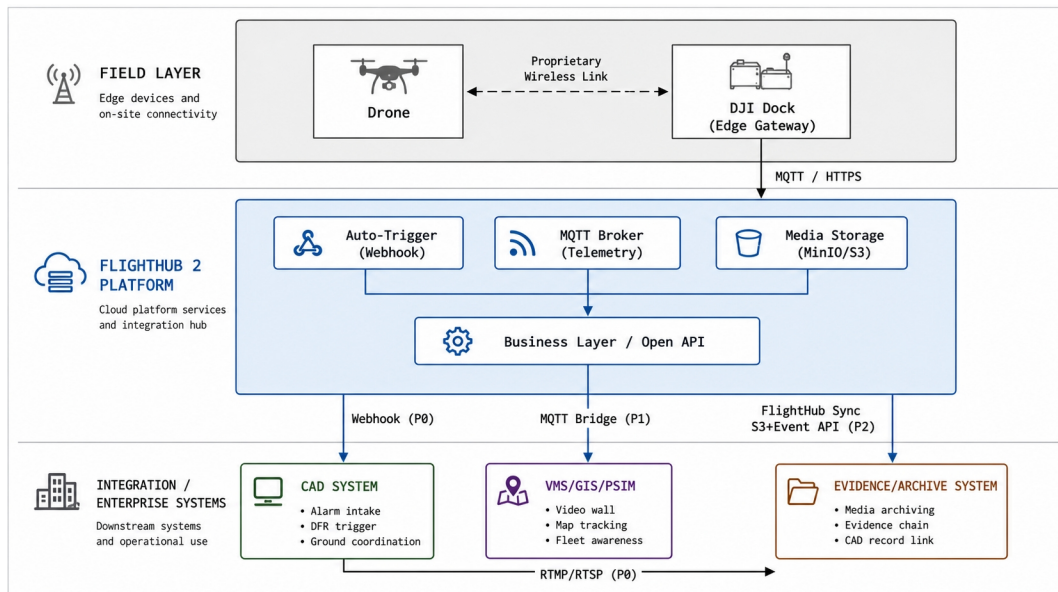
Where multiple Docks are deployed, concurrent feeds from all active drones appear on the centralized video wall simultaneously — giving the command center a unified aerial picture across the full incident area.



Figure 3.1.2-H Live Video Feed — Multi-Stream Display on Command Center Video Wall

The same video feed, drone position, and map annotations — incident markers and coverage zones — are accessible to ground units through their CAD terminals en route. Every responding team operates from the same aerial picture before anyone reaches the scene.





*Figure 3.1.2-I Shared Operational Picture —
Drone Position and Map Annotations Across All Connected Terminals*

Team & System Interaction:

- **Command Center** is the primary consumer at this stage — operators monitor drone feeds on the VMS video wall and track positions on the CAD map simultaneously
- **Dispatch Center** monitors the drone's real-time position on CAD to coordinate ground unit approach routes
- **Patrol & Response Ground Team** accesses the drone's live video feed, position data, and map annotations through their CAD terminals en route
- **Drone Emergency Response Team** manages camera orientation and zoom through FH2, and pushes map annotations to mark key points of interest identified during aerial observation

Real-time aerial video, live position data, and shared map annotations give every responding team a unified operational picture from the moment the drone arrives on scene.

Stage 4 — Severity Assessment & Resource Allocation

With live aerial video on scene, the incident commander classifies the incident — scale, nature, immediate risks, and resource requirements — before the first ground unit arrives. That assessment drives immediate dispatch decisions: hold the current allocation, redirect units already en route, or escalate to specialist response.



*Figure 3.1.2-J Severity Assessment —
Command Center Decision-Making Based on Live Aerial Intelligence*

Team & System Interaction:

- **Command Center** leads the assessment — the incident commander classifies severity and communicates the resource decision to the Dispatch Center; decisions are made on confirmed visual intelligence rather than radio description
- **Dispatch Center** acts on the assessment in real time — reallocating units, requesting specialist resources, or escalating to partner agencies through CAD
- **Drone Emergency Response Team** adjusts the drone's flight pattern on request — expanding coverage, descending for closer observation, or repositioning to cover a secondary area of interest

The command center's ability to assess before ground units arrive turns the response interval from dead time into decision time.

Stage 5 — Incident Closure & Data Archiving

When the incident is resolved, the DFR system supports an automated evidence handoff — reducing manual file retrieval, separate login steps, and dispatcher intervention.

11. Mission marked complete in FH2 → Event API pushes `media\_file\_uploaded` notification

12. Media files sync to the evidence management system via FlightHub Sync S3

13. Event API pushes `file\_sync\_success` → evidence system confirms receipt and initiates archiving

14. Flight logs and AI detection records are written to the archive database

15. The complete evidence package is linked to the incident record in CAD

Team & System Interaction:

- **Evidence / Archive Team** receives the `file\_sync\_success` event notification — the archiving workflow can proceed automatically while the team verifies receipt and evidence controls
- **Drone Emergency Response Team** confirms mission completion in FH2 and verifies that media files have been successfully uploaded; any gaps in the media record are flagged before the Dock returns to standby

The evidence chain can be closed shortly after mission completion, with traceability supported by sync status, event notifications, archive records, and customer evidence controls.

3.1.3 Role Breakdown & System Interaction

A DFR program is not a technology deployment — it is an operational system. The hardware and software perform the automated functions, but the people behind them determine whether the system operates reliably at scale. A mature DFR deployment requires a clearly defined team structure, with each role carrying specific responsibilities and operating within a defined system boundary.

The six roles below represent the recommended staffing configuration for a production DFR program.

DFR Program Manager

The operational core of the DFR program. This team owns the full lifecycle of drone readiness — from pre-deployment configuration through post-mission verification.

- Responsible for Dock deployment site selection, NFZ configuration, 3D Cloud Modeling, and coverage grid design in FlightHub 2
- Assigns Dock pilots to coverage zones and maintains the pilot-Dock-aircraft assignment matrix
- Monitors active missions across the fleet via the FH2 Organization Dashboard
- Verifies media upload completion and flags gaps in the mission record after each response

This team is the bridge between the DFR platform and the operational command — they ensure the system is ready before every alarm fires and accountable after every mission closes.

Dock Pilot

The human layer in the automated response loop. Each Dock pilot is assigned a specific Dock and coverage zone, and is responsible for monitoring that Dock's automated missions in real time.

- Monitors flight progress, battery status, and system alerts via FH2 Virtual Cockpit during active missions
- Controls camera orientation and zoom on request from the command center during aerial observation
- Stands by to take manual control if the automated mission encounters a condition that requires human intervention
- Manages Dock return-to-home, battery swap, and readiness confirmation for follow-on deployments

The Dock pilot does not fly the drone in a DFR response — they supervise it, and intervene only when the system cannot resolve a condition on its own.

Command Center

The decision-making hub of the DFR response. The command center receives the aerial intelligence generated by the drone and translates it into operational decisions.

- Monitors live drone feeds across all active Docks on the VMS video wall
- Tracks drone and ground unit positions on the unified CAD operational map
- Conducts severity assessment based on aerial imagery before ground units arrive
- Issues resource allocation decisions — additional units, specialist response, escalation — to the Dispatch Center
- Coordinates multi-agency response using the shared aerial situational picture

The command center's effectiveness in a DFR deployment is directly proportional to the quality of the aerial feed and the speed at which it arrives. The drone creates the window; the command center uses it.

Dispatch Center (CAD)

The integration point between the civilian alarm system and the DFR response chain. The Dispatch Center is where the 911 call becomes a drone mission.

- Receives incident records from the 911 call center and classifies them for DFR eligibility
- The Auto Dispatch Workflow starts the FH2 Workflow when a qualifying incident is received through the inbound trigger endpoint; device dispatch follows Auto-recommend or Manual assignment, with auto-dispatch firing only when event level $\geq$ the configured threshold
- Coordinates ground unit routing using drone position data visible on the CAD map
Manages multi-agency notification and escalation based on command center assessment

The Dispatch Center does not operate the drone — but the quality of the incident record it passes to FH2 determines whether the right Dock is dispatched to the right location.

Evidence / Archive Team

The team responsible for closing the post-incident data loop. In a DFR deployment, evidence archiving is automated — but the Archive Team ensures the automation is working and the evidence record is complete.

- Reviews and validates incoming mission evidence packages — confirming footage, flight logs, and detection records are complete and correctly attributed to the incident
- Maintains the chain of custody for all mission data — supporting evidence traceability when paired with customer evidence controls and authorized access processes
- Manages retention schedules and access controls in accordance with organizational and regulatory requirements
- Identifies and resolves gaps in the evidence record — coordinating with the DFR Program Manager when mission data is incomplete or missing

In a DFR deployment, the Archive Team's primary role shifts from manual file handling to evidence governance — the system moves the files; the team ensures they hold up.

IT / Platform Team

The team that keeps the integration running. The IT / Platform Team owns the technical infrastructure that connects all other roles and systems.

- Configures and maintains the Auto Dispatch Workflow workflow POST / inbound trigger endpoint between CAD and FH2
- Manages Public Cloud Telemetry Data Forwarding or On-Premises MQTT Bridge configuration for telemetry forwarding to CAD/GIS platforms
- Maintains S3 sync endpoints and Event API subscriptions for the evidence archiving pipeline
- Enforces access control, network security, and data sovereignty requirements across all integrated systems

- Re-validates integration configurations when the operational environment changes

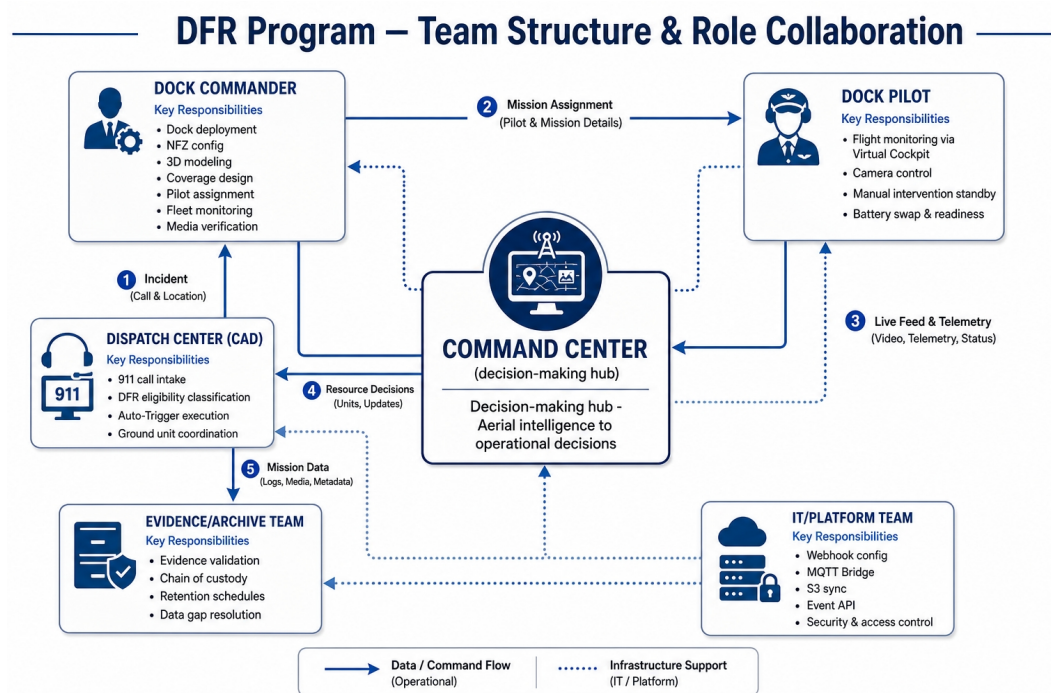


Figure 3.1.3-A DFR Role Interaction — Six-Role Team Coordination Structure

How the Roles Work Together

No single role operates in isolation. The DFR workflow functions because each team's output feeds the next team's input — connected by the integration interfaces configured before the first alarm fires.

Workflow Stage	Lead Role	Supported By
Pre-deployment setup	DFR Program Manager	IT / Platform Team
Alarm intake & Auto Dispatch Workflow	Dispatch Center (CAD)	—
Mission monitoring	Dock Pilot	DFR Program Manager
Aerial intelligence delivery	Dock Pilot	Command Center
Severity assessment & dispatch	Command Center	Dispatch Center
Evidence archiving	Evidence / Archive Team	DFR Program Manager

A DFR program is only as reliable as its weakest role configuration. Gaps in pilot coverage, misconfigured integrations, or undefined archiving workflows will surface under operational pressure — not during testing.

3.1.4 System Integration — From Requirements to Architecture

The DFR workflow connects multiple enterprise systems through four integration interfaces. Understanding how data flows between these systems — and in what sequence — is the foundation for designing a reliable integration architecture.

System Data Flow Architecture

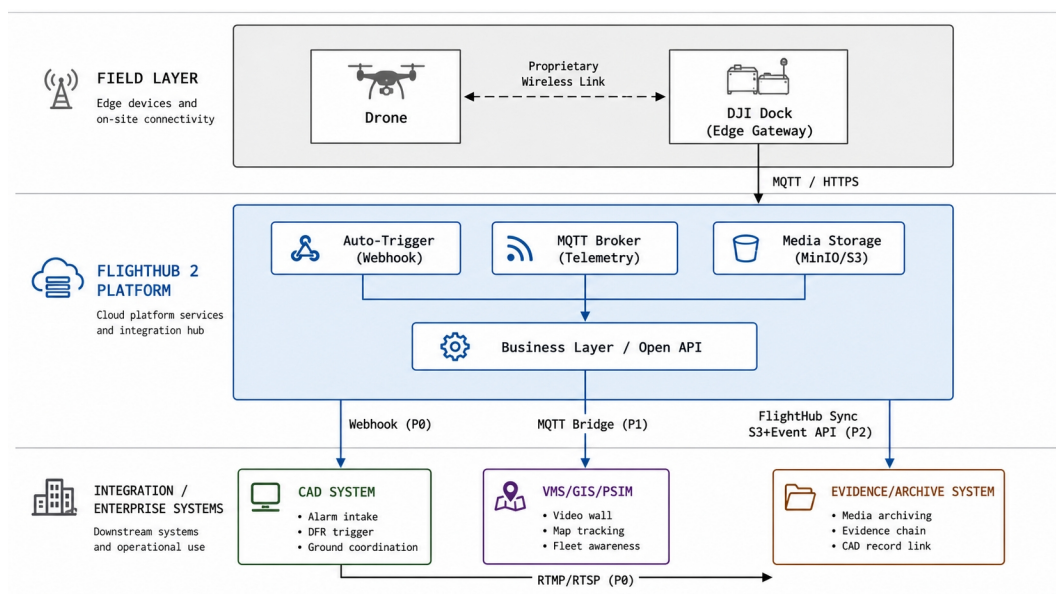


Figure 3.1.4-A DFR System Integration Architecture — Full Data Flow Topology

Integration Interface Summary

Priority	Integration Domain	Interface	Protocol	Data Flow	Target System
P0	DFR Emergency Trigger	Auto Dispatch Workflow (workflow POST / inbound trigger endpoint)	HTTPS POST	HTTPS POST	FlightHub 2
P0	Live Video Streaming	Sync Livestream / Open API	RTMP / RTSP / WebRTC	FH2 → VMS / CAD	VMS, CAD
P1	Drone Telemetry	Telemetry Data Forwarding / MQTT Bridge	MQTT	FH2 → CAD / GIS / PSIM	CAD, GIS, PSIM
P2	Media & Evidence Archiving	FlightHub Sync S3 + Event API	S3 / HTTPS	FH2 → Archive System	Evidence System

Recommended Integration Sequencing

Integration should proceed in priority order across three deployment phases. Each phase validates the foundation before the next layer of complexity is added.

Phase	Deployment Form	Integration Steps	Exit Criterion
Phase 1	Public Cloud	1. Configure Auto Dispatch Workflow; connect CAD test environment 2. Validate coordinate push → mission generation 3. Configure Sync RTMP/RTSP to test VMS; validate stream stability	Test alarm → FH2 mission generated → live stream on VMS
Phase 2	All-in-One Appliance	1. Reconnect Auto Dispatch Workflow to private FH2; re-validate trigger 2. Connect Sync RTMP/RTSP to production VMS 3. Configure Telemetry Data Forwarding / MQTT Bridge; validate telemetry on CAD map	P0 + P1 stack operating reliably on private infrastructure
Phase 3	On-Premises	1. Connect Open API for event-triggered livestream session initiation and pull-address retrieval 2. Configure Sync S3 to evidence storage 3. Subscribe Event API; validate archiving workflow 4. Full end-to-end validation: alarm → dispatch → feed → closure → archived evidence	All four integration domains active and verified in a live mission

04 | FlightHub 2 Data Security and Compliance for DFR

4.0 Chapter Overview

DFR connects drones, docks, command centers, and third-party business systems. Its value comes from fast response, but its entry requirement is data security.

In public safety, campus security, energy sites, and critical infrastructure scenarios, DFR data can include live video, device location, task status, alarm coordinates, media files, flight records, and user permission information. After this data enters DJI FlightHub 2 (FH2), IT, legal, cybersecurity, and business stakeholders usually review five questions:

01

Q: What data does DFR generate?
Live video, device telemetry, task data, media files, flight records, event notifications, identity and permission data.

02

Q: Which systems does the data pass through?
DJI Dock 3 / drone, FH2, integrator or end-user third-party systems, third-party storage, VMS / CAD / GIS / PSIM / Evidence systems.

03

Q: Where is the data processed and stored?
It depends on FH2 Public Cloud or FH2 On-Premises deployment, and whether the project uses FlightHub Sync, third-party object storage, or a local network environment.

04

Q: Who can access the data?
It depends on organization roles, project roles, SSO / OAuth, 2FA, OpenAPI tokens, Webhook / EventAPI configuration, and third-party system permissions.

05

Q: How can security capability be explained?
Through transmission encryption, identity and access control, data storage, third-party integration controls, operations processes, ISO 27001 management system evidence, and deployment responsibility boundaries.

4.1 Data Types and Sensitivity Analysis

The core of DFR is turning the drone into a schedulable, remotely accessible, continuously operated response node. This node collects scene information early in an incident and sends it to the command center, third-party systems, or evidence workflows.

As the platform layer, FH2 processes several types of data: livestreams, telemetry, tasks, media, events, identity, and operational records. Each data category has a different sensitivity profile, so controls should be designed separately.

Data Type	Typical Content	Sensitivity	Main Risk	Control Focus
Live video	Dock security camera, payload camera, livestream sharing, livestream forwarding	High	Unauthorized viewing, excessive sharing, secondary forwarding	Project permissions, livestream sharing management, forwarding channel control, session expiration, minimum visibility scope.
Telemetry and location	Device SN, coordinates, altitude, battery, attitude, signal status, task status, flight trajectory	High	Exposure of patrol routes, response zones, critical facility locations, and operational patterns	MQTT account and password, topic scope control, project isolation, transmission protection, third-party broker permission control.
Task and control data	Waylines, task plans, Auto Dispatch Workflow parameters, FlyTo target, alarm level, return strategy	High	Incorrect dispatch, unauthorized task creation, route mismatch with site constraints	Incorrect dispatch, unauthorized task creation, route mismatch with site constraints
Media and evidence files	Photos, videos, panoramas, SRT files, annotations, AI detection annotations, reports	High	File loss, incorrect evidence ownership, unclear retention rules	Project-level storage rules, object path conventions, sync status monitoring, deletion after 200 OK confirmation, archive validation.

Data Type	Typical Content	Sensitivity	Main Risk	Control Focus
Event notifications	File sync, media upload, AI alarm, wayline file update, model reconstruction completion, abnormal return	Medium-high	Forged callbacks, duplicate processing, cross-project push errors	HMAC-SHA256 signature validation, fixed IP allowlist, idempotent processing, project-scope configuration, failure log checks.
Operations and support records	Device alarms, task status, workflow task logs, EventAPI delivery logs, support authorization records	Medium	Unclear accountability, insufficient troubleshooting evidence	Project-level operations process, support authorization process, log retention boundary explanation, SIEM / ticketing workflow handoff.

DFR data security covers the full chain: where data comes from, where it goes, who controls it, who can access it, how it is retained and deleted, and what responsibilities sit with DJI, integrators, end users, and third-party systems.

4.2 DFR Data Flow

DFR data flow can be explained by data type. The diagram below identifies telemetry, task control, live video, media files, third-party events, and identity / access control by data path.

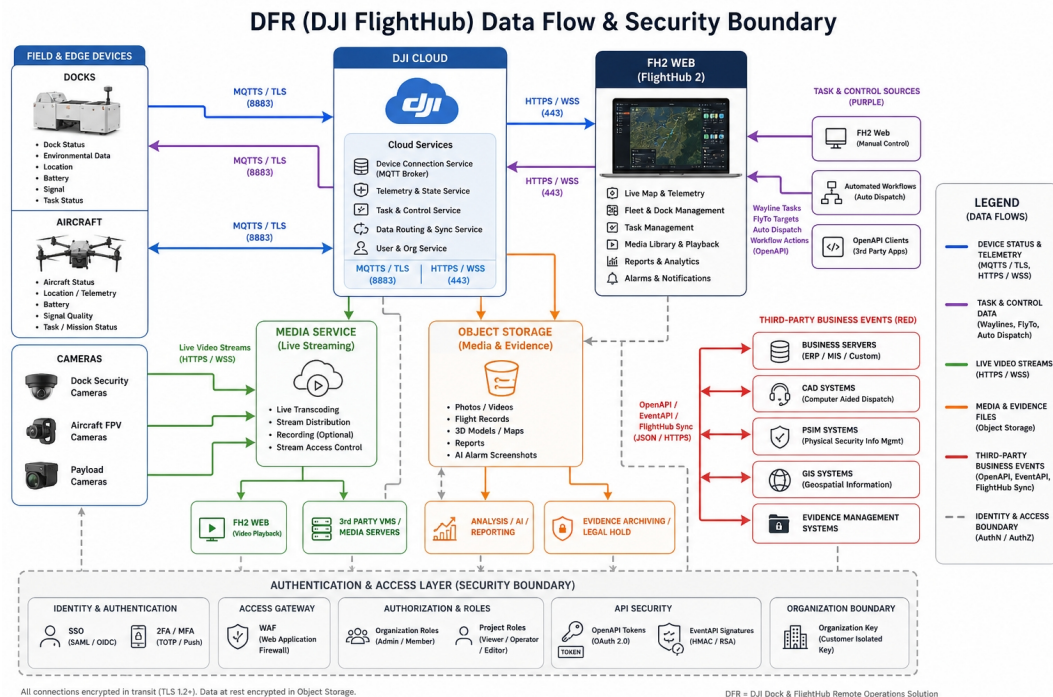


Figure 4.2-A DFR Data Flow and Security Boundary

- **Blue line:** device status and telemetry. Docks and aircraft report device status, location, signal, battery, task status, and related information to DJI Cloud through MQTTS / TLS, HTTPS / WSS, and similar links.
- **Purple line:** task and control data. FH2 Web, automated workflows, or OpenAPI sends wayline tasks, FlyTo targets, Auto Dispatch Workflow actions, and return actions to docks / aircraft.
- **Green line:** live video. Video from the dock security camera, aircraft FPV camera, or payload camera enters Media Service and is then provided to FH2 Web or third-party VMS / Media Servers.
- **Orange line:** media and evidence files. Photos, videos, flight records, models, reports, and AI alarm screenshots enter object storage and are used in the FH2 media library, analysis, reporting, or evidence archiving workflows.

- **Red line:** third-party business events. Business servers, CAD, PSIM, GIS, Evidence systems, and other business platforms exchange data with FH2 through OpenAPI, EventAPI, and FlightHub Sync.
- **Gray dashed line:** identity authentication and access boundary. Auth, WAF, organization roles, project roles, SSO, 2FA, OpenAPI tokens, Organization Key, and EventAPI signatures jointly affect access rights.

4.3 Deployment Types and Data Control

When IT or compliance teams ask where the data is, the core concern is control. The data control boundary of FH2 is mainly determined by deployment type.

Deployment Type	Data Control Positioning	Operations and Responsibility Boundary
FH2 Public Cloud	Data runs in DJI-managed cloud regions. FH2 data for European users is stored in the AWS Frankfurt data center; other overseas regions use AWS United States data centers; China user data is stored on China servers.	DJI is responsible for cloud service infrastructure, platform operations, security updates, and cloud-side security controls. The project party is responsible for account governance, project permissions, third-party system permissions, data retention, and internal compliance review.
FH2 AIO / private all-in-one	A pre-integrated private deployment type in a project-controlled environment, suitable for stronger local control while reducing the complexity of full private deployment.	The project party manages the local network, accounts, storage, and operations. DJI provides FH2 platform capabilities and delivery support.
FH2 On-Premises	Deployed in the end user's intranet, own servers, or designated cloud environment. Data storage and processing can stay in local or specified infrastructure.	The project party takes more responsibility for infrastructure, network, storage, backup, upgrades, and security policy management. DJI provides FH2 product capabilities and security controls within the interface boundary.

4.4 Security Control Framework

DFR security should be explained by control domain, covering transmission, identity, storage, network, development, operations, vulnerability handling, incident response, and third-party boundaries.

4.4.1 Link and Transmission Security

Link	Security Mechanism	Meaning for DFR
OcuSync	AES-256 encryption; random session keys generated at each power-on; secure key negotiation and link encryption protect against hijacking, man-in-the-middle attacks, replay, and eavesdropping.	Protects the control and image transmission link between aircraft and ground control devices.
4G enhanced transmission	AES-256 encryption; mutual authentication between air system and ground control; device certificates and private keys are protected by TrustZone; the enhanced transmission server only forwards encrypted data and does not collect or store it.	Applies to remote operations that need cellular link enhancement.
Cloud connection	Ground control establishes TLS links with cloud infrastructure; TLS 1.2 or higher; the air system calculates authentication tokens in a trusted zone and binds tokens to device SN.	Protects cloud connections among dock, aircraft, controller, and FH2.
FH2 Web and server	Protects browser access and management operations.	Protects browser access and management operations.
Dock and FH2 server	Dock and FH2 server use MQTT with TLS.	Protects dock device status, task data, and telemetry transmission.
FH2 and app / device	FH2 communicates with apps or devices through HTTPS, MQTTS, WSS, and anti-hotlinking mechanisms.	Protects real-time data, media, and device connections.

The security controls of DFR are distributed across the device link, 4G link, cloud connection link, web link, and third-party integration link. TLS is one layer, not the entire security model.

4.4.2 Identity and Access Control

FH2 uses unified account authentication and role-based access control (RBAC). FH2 permissions exist at both organization and project levels: organization roles control global resources such as members, devices, and projects; project roles control viewing, control, media, and task operations within a specific DFR project.

Common DFR roles can be mapped as follows:

DFR Role	FH2 Organization Role	FH2 Project Role	Configuration Method
IT / Security administrator	Super Administrator or Organization Administrator	Project Admin	Manage SSO, 2FA, members, projects, devices, tokens, webhooks, Sync rules, and security configuration. Keep this group minimal.
DFR operations lead	Organization Administrator or Member	Project Admin	Manage task plans, automated workflows, device status, media files, and project configuration.
Integrator developer	Organization Administrator or Member	Project Admin	Grant permissions required for OpenAPI, EventAPI, FlightHub Sync, or frontend component access.
Command center pilot / duty operator	Member	Project Device Operator	View project status, livestream, and device state; execute Live Flight Controls, FlyTo, RTH, pause, or cancel tasks when authorized.
Device maintainer	Device Maintainer	Project Device Admin	View and maintain dock / aircraft status, alarms, device information, and maintenance operations.
Evidence/ media administrator	Member	Project Output Admin	View, download, share, annotate, and archive media files and reports.
Read-only reviewer	Member	Project Member/ View-only	View project status, flight records, media catalog, and reports.

FH2 supports the following identity controls:

Capability	Lead Role
SSO	FH2 On-Premises supports enterprise SSO via SAML assertions with IdP metadata and X.509 public key certificate. FH2 Public Cloud relies on DJI's unified account system (OAuth 2.0).
MFA	Individual members can enable Multi-Factor Authentication by binding an authenticator app (TOTP). Organization-wide enforced MFA must be governed through internal user onboarding policy.
On-Premises local account	On-Premises can use a local account system with manual account creation and management by administrators.
On-Premises third-party account	On-Premises can connect to a third-party account system through OAuth2.
OpenAPI header	OpenAPI uses headers such as X-User-Token and X-Project-Uid for authentication and access control.
EventAPI signature	EventAPI uses HMAC-SHA256 based on the Organization Token to generate signatures. The receiver must validate x-dji-signature.

4.4.3 Storage and Data Protection

Storage Scenario	Control Point
FH2 Public Cloud	FH2 data for European users is stored in AWS Frankfurt; other overseas regions use AWS United States data centers. Sensitive personal data uses storage encryption, and databases enable global encryption at rest by default.
FlightHub Sync third-party storage	Supports third-party S3-compatible storage buckets. Independent synchronization rules and target buckets can be configured per project.
Media File Direct Transfer	When enabled, media files collected by a current plan can bypass FH2 and transfer directly to a third-party platform. The project party must define target storage permissions, archive paths, and upload confirmation mechanisms.
On-Premises	Data can be processed and stored inside the customer's own network and designated infrastructure. The project party is responsible for local servers, networks, backups, permissions, and security operations.

4.4.4 Network and Infrastructure Protection

Control Domain	Security Capability
Cloud provider infrastructure	DJI cloud services rely on cloud providers with security qualifications and high reliability, such as AWS and Alibaba Cloud.
Network protection	Network requests pass through multiple layers of security protection, such as anti-DDoS and WAF.
Host and application security	DJI performs log security checks, vulnerability scanning, intrusion detection, code analysis, and regular penetration testing.
Operations permissions	DJI operations teams follow Need-to-Know and least-privilege principles and restrict server-side operations through SOPs.
Availability and recovery	DJI has service monitoring and failure drill mechanisms. Cloud service databases can be restored from backups created within the last 30 days and support point-in-time restoration within the last 7 days.

4.4.5 Secure Development, Vulnerabilities, and Incident Response

DJI performs regular penetration testing and code analysis against online applications. If vulnerabilities are found, the development team remediates them in a timely manner based on risk severity. DJI has operated a Bug Bounty Program since 2017 to encourage responsible discovery and reporting of potential vulnerabilities.

FH2 security capability is built on continuous development, testing, vulnerability handling, and operations processes. The design, development, and operation services of DJI FlightHub 2 are included in DJI's information security management system.

4.5 What ISO 27001 Represents

DJI FlightHub 2 has obtained ISO 27001 certification issued by BSI. The certification covers DJI's information security management practices for FlightHub 2 design, development, and operation services.

Item	Value
Certificate	ISO/IEC 27001:2022
Certificate number	IS 717854
Certified organization	Shenzhen DJI(DJ-Innovations) Technology Co., Ltd. - Enterprise Software Department
Scope	The provision of DJI FlightHub 2 product design, development and operation service.
Statement of Applicability	Version 5.0 dated Dec. 24, 2024
Effective date	2026-01-22
Latest issue date	2025-12-15
Valid until	2029-01-21
Capability Domain	FH2-Related Capability
Risk management	FH2 design, development, and operation services are included in information security risk identification, assessment, treatment, and review processes.
Security controls	Transmission, storage, access control, operations permissions, vulnerability handling, and service availability are supported by the management system.
Continuous improvement	Information security controls are maintained through risk assessment, internal review, vulnerability remediation, and operations review.
Independent validation	BSI certification provides third-party evidence of information security management capability.

ISO 27001 certification does not automatically replace the end user's own compliance review. GDPR, NIS2, CJIS, and similar legal or industry requirements are not automatically covered by ISO 27001 — specific regulatory applicability needs separate assessment.

4.6 Audit Logs and Accountability

Audit logs answer accountability questions in a DFR project. FH2 supports operations troubleshooting and responsibility analysis through modular record capabilities.

Module	Record Capability	Usage Boundary
Automated workflow	Workflow task logs and operation logs can show execution status, operator, and operation records.	Used for workflow troubleshooting and operational records.
EventAPI	Provides 7-day delivery log statistics, including success rate, total events, failure count, and downloadable CSV for the last 7 days.	Used for event push delivery checks.
FlightHub Sync	Shows file synchronization status and errors, and supports downloading full synchronization history logs from the past 30 days.	Used for sync troubleshooting and archive checks.
Task and flight records	DJI Dock 3 supports viewing and exporting flight records, including task details, flight status, and device information.	Used for task review and flight process analysis.

4.7 Third-Party Integration Security

The commercial value of DFR comes from integration. Drone data must enter CAD, VMS, GIS, PSIM, Evidence, SOC, or other third-party business platforms to support response, handling, and review.

4.7.1 CAD / Alarm / PSIM: Alarm Trigger and Auto Dispatch Workflow

In Auto Dispatch Workflow scenarios, a third-party system sends events to an FH2 workflow through the workflow POST endpoint or inbound trigger endpoint. The request uses X-User-Token and lowercase x-project-uuid. FH2 applies level-threshold based auto-dispatch: event levels are 1–5, the default threshold is 3, and auto-dispatch fires only when event level $\geq$

threshold. In Auto-recommend mode, FH2 selects the most suitable idle device and uses the device list as fallback if no suitable device is available. In Manual assignment mode, the project can use a fixed device assignment or require a per-trigger manual pick.

POST Address: For public cloud users, the default addresses are International Server: <https://es-flight-api-us.djigate.com>; European Server: <https://es-flight-api-eu.djigate.com>. For on-premises users, use the custom domain: <https://{host}:30812/openapi/v0.1/workflow>.

Risk	Control
Forged alarm	Use authentication headers, limit request sources by IP records, and validate project ID and workflow ID.
Excessive alarms	Set automatic trigger level thresholds; low-level events should notify only and not dispatch automatically.
Incorrect coordinates	The third-party system should validate coordinate format and WGS84 coordinates; FH2 should combine this with flight safety checks.
No human response	Configure recipients and timeout. Cancel follow-up steps after timeout.
Wrong dispatch	Keep manual immediate-dispatch and dispatch-cancel paths available.

4.7.2 VMS / Command Center: Livestream Forwarding

FH2 supports forwarding livestreams to third-party live platforms. RTMP and RTSP are supported in both Public Cloud and On-Premises. GB28181 is only supported in On-Premises.

Risk	Control
Unauthorized live viewing	Third-party VMS and FH2 sharing links should each use independent accounts, access passwords, viewing permissions, and session controls.
RTSP address leakage	RTSP addresses should be generated and managed by FH2. VMS should use EventAPI or another mechanism to reacquire or update addresses.
Uncontrolled forwarding cost	In Public Cloud, projects should control channel count, viewer scope, and validity period.
Private deployment firewall policy	In On-Premises, inbound and outbound traffic between VMS / Media Server and FH2 should be allowlisted based on the project network design.

4.7.3 MQTT / GIS / PSIM: Telemetry Synchronization

Public Cloud uses Telemetry Data Forwarding under FlightHub Sync: FH2 acts as an MQTT client and pushes telemetry to a customer-managed MQTT broker. On-Premises uses MQTT Bridge: a broker-to-broker bridge configured via REST API POST /bridges/v1/resources. Both expose the thing/product/{sn}/... topic structure.

Risk	Control
Over-subscription	Limit subscription scope by project, device, topic, and business purpose.
Weak broker password	MQTT brokers must use independent accounts and strong passwords. Split accounts by project or system.
Unencrypted transmission	Prefer MQTTS (MQTT over TLS). Network teams should confirm MQTT / MQTTS ports, certificates, and firewall policies.
Network unavailable	When connection is interrupted, third-party systems should support reconnect, status alarm, and data recovery strategies.
Broker compatibility	If another customer broker is used, compatibility testing should verify authentication, TLS, topic, QoS, and message format.

4.7.4 Evidence / S3 / Archive: File Synchronization and Evidence Archiving

FlightHub Sync can synchronize media files, waylines, 3D / 2D models, flight records, and annotation information to third-party platforms. Media files and flight logs are FH2 → third-party only. Flight routes / waylines, 3D / 2D models, and custom zones / annotations are bidirectional where OpenAPI-based integration is used.

Risk	Control
File synchronization failure	Check sync status and error logs; download 30-day sync history; define retry processes for failed files.
Incorrect directory management	Use project-level sync rules and independent buckets. Object paths should include organization, project, task, or media file type identifiers.
Incomplete chain of custody	The third-party Evidence system should record receive time, object key, file size, hash, or platform-internal evidence ID.
Excessive permissions	Bucket permissions should follow least privilege. IAM policies should restrict bucket, prefix, read/write actions, and temporary credential validity.

4.7.5 EventAPI: Business Event Push

Risk	Control
Forged event	The receiver must validate x-dji-signature.
Duplicate event	The third-party system should process events idempotently based on event ID.
Excessive push scope	Rules should select all projects or specific projects intentionally.
Delivery failure invisible	Use 7-day delivery logs, success rate, failure count, and CSV download for troubleshooting.
Firewall blocking	FH2 uses fixed IPs for push. IPs differ by region and can be allowlisted accordingly.

4.7.6 OpenAPI and Frontend Components: Deep Integration

FH2 OpenAPI follows RESTful conventions and enables third-party platforms to interact with FH2. OpenAPI covers organization and project management, device management, livestream management, wayline management, task management, model management, and annotation management.

Risk	Control
Token leakage	Tokens should not be hardcoded in public frontends or shared documents. Expiration and rotation mechanisms should be defined.
Cross-project access	Each integration should bind to a clear project ID and projectToken. Avoid reusing credentials across projects.
Frontend component misuse	Components with control power, such as Virtual Cockpit and wayline editing, should only be authorized for necessary roles.
Missing API audit	Third-party systems should record API call logs and align them with FH2 project permission management.

4.8 Public Trust and Privacy Governance

Public safety DFR is often subject to privacy review. Privacy review focuses on purpose, scope, retention, access, and oversight. Data encryption is only one control point.

Governance Principle	Control
Purpose limitation	DFR should operate around defined event types. It should not become boundaryless persistent monitoring.
Area limitation	Use projects, task areas, custom flight areas, GEO Zones, and no-landing zones to define operating scope.
Least access	Command centers, pilots, device administrators, evidence administrators, IT administrators, and third-party systems should be authorized by role.
Retention classification	Event-related media, non-event media, automatic recordings, report materials, and legal-hold materials should use different retention strategies.
Transparency	Public safety projects can include public explanation, authorization process, data-use boundaries, and complaint channels.
Human-in-the-loop	The drone provides scene intelligence and evidence material. Enforcement or security decisions should still be made by authorized personnel.

4.9 IT Review Delivery Package

When a DFR project enters IT review, it should provide a reviewable, configurable, and traceable delivery package.

Deliverable	Purpose	Content
DFR data flow diagram	Explain where data comes from and where it goes.	Dock / aircraft / FH2 / third-party systems / storage / workflow POST or inbound trigger endpoint / outbound Webhook or EventAPI / MQTT / VMS / CAD.
Deployment architecture diagram	Explain Public Cloud or On-Premises boundaries.	Network zones, FH2 Server, database, object storage, MQTT Broker, VMS, Evidence, Firewall.
Deployment type comparison table	Support cloud or private deployment selection.	Data location, control rights, operations responsibility, feature differences, integration depth, limitations.
Data classification table	Support risk assessment.	Live video, telemetry, tasks, media, events, identity, operations records.
SSO / 2FA explanation	Support identity integration.	SAML SSO, 2FA, On-Premises OAuth2, local account boundary.
Third-party integration explanation	Support CAD / VMS / MQTT / S3 / EventAPI / OpenAPI review.	Inbound trigger headers, outbound webhook signatures, tokens, fixed IPs, topics, bucket permissions.
Storage and retention strategy	Define media and evidence lifecycle.	Auto recording for 7–90 days, media archive, file synchronization, auto-delete risk, deletion and recycle bin mechanisms.
Port and network list	Support firewall configuration by network teams.	On-Premises domains, ports, MQTT Broker, inbound trigger endpoint, outbound Webhook / EventAPI, VMS, third-party storage.
ISO 27001 evidence summary	Support DJI / FH2 security management system explanation.	BSI ISO 27001, design / development / operation services, risk management, security controls, continuous improvement.

4.10 Shared Responsibility Model

DFR data security is jointly delivered by DJI, integrators, and end users.

Responsible Party	Main Responsibilities
DJI / FH2	Provide FH2 platform capabilities, cloud-side security controls, identity and permission capabilities, transmission and storage protection, OpenAPI / EventAPI / FlightHub Sync interface security mechanisms, and ISO 27001 management system support.
Integrator	Design third-party system connections, configure interface authentication, validate network connectivity, integrate VMS / CAD / PSIM / Evidence systems, and deliver project-level data flow documentation.
End user	Manage organizational accounts, authorize project members, run internal approval processes, define data retention strategy, manage third-party system permissions, secure local networks, conduct compliance review, and run incident response processes.

FH2 provides information security capabilities and deployment choices. GDPR, NIS2, CJIS, and other legal or industry requirements need separate assessment based on project region, data type, organizational responsibilities, and deployment type.

4.11 Conclusion

The core of DFR data security is to clearly explain data collection, transmission, access, storage, forwarding, deletion, and governance.

FH2 provides value to DFR projects by acting as the unified platform layer. It connects docks, aircraft, mission plans, livestreams, media files, automated workflows, and third-party systems. It also provides an explainable, configurable, and reviewable data security foundation through RBAC, SSO, 2FA, transmission encryption, storage encryption, FlightHub Sync, EventAPI, OpenAPI, On-Premises deployment, and the ISO 27001 management system.

The project party still needs to make key decisions: choose cloud or private deployment, define who can access what, decide which data enters third-party systems, set media retention strategies, manage accounts and tokens, maintain local networks and storage, and place DFR inside its own IT, legal, and operations processes.

FH2 gives DFR teams response speed and gives IT teams critical control points. A reliable data security solution must be designed around deployment type, data flow, access control, storage strategy, and the organization's own governance requirements.

Appendix A: FlightHub 2 Internal Data Flow Reference

How to Use This Appendix

This appendix contains internal FlightHub 2 service architecture detail. The service component names used here (such as EMQX, SRS, and the internal service pipeline components) are DJI-internal identifiers — they are not externally documented APIs and cannot be directly accessed or called by third-party integrators. Their purpose in this appendix is to help integrators understand platform-side behavior at the integration boundary, not to expose integration endpoints.

Who should read this appendix: integration architects troubleshooting production issues, IT teams configuring network access, and technical leads preparing support escalations. This appendix is not required reading for initial integration design — the interface and scenario guidance in Sections 2.3 and 2.4 is sufficient for that purpose.

The four data flows below describe the internal processing sequences that underpin device registration, live streaming, mission dispatch, and media archiving. For each flow, this appendix identifies what triggers it, what an external system can observe at the integration boundary, and what symptoms indicate a platform-side failure rather than a configuration error on the integrator's side.

If you are troubleshooting an integration issue and need to identify whether a failure originated inside the FH2 platform or at your integration boundary, use the observable symptoms listed under each data flow. If you are filing a support ticket with DJI, the service component names in the detailed flow descriptions provide the vocabulary needed to describe the failure point accurately.

This appendix provides the complete component-level data flow documentation for FlightHub 2. It is intended for teams conducting deep technical integration, deployment troubleshooting, or platform architecture review.

Service topology knowledge has two direct applications: integration architecture design and deployment troubleshooting. This section is technical by design — worth reading carefully.

Data Flow 1: Device Registration & Command Delivery — Inside the FlightHub 2 Service Layer

Device-to-cloud connectivity underpins all FH2 operations. The flow below traces how it works — from service layer to component level.

Service Component Reference

Component	Role	Analogy
pf-device-service	Receives raw MQTT messages from EMQX; normalizes device protocol into platform-readable format	The protocol translator at the network boundary
pf-iot-core	Applies business rules; routes processed data to the appropriate downstream service	The routing engine — determines the next handler for every message
tas-service	Tracks mission state; manages the full wayline task lifecycle; triggers async downstream events	The task controller — maintains authoritative state for every active operation
es-management	Central management hub; coordinates read/write operations across the storage layer	The orchestration layer between business logic and persistence
MySQL / PG	Persistent storage for device state, mission records, and operational logs	The system of record — durable, structured, queryable
Redis	Cache for active device status and high-frequency access data	The in-memory index — low-latency reads for live operational data
RabbitMQ	Async message queue; routes inter-service events to trigger downstream workflows	The event bus — decouples services and absorbs load spikes

Core Services Involved

- **Device Registration** — On first connection, the device (Dock / RC) sends an HTTP request to obtain the MQTT relay service address and authenticate against the platform. This is the handshake that brings a device onto the FH2 network.
- **EMQX (MQTT Relay Service)** — The central message bus. All uplink traffic (device property reports, event notifications) and downlink traffic (service commands) pass through EMQX. Think of it as the air traffic control radio — every message between the platform and devices goes through this channel.
- **Business Service Layer** — Once messages arrive from EMQX, they enter a service pipeline that interprets device data, applies business logic, updates mission state, and issues commands back to the device. It is the operational brain behind every action visible to the operator.

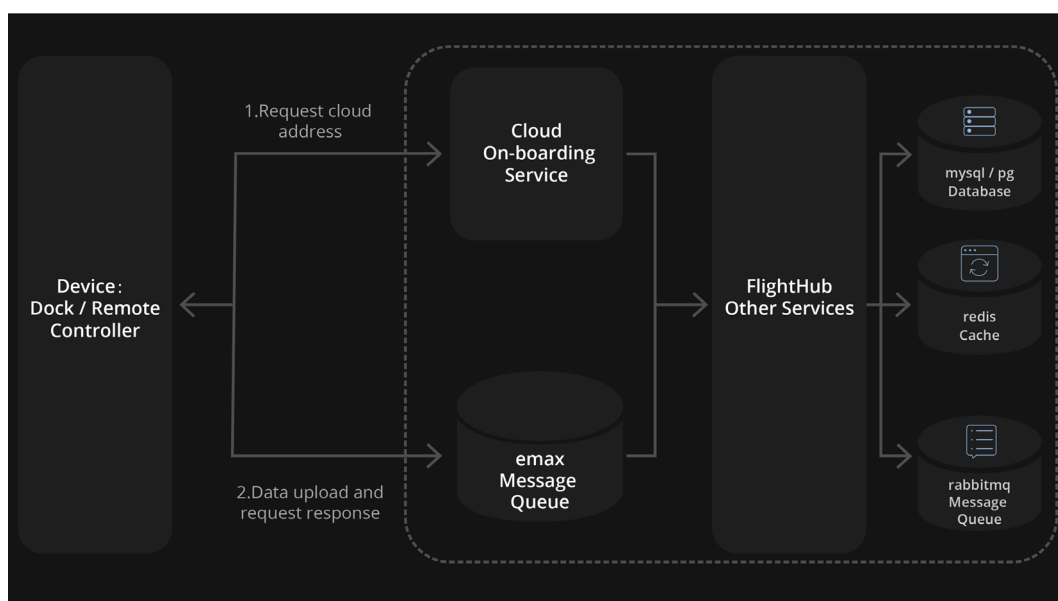


Figure A.1-A Data Flow 1 — FlightHub 2 Service Layer Architecture Overview

① Device Registration

When a Dock or RC comes online, it initiates registration with the FH2 platform to obtain a valid MQTT connection endpoint.

Device (Dock / RC)

```

|
|-- HTTP Request --> Cloud Onboarding Service
|
|<-- MQTT Address -----+
|
+-- MQTT Connect --> EMQX (Relay Service)

```

- Device sends HTTP request to obtain the MQTT relay address and authentication credentials
- Device establishes MQTT connection to EMQX and begins reporting device properties via `thing/product/{sn}/events`

② Device Cloud Connectivity & Command Delivery — Component Breakdown

The Business Service Layer comprises four service components and three infrastructure components. Here is how they are structured and how they interact:

Uplink — Device Property Reporting & Event Notification

Device state, telemetry, and event notifications originate at the device and travel upward through a sequential service chain — each stage with a defined responsibility:

10.pf-device-service receives the raw MQTT payload from EMQX and normalizes it into a structured, platform-readable format

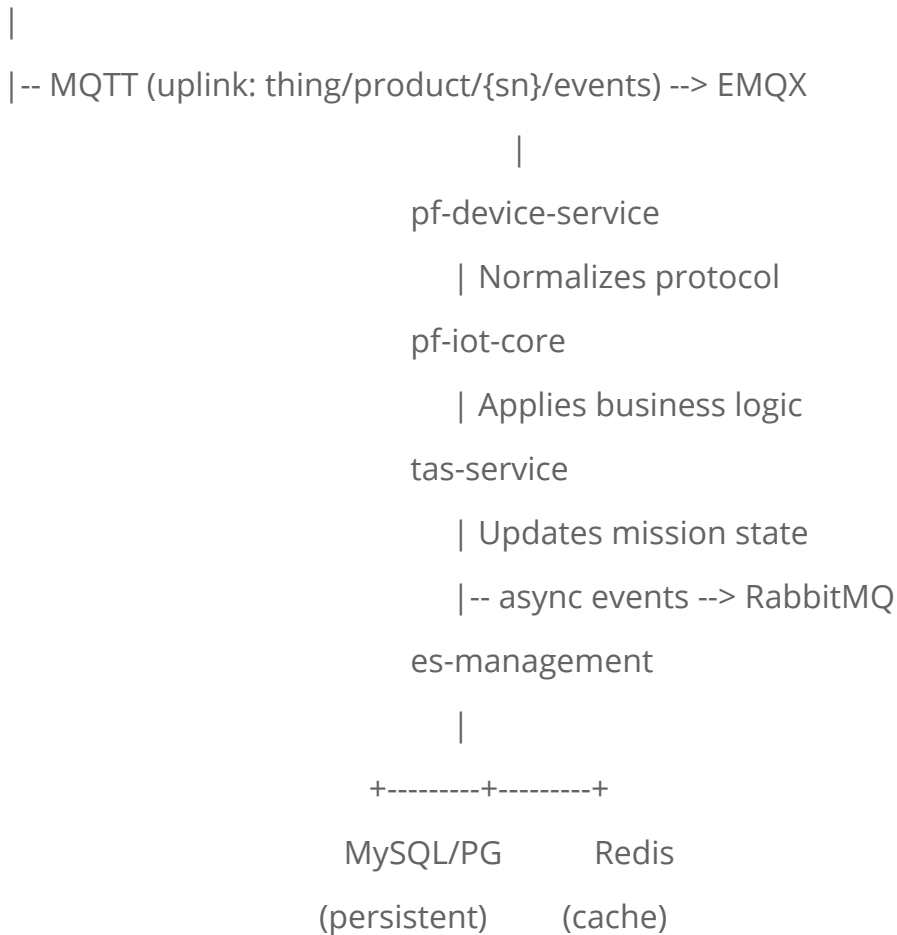
11.pf-iot-core applies business rules to the normalized data and determines the downstream routing path

12.tas-service updates mission state based on the incoming data and queues any follow-up actions via RabbitMQ

13.es-management writes the final record to MySQL/PG for persistence and refreshes the active device status in Redis for low-latency access

Outcome: the platform maintains a real-time, authoritative view of every connected device — ready to inform the next operator action or trigger an automated workflow.

Device



Downlink — Service Command Delivery

When an operator dispatches a mission or issues a command, the instruction traverses the service chain — each stage with a defined responsibility:

14.es-management receives the operator instruction from the frontend and initiates the command dispatch sequence

15.tas-service validates mission state, prepares the command payload, and triggers the downstream dispatch via RabbitMQ

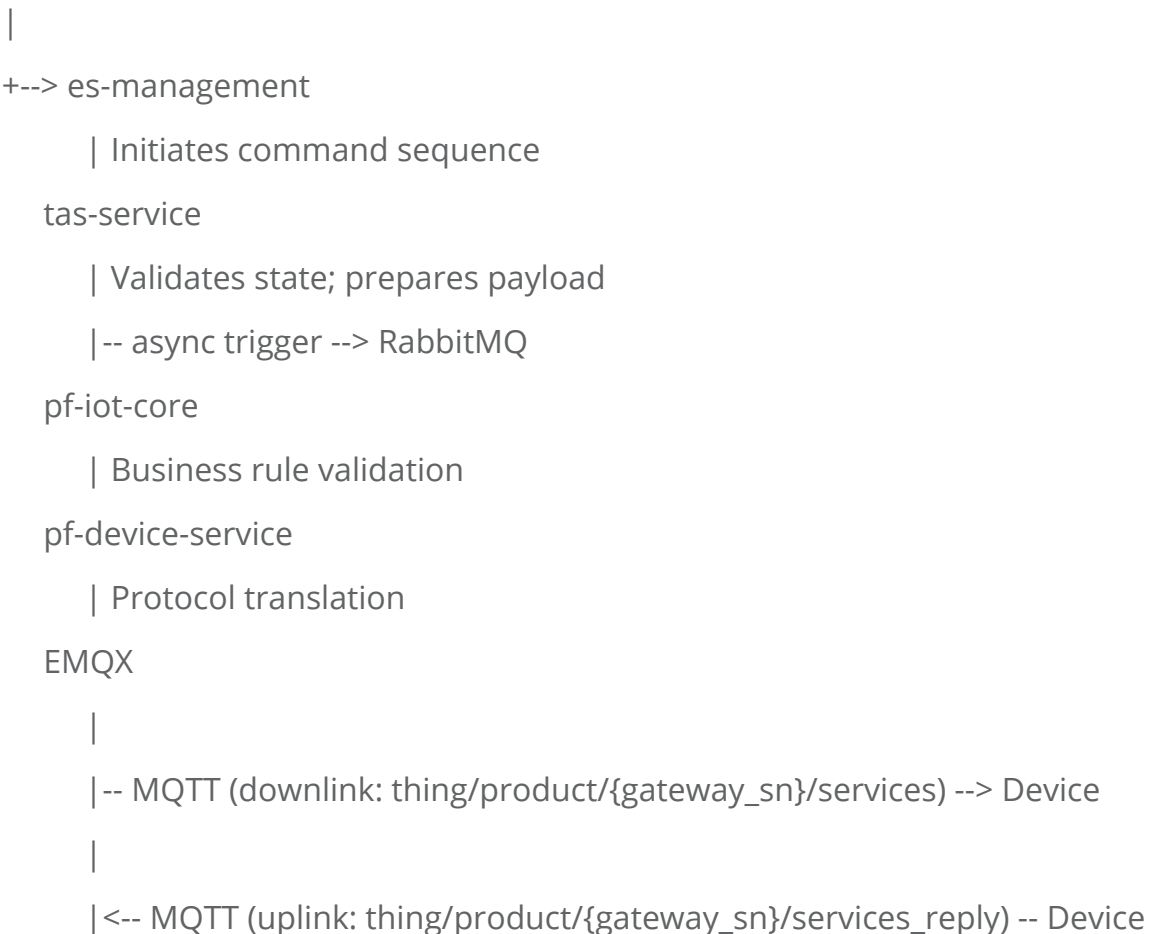
16.pf-iot-core receives the routed instruction, applies final business rule validation, and passes it to the device protocol layer

17.pf-device-service translates the instruction into the device protocol format and forwards it to EMQX for delivery

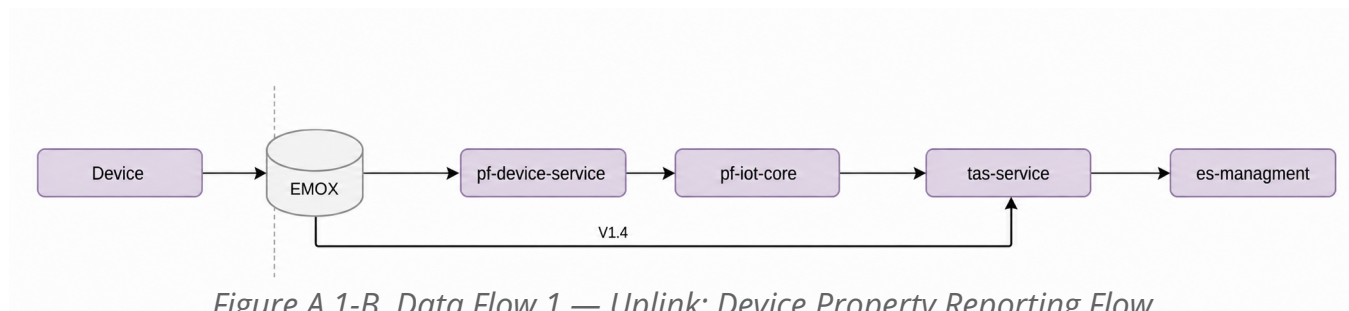
18.EMQX routes the outbound command to the target device via the designated downlink channel; device executes and returns acknowledgment via the reply channel

Outcome: every operator action is delivered to the device with a confirmed acknowledgment, with execution state tracked and persisted throughout.

Operator Action (Frontend)



- Commands delivered via thing/product/{sn}/services (downlink)
- Acknowledgment returned via thing/product/{sn}/services_reply (uplink)
- RabbitMQ propagates async execution events across services during command processing



Data Flow 2: From Lens to Screen — How Live Video Travels Through FlightHub 2

Live streaming is the real-time visual backbone of DFR operations. The flow below traces how a drone's camera feed reaches an operator's screen — from stream initiation to live playback.

Service Component Reference

Component	Role	Analogy
live-service	Manages stream session state; coordinates between device, SRS, and frontend	The session controller — owns the stream lifecycle from start to termination
SRS	Receives RTMP push stream from device; re-serves it as RTSP/RTMP to authorized clients	The broadcast relay — ingests one feed, distributes it to multiple consumers

Core Services Involved

- Live Streaming Service (live-service) — Manages the full streaming lifecycle: session initiation, device command dispatch, and viewer credential distribution. When an operator starts a live stream, live-service translates that action into a coordinated sequence across device, media server, and frontend — returning both a viewer UUID and stream token to the client upon successful session establishment.
- SRS (Simple Realtime Server) — The dedicated media server embedded in FH2. The device pushes one RTMP video stream into SRS; SRS then serves that same feed to multiple viewers simultaneously — command center display, field tablet, and third-party VMS can all pull from the same source at the same time. Think of it as a broadcast relay: one feed in, many viewers out.

- EMQX (MQTT Relay Service) — Carries all control signaling between the platform and the device: stream start/stop commands (downlink) and push confirmation (uplink). Video does not pass through EMQX — only control messages do. When a stream fails to start, EMQX is the first place to check for command delivery issues.

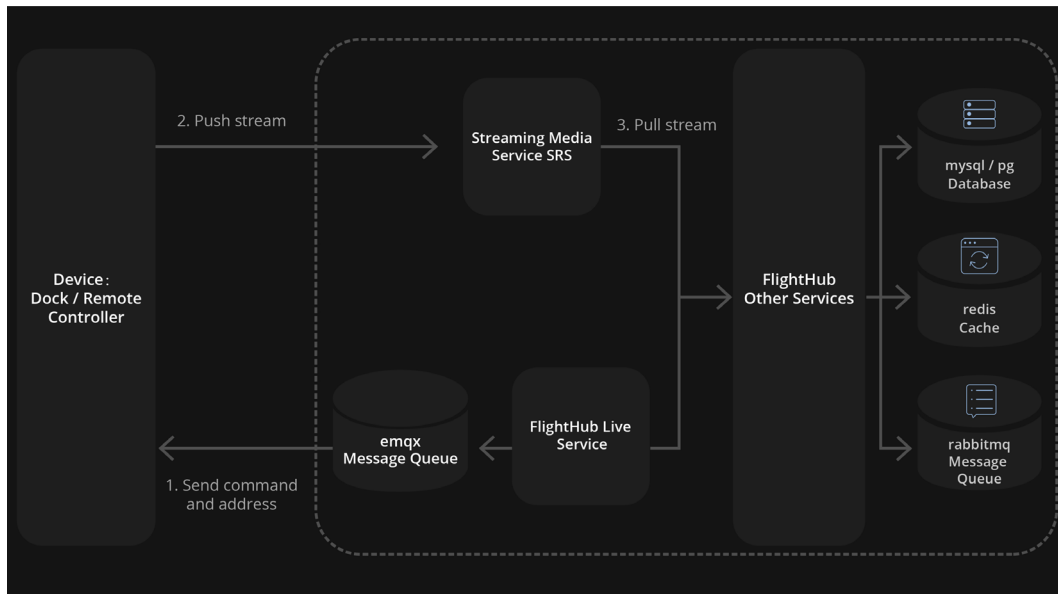


Figure A.1-C Data Flow 1 — Downlink: Command Delivery Flow

① Stream Initiation — Control Signaling

The operator triggers a live stream from the FH2 frontend. The platform issues a command to the device via MQTT, and the device prepares to push its video feed.

Operator (Frontend)

```

|
|-- HTTP Request --> es-management
|
| live-service
|
| |-- MQTT (downlink: .../services) --> EMQX --> Device
|
| <-- MQTT (uplink: .../services_reply) -- EMQX <-- Device
  
```

- Operator sends HTTP request to es-management to initiate a live stream session
- live-service issues a stream start command to the device via EMQX downlink channel
- Device parses the command, resolves the SRS push address, and confirms readiness via services_reply uplink

② Inside the Live Streaming Service — Components & Interactions

The following components are specific to the live streaming pipeline and handle the video path from device to viewer:

Components shared with Data Flow 1 (EMQX, es-management, Redis, MySQL/PG) retain the same roles as previously described. Their behavior in this flow is noted where relevant.

Uplink — Device Stream Push

Once the device receives the stream start command, it initiates a video push to SRS. Control confirmation travels back through EMQX while video takes a direct path to the media server.

19. Device encodes the camera feed and pushes it as an RTMP stream directly to SRS — bypassing EMQX entirely

20. SRS receives the inbound RTMP stream, validates push credentials, and registers the stream as available for pull

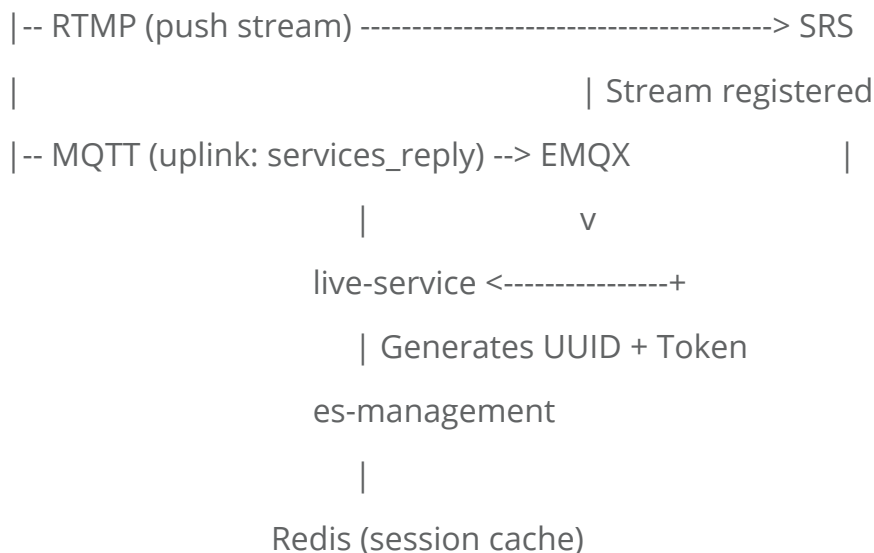
21. live-service is notified of successful stream registration; it generates a viewer UUID and stream token, returning both to the frontend via es-management

22. Redis caches the active session token for low-latency viewer authentication on subsequent pull requests

Outcome: the SRS media server holds a live, authenticated stream — identified by a unique UUID and secured by a viewer token, ready to serve authorized viewers on demand.

Device

|



Downlink — Stream Delivery to Viewer

With the stream registered in SRS and credentials issued, the frontend pulls the live feed directly from the media server.

23.Frontend receives the viewer UUID and stream token from es-management and uses them to authenticate against SRS

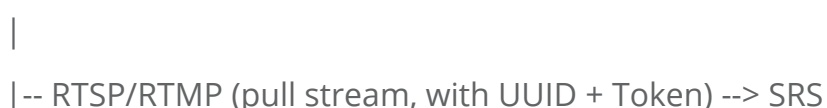
24.SRS validates the token via Redis cache and serves the live stream via RTSP or RTMP pull

25.live-service monitors session state throughout playback; on stream termination, it issues a stop command to the device via EMQX and clears the session from Redis

26.MySQL / PG records the completed stream session for historical reference and audit

Outcome: the operator receives a real-time aerial video feed authenticated and served directly from SRS — with session state fully managed by the platform throughout the stream lifecycle.

Frontend



| Validates token via Redis

live-service

| Monitors session state

| -- MQTT (stop) --> EMQX --> Device

|

MySQL/PG (session record)

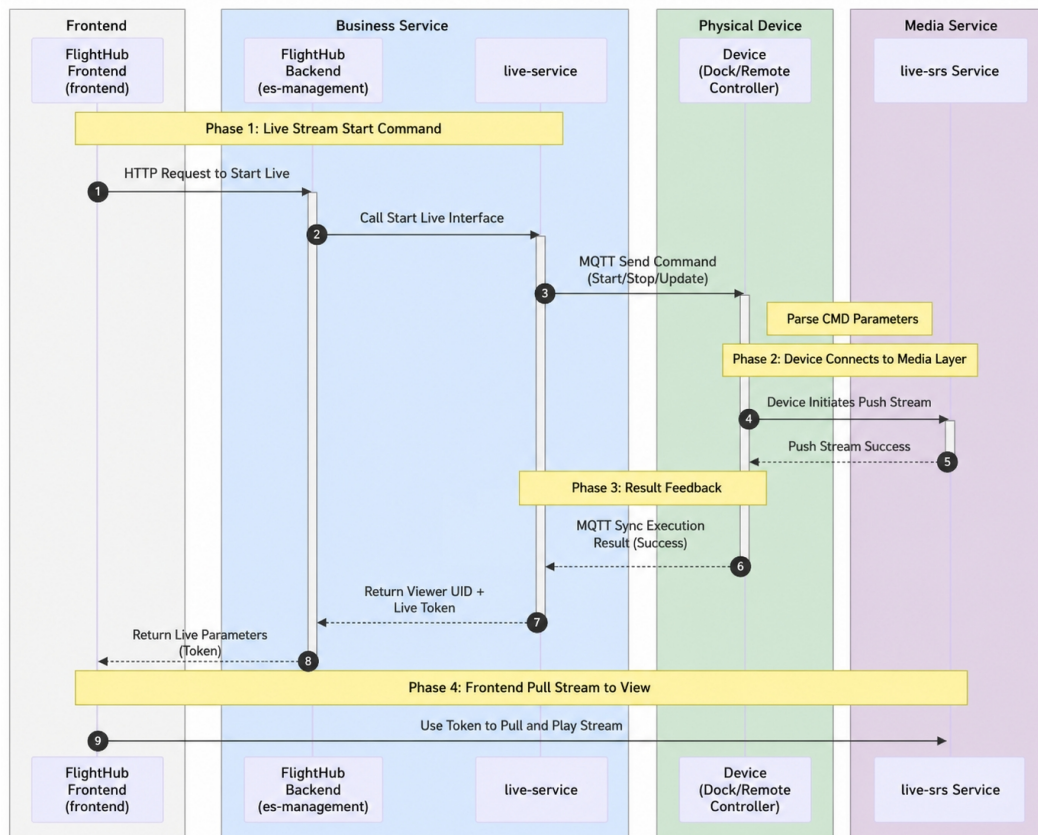


Figure A.2-A Data Flow 2 — Live Streaming: Stream Initiation Signaling

Data Flow 3: From Plan to Propeller — How a Wayline Mission Is Dispatched Through FlightHub 2

Before a drone can execute an automated flight, the platform must deliver the wayline file and mission parameters to the device in a precise sequence. The flow below traces how a mission dispatch works — from operator trigger to device execution.

Service Component Reference

Component	Role	Analogy
Traefik Gateway	API gateway — routes all inbound HTTP requests to the appropriate backend service	The switchboard — receives every call and connects it to the right department
es-task-management	Mission orchestrator — coordinates file access, constructs mission command payload, and triggers delivery	The mission controller — owns the full dispatch sequence from request to device acknowledgment
file-storage	Manages wayline file references and generates pre-signed STS URLs for device-side download	The file librarian — issues time-limited access passes to specific files
Minio	Object storage — holds all wayline route files; serves direct device downloads via STS URL	The file warehouse — stores the route; access controlled by pre-signed keys

Core Services Involved

- Task Management Service (es-task-management) — Orchestrates the full mission dispatch sequence: receives the operator's dispatch request, coordinates wayline file access, and delivers the mission command to the device. Every automated mission starts here.
- Minio (Object Storage) — Stores all wayline route files uploaded by operators. When a mission is dispatched, the device downloads the wayline file from Minio directly using a pre-signed STS URL embedded in the mission command — without routing through the business service layer.
- EMQX (MQTT Relay Service) — Carries the mission command (downlink) and flight

progress reports (uplink) between platform and device. The wayline file itself does not pass through EMQX — only the command containing the download reference does.

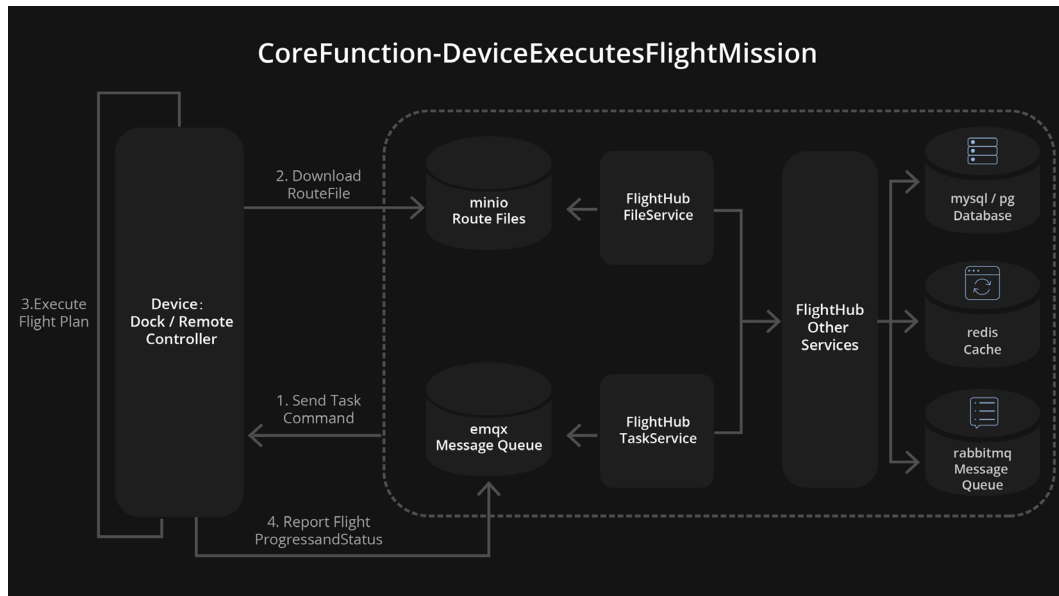
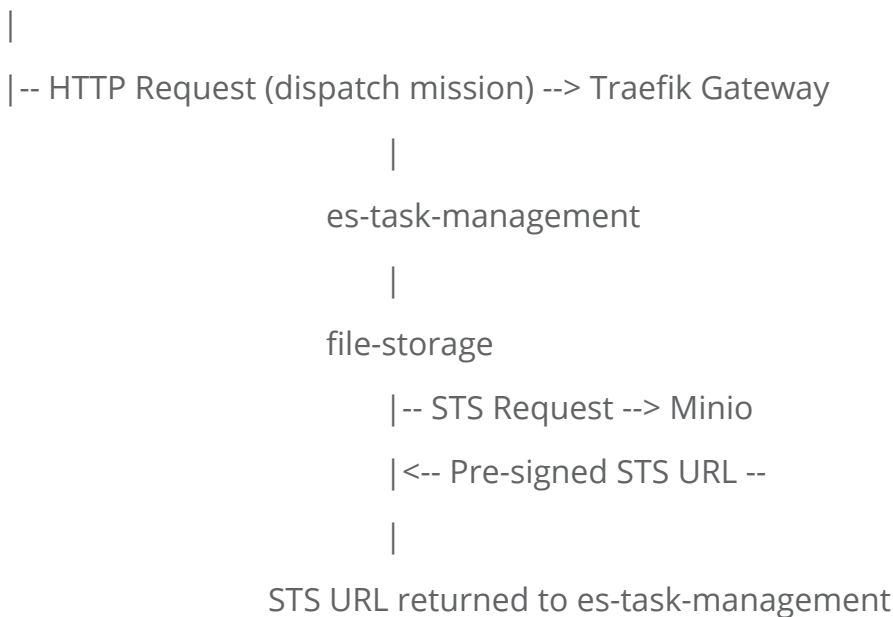


Figure A.2-B Data Flow 2 — Live Streaming: Device Stream Push Flow

① Wayline File Access Authorization

Before the mission command is sent, the platform generates a pre-signed STS URL that authorizes the device to download the wayline file directly from Minio.

Operator (Frontend)



- Operator sends mission dispatch request via HTTP through the Traefik API gateway
- es-task-management requests a pre-signed STS URL from file-storage

- file-storage generates a time-limited, scoped STS URL pointing to the specific wayline file in Minio
- STS URL is returned to es-task-management for embedding in the mission command payload

② Inside the Mission Dispatch Service — Components & Interactions

With the STS URL in hand, the platform constructs the mission command and delivers it to the device. This section breaks down how the dispatch service, gateway, and device interact during command delivery and execution.

Components shared with Data Flow 1 (EMQX, tas-service, RabbitMQ, MySQL/PG) retain the same roles as previously described.

Downlink — Mission Command Delivery & Wayline Execution

The platform delivers the mission command to the device via MQTT. The device uses the embedded STS URL to download the wayline file directly from Minio, then begins execution. 27.es-task-management constructs the mission command payload — containing RouteID, device SN, and the pre-signed STS download URL — and passes it to tas-service for delivery

28.tas-service calls the platform service layer via HTTP or RabbitMQ and triggers EMQX to push the mission command to the target device

29.Device parses the command, extracts the STS URL, and downloads the wayline file directly from Minio — bypassing the platform service layer entirely

30.Device loads the wayline file and begins autonomous flight execution; it continuously reports flight progress and status back to the platform via EMQX uplink

Outcome: the device receives a complete, authorized mission package — wayline file downloaded directly from object storage, mission parameters delivered via MQTT — and begins autonomous execution with continuous progress reporting back to the platform.

Operator (Frontend)

|

| -- HTTP --> Traefik --> es-task-management

| Constructs payload (RouteID + SN + STS URL)

tas-service

| -- HTTP/MQ --> EMQX

|

MQTT (downlink: .../services) --> Device

|

Device parses STS URL

|

S3 Download --> Minio (wayline file)

|

Device executes wayline

|

MQTT (uplink: .../events) --> EMQX --> tas-service

(flight progress)

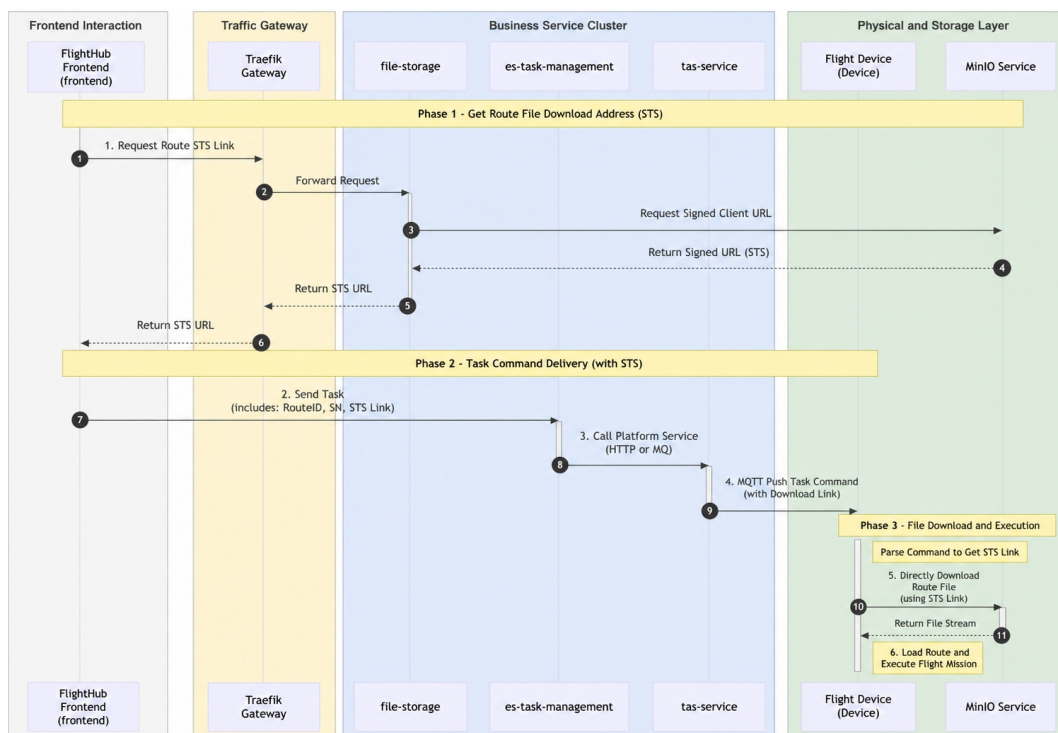


Figure A.2-C Data Flow 2 — Live Streaming: Stream Delivery to Viewer

Data Flow 4: From Flight to Archive — How Mission Media Is Uploaded Through FlightHub 2

After a mission completes, collected imagery and video do not transfer manually — FH2 orchestrates the entire upload, storage, and notification sequence automatically. The flow below traces how mission media travels from device storage to the platform archive.

Service Component Reference

Component	Role	Analogy
file-storage	Coordinates the full upload workflow: credential issuance, callback processing, and metadata registration	The logistics coordinator — manages access, confirms delivery, and updates the manifest
Minio	S3-compatible object storage; receives direct device uploads via pre-signed STS URL	The warehouse — stores the goods; access controlled by time-limited keys

Core Services Involved

- File Storage Service (file-storage) — The central coordinator for all media upload workflows. Manages pre-signed URL generation, upload credential issuance, upload completion callbacks, and metadata registration. When a device reports media ready for upload, file-storage owns the handoff from device to platform.
- Minio (Object Storage) — The S3-compatible object storage component embedded in FH2. All mission photos, videos, and AI result files are written here directly by the device using a pre-signed STS URL — no routing through the business service layer. Think of it as the warehouse: the platform issues the loading dock key, and the device delivers the goods directly.
- EMQX (MQTT Relay Service) — Carries device-side upload readiness notifications (uplink). Media files themselves do not pass through EMQX — only the control signals that initiate and confirm the upload sequence do.

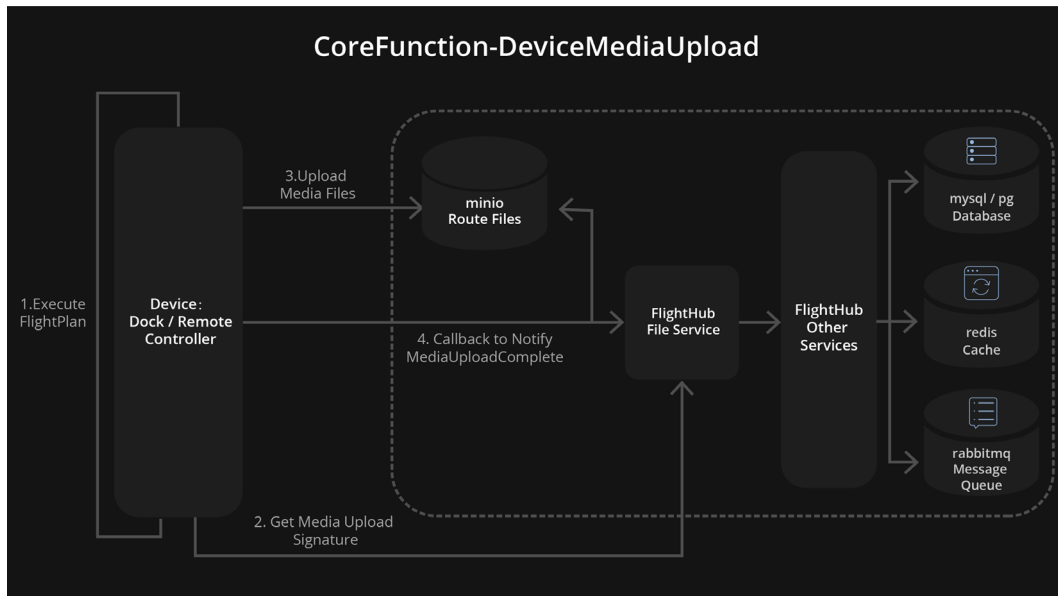


Figure A.3-A Data Flow 3 — Mission Dispatch: Wayline File Authorization

① Upload Credential Request

Before uploading, the device must obtain a pre-signed STS URL — a time-limited, scoped access token that authorizes direct write access to Minio.

Device (Dock / RC)

```

|
|-- HTTP Request --> file-storage
|
|-- STS Request --> Minio
|<-- Pre-signed STS URL --
|
|<-- STS URL returned ----+

```

- Device requests upload credentials from file-storage via HTTP
- file-storage generates a pre-signed STS URL scoped to the specific mission and file set
- STS URL is returned to the device, carrying a time-limited write authorization directly to Minio

② Inside the Media Upload Service — Components & Interactions

With upload credentials in hand, the device writes media directly to Minio. This section

breaks down the components from credential issuance through to metadata persistence and platform notification.

Components shared with Data Flow 1 (EMQX, tas-service, es-management, PostgreSQL) retain the same roles as previously described.

Uplink — Device Media Upload

The device uploads mission media directly to Minio using the STS URL, then notifies the platform via HTTP callback.

31. Device uploads photos and videos directly to Minio using the pre-signed STS URL — the upload bypasses the business service layer entirely

32. Device sends an HTTP callback to file-storage upon upload completion, reporting the file list and upload status

33. file-storage validates the callback, registers file metadata to PostgreSQL, and notifies es-management of the completed upload

34. es-management updates the mission record and makes the media accessible in the FH2 media library; tas-service updates task completion state and triggers any configured downstream workflows via RabbitMQ

Outcome: mission media is stored in Minio, indexed in PostgreSQL, and immediately accessible through the FH2 media library — with task completion state updated and any downstream workflows (e.g., Cloud Modeling, third-party sync) triggered automatically.

Device

```
|  
|-- S3 Upload (via STS URL) -----> Minio  
|                                     | Upload registered  
|-- HTTP Callback (upload complete) --> file-storage <-----+
```

| Validates & registers metadata
 PostgreSQL
 |
 es-managment
 | Updates media library
 tas-service
 | Updates task state
 |-- async trigger --> RabbitMQ
 (Cloud Modeling / third-party sync)

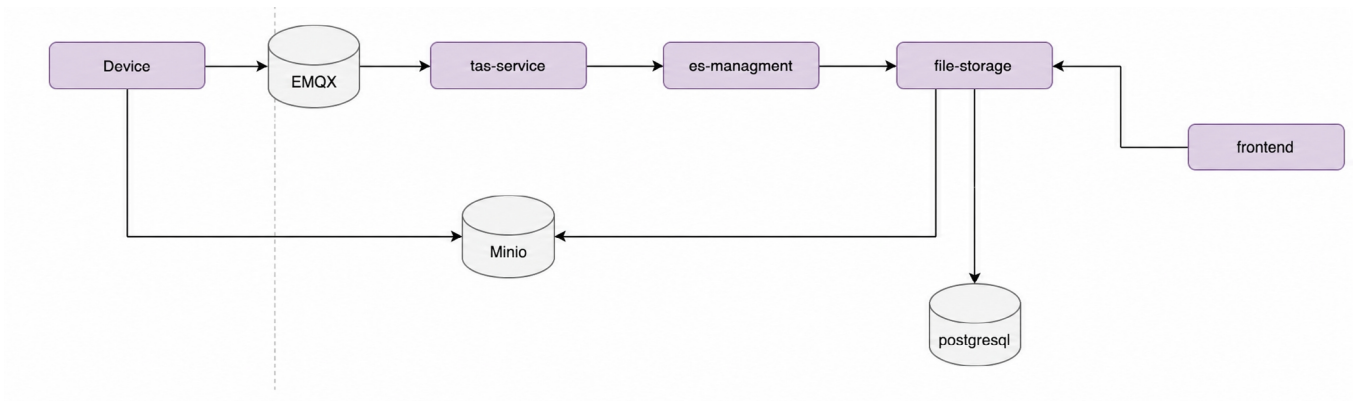


Figure A.4-A Data Flow 4 — Media Upload: Upload Credential & File Transfer Flow

Appendix B: FlightHub 2 On-Premises Network Port Configuration Reference

This appendix provides the complete network port reference for FlightHub 2 On-Premises deployments. It covers ports required for device and platform access, internal management ports that must not be publicly exposed, and reference links for the full deployment manual and security architecture documentation.

Subnet Configuration

The following subnets are occupied by FlightHub 2 On-Premises. Ensure they do not conflict with existing network ranges before deployment.

Subnet	Description	Usage
10.0.0.0/4	FlightHub 2 On-Premises internal subnet	SUB_NET_DB
10.0.1.0/4	FlightHub 2 On-Premises internal subnet	SUB_NET_GATEWAY
10.0.2.0/24	FlightHub 2 On-Premises internal subnet	SUB_NET_SELF_SERVICE

B.1 Ports Required for Device and Platform Access

The following ports must be open for inbound access from devices and authorized users. Source IP ranges should be restricted to known device and user address ranges wherever possible. Ports marked "Device connectivity" must be reachable from Dock / drone IP ranges; ports marked "Office access" should be restricted to corporate networks.

Port	Protocol	Service	Purpose	Access Type
4000	TCP	Deployment Tool (Visualization)	Visual deployment tool access	Office access
80	TCP	FlightHub 2 Web (HTTP)	Platform web access	Office access
443	TCP	FlightHub 2 Web (HTTPS)	Platform web access (encrypted)	Office access
123	UDP	NTP Time Synchronization	Device time sync with FlightHub 2	Device connectivity
1883	TCP	EMQX MQTT	Device MQTT connection to FlightHub 2	Device connectivity
8883	TCP	EMQX MQTTS (Encrypted)	Device encrypted MQTT connection	Device connectivity
20000	TCP	Agent-EMQX	Device connection to FlightHub 2	Device connectivity
20001	TCP	Agent-EMQX (Encrypted)	Device encrypted connection	Device connectivity
1935	TCP	RTMP Live Streaming	Live video push; office + device access	Office + Device
8000	TCP	WebRTC Live Streaming	Live video pull; office + device access	Office + Device
30802	TCP	MinIO Storage Service	Device media upload to object storage	Device connectivity
30804	TCP	MinIO-Static Service	Platform static file service; office + device	Office + Device
30806	TCP	Livestream Service	Live streaming; office + device access	Live streaming; office + device access
30808	TCP	Agent-EMQX WebSocket	WebSocket service; office + device access	WebSocket service; office + device access
30812	TCP	FlightHub 2 Backend Service	Core platform service; office + device access	Core platform service; office + device access

Port	Protocol	Service	Purpose	Access Type
30813	TCP	Map-Custom Service	Map customization service; office + device	Office + Device
30814	TCP	Map-Server Service	Map server service; office + device	Office + Device
38080	TCP	Bridge API Service	Bridge API; office access required	Office access
38081	TCP	TAS API Service	Task API service; device connectivity required	Device connectivity
30826	TCP	Terra Connection / K3s Service	Terra access to FlightHub 2; K3s service	Office access
30827	TCP	Terra Connection / K3s Service	Terra access to FlightHub 2; K3s service	Office access
19090	TCP	Dronenet Monitor Service	Monitoring service; office access	Office access
30828	TCP	Loki Log Component	Log aggregation service; office access	Office access
30816	TCP	RabbitMQ Message Queue	Message queue; office access	Office access
30824	TCP	TS-Gateway Service	Gateway service; office access	Office access
30825	TCP	TS-Admin Service	Admin service; office access	Office access
30829	TCP	Elevation Data Service	Terrain elevation data; office access	Office access
30830	TCP	DRTC-Manager Livestream Service	Live streaming management; office access	Office access
30854	TCP	Live-Stream-Hub	Livestream hub; office access	Office access
10000–10100	UDP	Livestream Service (UDP Range)	UDP ports for live streaming	Office + Device

Security note: All communication between DJI Dock / drones and FlightHub 2 is encrypted using AES-256 with authenticated session establishment. If docks connect via a dedicated private network or VPN tunnel, device-facing ports (MQTT, transmission) do not need to be exposed to the public internet. Apply firewall rules scoped to private network CIDR ranges only.

B.2 Internal Management Ports (Must Not Be Publicly Exposed)

The following ports are used for internal FlightHub 2 operations and administration. They must not be exposed externally. Ports marked "FlightHub 2 internal use only" require no firewall configuration.

Port	Service	Note
8080	Gateway Dashboard	Internal ops only — do not expose externally
30800	O&M Management Portal	Internal ops only — do not expose externally
30801	MinIO Admin Console	Internal ops only — do not expose externally
30805	Terra Backend Service	FlightHub 2 internal use only
30811	Grafana Monitoring Console	Internal ops only — do not expose externally
30815	Node Exporter (Metrics)	FlightHub 2 internal use only

B.3 Hardware Sizing Reference

The following specifications provide sizing guidance by fleet scale for On-Premises server infrastructure.

Fleet Scale	vCPU	Memory	Data Disk	Outbound Bandwidth
~20 Docks	8	32 GB	1 TB SSD	≥ 100 Mbps
~50 Docks	16	64 GB	3–4 TB SSD	≥ 250 Mbps
~100 Docks	32	128 GB	5 TB SSD + cold storage	≥ 500 Mbps

Separate the OS disk from the data disk. Use SSD for primary data storage. For large-scale media archiving, consider a dedicated cold storage tier.

B.4 Reference Documentation

FlightHub 2 On-Premises Version Deployment Manual (V1.5)

FlightHub 2 On-Premises Version Deployment Manual (V1.5)

DJI Drone Security White Paper V3.1 (2025)

DJI Drone Security White Paper V3.1 (2025)

Contact

400-700-0303

DJI Enterprise Email

enterprise@dji.com

Address

DJI Sky City, Xianyuan Road, Xili Subdistrict, Nanshan District, Shenzhen,
Guangdong Province, China

Postal Code

518000