



LANDING.AERO / RESEARCH & PRACTICAL GUIDANCE

Aircraft Tail Swaps: An Event Contract for Airline Systems

LANDING.AERO is a Montreal software studio that builds custom, AI-assisted software for flight operations: operations tools, data integration, workflow automation and dashboards designed around how an airline, charter operator or flight department actually works.

Published by LANDING.AERO · 2026-10-24 · aircraft tail swap · aircraft swap process · operations control center · flight operations integration · aidx · event contracts · crew scheduling · aircraft rotation · airline software

Original article: <https://www.landing.aero/articles/en/aircraft-tail-swaps-event-contract>



In this article

1. Executive Summary
 2. Introduction and Background
 3. Key Changes
 4. Implementation Considerations and Process Changes
 5. Failure Semantics and Recovery
 6. Packaged Coverage and the Bridge Decision
 7. Data Analysis and Evidence
 8. Implications and Future Directions
 9. Conclusion
-

Executive Summary

An **aircraft tail swap** is a change to the individual aircraft assigned to a flight leg, potentially accompanied by an equipment-type change. Jeppesen's published aircraft-assignment interface distinguishes those changes. (www.jeppesen.com) Flight Information Exchange Model (FIXM) identity requirements explain that an aircraft substitution can change registration without changing the flight identification. (fixm.aero) The integration question is therefore which assignment is authoritative, when it applies, and whether downstream systems have applied the same version.

Aviation Information Data Exchange (AIDX) provides an operational-flight-data vocabulary. The International Air Transport Association (IATA) describes approximately **180 data elements**. (www.iata.org) Its implementation guide leaves transport unspecified and recommends bilateral implementation agreements. (www.iata.org) A documented aircraft-registration field does not establish the operator's approval authority, transaction boundary, or completion policy. This report proposes those elements as an operator-specific contract rather than additional AIDX requirements.

The proposed design separates **proposed, approved, published, and acknowledged** states. An approved assignment receives a monotonically increasing version for its flight leg; every consumer reports the version it applied. Event identity, payload-schema version, and assignment version have separate purposes. CloudEvents requires distinct events to have unique source-and-identifier combinations; its specification version describes the envelope. (github.com) It does not version the aircraft assignment. Broker confirmation also cannot substitute for application acknowledgment, as RabbitMQ explicitly distinguishes publisher confirms from consumer acknowledgments. (www.rabbitmq.com)

Packaged capability should be assessed first. PDC describes shared schedule data between planning and day-of-operations optimization. (www.pdc.com) Lufthansa Systems describes integrated tail assignment; IBS Software describes integrated operational and crew recovery. (www.lhsystems.com) (www.ibsplc.com) These are vendor descriptions, not measured proof of an operator's complete subscriber handoff. Broker settings also impose concrete boundaries: Amazon Simple Queue Service (SQS) first-in, first-out (FIFO) queues deduplicate within **5 minutes**. (docs.aws.amazon.com) Azure Service Bus documents a **10-minute default** and a configurable range from **20 seconds to 7 days**. (learn.microsoft.com) Those windows are transport

parameters, not business-replay guarantees. The decision should follow the acceptance matrix below: retain an existing product when its interfaces pass, configure a supported gap where possible, and commission a narrow bridge only for missing ownership, versioning, reconciliation, or acknowledgment behavior.

Introduction and Background

For this report, an aircraft tail swap means replacing the aircraft assignment associated with a published flight leg. It does not mean changing an aircraft's registration. The distinction between the flight and its assigned aircraft matters: Avinor describes aircraft registration as identifying an aircraft, while FIXM's identity requirements explicitly discuss substitutions that preserve flight identification. (xwiki.avinor.no) (fixm.aero)

The audience is the airline **operations control center (OCC)** manager or integration lead investigating a same-day handoff. The question is whether a change approved in rotation management reaches crew scheduling, flight planning, station operations, passenger systems, and maintenance interfaces with an unambiguous identity and version. The report addresses data ownership and acceptance evidence. Operational decisions remain governed by the operator's approved procedures.

AIDX is relevant because IATA defines it as a global **Extensible Markup Language (XML)** messaging standard for exchanging flight data between airlines, airports, and other participants. Its scope includes operational flight-leg information, rather than a prescribed application architecture. (www.iata.org)

Likewise, an optimizer's output and a committed assignment should be distinguished. N-Ops & Crew documentation provides a concrete example: its displayed tail-assignment optimization result is not saved until the user selects Save. That documented separation supports testing the point of commitment rather than assuming every displayed scenario is already effective. (n-raido.support.skywise.com)

All designs, example records, state names, thresholds, and calculations below are **proposals or synthetic examples**, unless explicitly attributed to a source. Public interfaces establish useful vocabulary and product scope; the installed interface profile remains the acceptance target. Research reflects documentation retrieved on October 24, 2026. LANDING.AERO is considered only as a custom integration procurement option in the decision table, rather than as an OCC product.

Key Changes

Separate flight-leg identity from aircraft identity

The proposed canonical key is a stable **flightLegId**, with mappings to the identifiers used by each connected system. Do not generate a new canonical leg merely because its assigned aircraft changes. Avinor's flight-leg identifier documentation requires enough attributes to uniquely identify the leg; its separate identification documentation warns that a flight identifier is not unique across time. (xwiki.avinor.no)

A proposed mapping record should preserve:

- **Operating identity:** the operating carrier, flight number, suffix where used, and the agreed operating-date interpretation.

- **Leg boundaries:** departure and arrival identifiers, plus a stable occurrence identifier where a route repeats.
- **External aliases:** each source's leg key and the marketing-flight references that point to the operating leg.
- **Aircraft reference:** a stable aircraft-master identifier and its current registration as separate assignment data.

These fields are an internal design proposal, not a claim that every exchange standard uses the same natural key. FIXM's discussion concerns air traffic management identification; its substitution example supports separating concepts without making a FIXM identifier a replacement for every airline commercial identifier. (fixm.aero)

Distinguish envelope version, schema version, and assignment version

Use **eventId** to identify the emitted event, **schemaVersion** to identify its payload contract, and **assignmentVersion** to identify committed business state for the leg. A retransmission of the same event should preserve its event identity. A newly committed reversal should receive a new assignment version.

CloudEvents supplies required context attributes including `id`, `source`, `specversion`, and `type`. It also recommends a different schema identifier for incompatible schema changes. Neither mechanism supplies the proposed airline assignment-version policy. (github.com)

Require consumers to compare assignment versions before changing their local projections. The comparison rule should specify the version's scope, initialization, and behavior after migration or restoration. A timestamp alone should not decide which competing assignment wins.

Separate a proposal from a committed change

The proposed state model has distinct meanings:

- **Proposed:** a scenario exists and may still change.
- **Approved:** the designated authority has committed an assignment and version.
- **Published:** the committed event has entered the agreed distribution mechanism.
- **Acknowledged:** the required consumer has durably recorded its application result for that version.

Approval is an operator-defined authority boundary. Publication and acknowledgment are distribution evidence. Lufthansa Systems describes an optional control where aircraft legs, rotations, and tail assignments remain unchanged until human validation, illustrating why proposal handling deserves an explicit interface boundary. (www.lhsystems.com)

An acknowledgment should identify its consumer, leg, event, applied assignment version, result, and observation time. It should never be interpreted as approval by another department unless the operator's contract explicitly assigns that meaning.

Pin the deployed exchange profile

SITA documents IATA_AIDX_FlightLegNotifRQ/FlightLeg/LegData/AircraftInfo/Registration and states that its AIDX application programming interface (API) supports **v21.2**. (www.developer.aero) These are verified documentation statements about that profile, rather than certification of the newest downloadable IATA schema.

IATA warns that implementation guides may lag releases, and its standards-download history records a later AIDX qualifier annotation change. The latest official XML Schema Definition (XSD) package was not directly inspected for this report. Before implementation, validate mappings against the exact contracted XSD and deployment profile. Do not infer compatibility from a guide filename alone. (www.iata.org) (standards.atlassian.net)

Implementation Considerations and Process Changes

Assign authority field by field

The proposed contract appoints a single authority for committed assignment state while preserving the responsibilities of consuming systems. IATA's guide calls for bilateral implementation agreements, making this ownership record an appropriate implementation deliverable rather than an assumed property of the message vocabulary. (www.iata.org)

Table 1 proposes field ownership. The entries are acceptance requirements to agree with the operator, not statements about every vendor's data model.

Field or record	Proposed owner	Meaning and acceptance evidence
flightLegId and aliases	Flight identity master or designated source	Same operating leg resolves consistently before and after a swap.
aircraftId, registration, configuration reference	Aircraft master	Registration resolves to the intended aircraft; configuration has its own reference.
assignmentVersion	Committed-assignment authority	Strictly advances for each committed change to that leg.
effectiveAt	Assignment authority	Business applicability time, distinct from publication or receipt.
committedAt and approval reference	Assignment authority	Identifies the durable commitment and its audit reference.
eventId, source, schema reference	Publisher or source adapter	Retransmission preserves event identity; schema changes are explicit.
swapId, changed-leg manifest	Assignment authority	Declares the intended scope of a multi-leg change.
Previous version and superseded event	Assignment authority	Makes stale updates and reversals recognizable.
Consumer result and applied version	Each subscriber	Records applied, rejected, or review-required outcomes without changing source authority.

Field or record	Proposed owner	Meaning and acceptance evidence
Required-consumer policy	Operator-designated process owner	States which results are necessary to close distribution for each event class.

The table separates **data authority** from **distribution responsibility**. An adapter can publish a source's committed assignment without acquiring the right to approve one. Where a legacy source cannot expose a durable version, define and document the bridge's serialization and reconciliation policy before treating its generated version as authoritative.

Map the consumers and their responsibilities

The subscriber list should follow actual installed-system boundaries. Jeppesen's documented OC026 messages can communicate aircraft and crew assignment data. (www.jeppesen.com) N-Flight Planning documents an Aircraft field containing the assigned aircraft's fleet identifier or registration. (n-flightplanning.support.skywise.com) These examples show why copying a registration to a display is narrower than proving all dependent applications have applied it.

Use a subscriber register with an owner and an observable result for each interface:

- **Rotation management:** apply the affected-leg manifest and identify any locally stale links.
- **Crew scheduling:** record the assignment version reviewed or applied and retain its own workflow outcome.
- **Dispatch and flight planning:** resolve the aircraft reference and report the status of dependent data handling.
- **Station operations:** update the agreed station projection and report receipt separately from application.
- **Passenger systems:** apply relevant aircraft or configuration changes under their own documented process.
- **Maintenance interfaces:** reconcile assignment references while preserving maintenance-record ownership.

These are proposed interface responsibilities. They do not prescribe crew decisions, dispatch actions, passenger handling, or maintenance release. Lufthansa Systems says its crew and OCC products allow teams to evaluate identical scenarios simultaneously; Swiss-AS describes AMOS planning as supporting scheduled and unscheduled maintenance events. Neither statement establishes that a particular cross-vendor deployment implements this acknowledgment contract. (www.lhsystems.com) (www.swiss-as.com)

Station scope also deserves explicit tests. EUROCONTROL describes **Airport Collaborative Decision Making (A-CDM)** milestones as triggers for downstream estimate updates, and its specification describes a registration discrepancy alert against a planned rotation. These are airport information-sharing mechanisms, not proof that the airline's passenger or crew subscriber applied the same version. (www.eurocontrol.int)

Publish a small, testable contract

Use a machine-readable contract for channels, payloads, correlation, and expected responses. AsyncAPI describes message-driven APIs and provides a correlation-identifier definition for tracing or matching messages. It can document the proposed exchange without deciding the airline's authority model. (www.asyncapi.com)

Hypothetical Example: synthetic committed-assignment record. The identifiers, versions, and times below are invented test data. The record is a proposed application payload, not an AIDX message or a complete CloudEvents envelope.

```
{
  "eventId": "evt-example-b",
  "flightLegId": "leg-example-eastbound",
  "assignmentVersion": 18,
  "previousAssignmentVersion": 17,
  "aircraftId": "aircraft-example-b",
  "previousAircraftId": "aircraft-example-a",
  "swapId": "swap-example",
  "changedLegs": ["leg-example-eastbound"],
  "effectiveAt": "2026-10-24T14:00:00Z",
  "committedAt": "2026-10-24T13:42:00Z",
  "schemaVersion": "assignment-contract.example"
}
```

The proposed validation policy should require essential fields, distinguish absent values from explicit clearing, and define how unknown fields are handled. For JavaScript Object Notation (JSON), JSON Schema makes properties optional by default unless required; it also distinguishes null from an absent property. Structural validity must therefore be designed explicitly, and it still does not establish business approval. (json-schema.org)

Make the sequence observable

Hypothetical Example: event sequence. This diagram describes the proposed handoff, not a documented vendor workflow.

```
sequenceDiagram
    participant O as OCC approval workflow
    participant A as Assignment authority
    participant P as Publisher
    participant C as Consumer
    participant M as Distribution monitor
    O->>A: Approve candidate with expected prior version
    A->>A: Commit assignment and outbox record
    A->>P: Expose committed event
    P->>C: Publish assignment event
    C->>C: Check identity, schema and version; persist result
```

```
C->>M: Report applied version or explicit exception
M->>M: Compare results with required-consumer policy
```

Microsoft's transactional-outbox explanation stores a business object and its events in the same database transaction. Where supported by the assignment source, this pattern addresses the gap between committing data and making an event available for publication. An external adapter needs equivalent source-supported recovery evidence rather than assuming it shares the source's transaction. (learn.microsoft.com)

Keep tracing separate from assignment correctness. World Wide Web Consortium (W3C) Trace Context standardizes propagation of tracing information across services. A trace can help diagnose delay, while the assignment version and consumer result establish which business state was applied. (www.w3.org)

Failure Semantics and Recovery

Deduplicate without suppressing later business changes

Define **idempotency** as the consumer's ability to process a repeated event without duplicating its intended side effects. Amazon Web Services (AWS) recommends tracking processed messages in idempotent consumers, while CloudEvents defines uniqueness through the combination of source and event identifier. (docs.aws.amazon.com) (github.com)

The proposed consumer should atomically persist its applied projection and processing result where its storage model permits. Otherwise, document the recovery mechanism that prevents a crash between those writes from generating contradictory acknowledgments. A duplicate of an applied event should return the recorded result, rather than create another downstream change.

Use explicit handling rules:

- **Same event, same content:** return the existing processing result.
- **Same event, different content:** reject the conflict and retain evidence for reconciliation.
- **New event, newer assignment:** apply according to the declared effective-time policy.
- **New event, older assignment:** record stale delivery without regressing the current projection.
- **Missing predecessor:** reconcile through a versioned snapshot or retrieve the missing history.
- **Unknown schema:** isolate the event with an observable exception rather than silently defaulting fields.

Kafka documents ordering within a topic partition, while Google Pub/Sub does not promise order across different ordering keys. Therefore, neither transport should be assumed to serialize an entire multi-leg swap merely because it can order messages for a particular key. (kafka.apache.org) (docs.cloud.google.com)

Define supersession, late arrival, and reversal

Supersession means a newer committed assignment replaces an older one within a declared scope. **Late arrival** means delivery time trails the event's business applicability. The proposed contract preserves both effective and observed times; it does not reinterpret a late event as the newest business state merely because it arrived last.

A reversal is a new committed change, with a new event identity and a higher assignment version, referencing the state it replaces. It should not delete the intervening audit history. Distinguish that from a database transaction that never committed: AWS's outbox guidance says not to send an event notification if the transaction rolls back. (docs.aws.amazon.com)

For a multi-leg change, publish a manifest of affected legs and their target versions. If the source supports atomic group commitment, expose that boundary. If it does not, the bridge should report incomplete groups and reconcile them under an agreed policy. A shared correlation identifier alone should not be treated as proof of atomic application.

For HTTP-based command interfaces, optimistic concurrency is another available tool. The Hypertext Transfer Protocol (HTTP) specification describes **If-Match** as preventing accidental overwrites by concurrent clients. (www.rfc-editor.org) A failed precondition can result in **412 Precondition Failed**. (www.rfc-editor.org) This is an optional concurrency pattern for an appropriate HTTP resource, not a universal property of aircraft-swap messages.

Treat omission and clearing as different operations

Interface-specific semantics matter. The AIDX guide says a missing element is not expected to clear an existing value. Jeppesen's documented aircraft-assignment interface describes a different case: omission of Aircraft ID removes the receiving application's previous assignment. An adapter must map the agreed message type explicitly rather than applying a universal "missing means unchanged" rule. (www.iata.org) (www.jeppesen.com)

The same care applies to acknowledgments. AIDX describes notifications rather than commands and provides a response message for acknowledgment. The proposed application acknowledgment adds the particular applied assignment version and consumer outcome required for this operator's closure policy. (www.iata.org)

Packaged Coverage and the Bridge Decision

Begin with the installed product's supported configuration and interfaces. PDC says its FlightTime Optimiser assigns aircraft tails and writes accepted changes back to the plan or standard week. Its FlightOps Solver uses the same engine and schedule data. Those claims justify evaluating existing coverage before adding another assignment store. (www.pdc.com)

Table 2 separates documented product scope from the evidence still needed in an operator's environment. The final row is a procurement option for custom integration, with proposed acceptance obligations rather than vendor-performance claims.

Option	Verified scope or evaluation role	Evidence to request
PDC optimisers	Tail assignment, accepted-change writeback, and shared planning/recovery data are described by the vendor. (www.pdc.com)	Show the installed publication boundary, version mapping, and required-consumer results.

Option	Verified scope or evaluation role	Evidence to request
Lufthansa Systems NetLine	Vendor describes optimization logic extending to automatic tail assignment. (www.lhsystems.com)	Demonstrate the licensed modules and external handoffs against the same swap tests.
IBS Software iFlight	Vendor describes integrated operational and crew recovery plans. (www.ibsplc.com)	Show which consumer results are exposed and how exceptions reconcile.
N-Ops & Crew	Documentation distinguishes a calculated solution from a saved result. (n-raido.support.skywise.com)	Identify the durable save event and its supported downstream interface.
LANDING.AERO, custom integration procurement option	Evaluate only for a defined missing handoff around existing systems.	Require the same ownership record, event contract, test results, and operating documentation.

The comparison does not rank vendors or imply that shared data automatically implements this report's contract. It distinguishes the scope documented publicly from the deployment-specific evidence required to close the decision. Custom integration should remain narrow enough that source authority and existing application responsibilities stay understandable.

FlightAtom's currently retrieved page states that source connections, refresh timing, and implementation scope must be agreed for the deployment. This report therefore does not treat that page as proof of a standardized aircraft-substitution interface. A product name or an older discovery description is insufficient evidence for the exact handoff being evaluated. (www.flightatom.com)

Jeppesen's published overview offers a useful interface-level contrast: it describes a repeating group for equipment changes across multiple legs. The document is historical, so its semantics are an example to compare with the operator's contracted interface, not confirmation of a current licensed capability. (www.jeppesen.com)

Use procurement questions that produce observable evidence:

- **Source access:** can the interface expose committed changes and a versioned current snapshot?
- **Authority:** which source wins when an external system proposes a conflicting assignment?
- **Consumers:** which endpoints provide durable application results, and which expose only receipt?
- **Recovery:** can the deployment replay, reconcile, and reverse without overwriting newer state?
- **Support boundary:** who owns the schema mapping and exception workflow after either side changes?
- **Coverage:** can configuration or a supported vendor interface satisfy the gap before custom work is added?

A bridge is justified when a required handoff remains unproven after this exercise. A new optimizer is a different purchase: this report's proposed bridge distributes committed decisions and accounts for consumer state rather than selecting aircraft rotations.

Data Analysis and Evidence

Quantify the actual delivery boundaries

The available evidence supports protocol limits and documented behavior, rather than a universal aircraft-swap propagation benchmark. Amazon SQS FIFO documents a **5-minute deduplication interval**.

(docs.aws.amazon.com) Azure Service Bus documents a **10-minute default duplicate-history window**, configurable from **20 seconds to 7 days**. (learn.microsoft.com) These are product-specific settings, not interchangeable promises about a consumer's database.

A replay outside a broker's duplicate window may still refer to an event the application already processed. The proposed contract therefore retains application-level processing identity for the operator's chosen reconciliation horizon. Google Pub/Sub also documents that redelivery can include subsequent messages already acknowledged, while retaining acknowledged messages enables replay. That reinforces the need to test business effects separately from transport receipt. (docs.cloud.google.com)

Measure application state, not messages sent

The proposed acceptance metrics are:

- **Version coverage:** required consumer-leg pairs reporting the target version divided by all required pairs.
- **Application delay:** elapsed time between source commitment and each durable application result.
- **Stale-state duration:** time a required consumer remains behind the effective assignment version.
- **Duplicate side effects:** additional business actions caused by repeated delivery of the same event.
- **Reconciliation completeness:** unresolved source-to-consumer version mismatches after recovery.

Hypothetical Example: synthetic calculation. Suppose a test changes **3 legs** and names **6 mandatory consumers** for each leg. The required coverage set has **18 consumer-leg pairs**. If **17** report the target version, coverage is **17/18, or approximately 94.4%**. Under the proposed closure rule, distribution remains incomplete. These are arithmetic test inputs and outputs, not airline performance data or recommended service levels.

Do not average away an unresolved subscriber. RabbitMQ states that publisher confirms are unaware of consumers, and its documentation recognizes application cases requiring explicit peer acknowledgments. A successful broker publication count cannot establish version coverage. (www.rabbitmq.com)

Execute the acceptance matrix

Table 3 defines a proposed minimum test matrix. Its synthetic versions and expectations test contract behavior rather than operational suitability.

Test	Synthetic stimulus	Required observable result
Single-leg swap	Change aircraft A to B for one operating leg.	Stable leg identity; advanced assignment version; required consumer results match.

Test	Synthetic stimulus	Required observable result
Multi-leg swap	Change an affected rotation segment.	Manifest identifies every changed leg; partial application remains visible.
Codeshare	Associate marketing references with the changed operating leg.	Aliases resolve to the operating leg without creating competing assignments.
Late swap	Deliver after its declared effective time.	Preserve business time; record late application and unresolved consumers.
Swap reversal	Commit A to B, then B to A.	Newer version and event identity; retained history; no regression on replay.
Duplicate event	Redeliver the same source and event identifier.	Existing result returned; no additional business side effects.
Out-of-order delivery	Deliver version 19 before version 18.	Current projection remains at 19; stale event is recorded.
Consumer outage	Resume a subscriber after missed changes.	Versioned snapshot or replay restores state and reports remaining mismatches.

The codeshare case reflects a real distinction in AIDX: the guide places codeshare information within the operating flight's data. The other expectations are the proposed operator-specific policy. Passing requires retained event records, source snapshots, and consumer results that an evaluator can inspect, rather than a demonstration that several screens display the same registration momentarily. (www.iata.org)

Implications and Future Directions

An effective integration assessment produces a bounded decision: existing coverage demonstrated, supported configuration required, or a specific missing bridge. AsyncAPI explicitly does not assume a particular architecture or topology, so a documented event contract need not force replacement of working applications or adoption of a particular broker. (www.asyncapi.com)

The proposed implementation sequence is deliberately narrow:

- **Inventory:** identify the committed-assignment source, installed interface profiles, and actual subscribers.
- **Specify:** agree field ownership, version scope, clearing semantics, and required-consumer policy.
- **Observe:** collect source events and consumer results in a test environment before enabling downstream writes.
- **Exercise:** run duplicates, reversals, partial groups, stale events, and outage recovery.
- **Accept:** retain artifacts showing which versions were applied and which exceptions remain.
- **Operate:** maintain mapping ownership, reconciliation access, and versioned documentation.

Historical replay deserves its own design decision. Kafka documents that events are not deleted merely because they were consumed; its retention model provides a transport capability. The operator still needs a policy for whether replay rebuilds a projection, re-emits notifications, or creates a new acknowledgment.

Retention and replay permission should therefore be specified separately. (kafka.apache.org)

Time semantics also need explicit labels. CloudEvents defines its time attribute as the time the occurrence happened; the proposed contract additionally distinguishes assignment applicability, commitment, publication, and consumer observation. Where an AIDX mapping is used, the guide documents Coordinated Universal Time (UTC) timestamps with a trailing Z. Neither convention removes the need to state what each timestamp means. (github.com) (www.iata.org)

Future schema or product changes should trigger targeted contract tests. In particular, validate required fields, null handling, identity mappings, and version comparisons before accepting a changed interface. JSON Schema allows prohibition of additional properties when that is the chosen policy, but compatibility policy should be explicit rather than an accidental consequence of a serializer's defaults. (json-schema.org)

Conclusion

The useful answer to “aircraft tail swap” for an OCC integration assessment is a versioned change to an identified flight leg, accompanied by evidence of what each required consumer applied. The aircraft registration is part of the assignment. It is insufficient by itself to express ownership, commitment, group scope, applicability, or downstream completion.

The proposed contract separates those concerns through stable leg identity, explicit assignment authority, distinct event and schema identities, and a durable application result from each required subscriber. It treats a reversal as a newer committed change and makes duplicate, stale, late, and partial delivery observable. These are proposed acceptance requirements, not operational instructions or guarantees inherited from a messaging standard.

The purchasing decision should follow the test evidence. If the existing rotation or OCC deployment exposes the committed state, propagates it through supported interfaces, and passes the consumer tests, additional integration may be unnecessary. Where a gap remains, specify it narrowly and evaluate configuration, supported interfaces, or custom integration against the same matrix.

The final deliverable should be reviewable: an ownership record, a deployed schema profile, an event contract, source and consumer evidence, and a reconciliation policy. Together, they allow the operator to determine whether everyone saw the same approved assignment version and to identify the exact handoff that still needs work.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. https://www.jeppesen.com/download/CA_docs/bookshelf/Interface%20Control%20Documents/MessageOverviews/OperationsControl/OCMessages.html
2. https://fixm.aero/documents/GUFI%20Requirements%20v2%201_Final.pdf
3. <https://www.iata.org/en/publications/info-data-exchange>
4. <https://www.iata.org/contentassets/cfe998bcf9214859afda9c8bf4ff75c3/aidx-xml-imp-guide-v22.1.pdf>
5. <https://github.com/cloudevents/spec/blob/main/cloudevents/spec.md?plain=1>

6. <https://www.rabbitmq.com/docs/confirmations>
7. <https://www.pdc.com/airline-solution/optimisers>
8. <https://www.lhsystems.com/solutions/operations-control-center/netlineops>
9. <https://www.ibsplc.com/product/airline-operations-solutions/flight>
10. <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/FIFO-queues-exactly-once-processing.html>
11. <https://learn.microsoft.com/en-us/azure/service-bus-messaging/duplicate-detection>
12. <https://xwiki.avinor.no/spaces/ACIDD/pages/243240088/AircraftRegistration>
13. <https://n-raido.support.skywise.com/support/solutions/articles/35000188223-tail-assign-optimizer>
14. <https://xwiki.avinor.no/plugins/viewsource/viewpagesrc.action?pageId=243240822>
15. <https://www.developer.aero/api-catalog/aidx-overview>
16. <https://standards.atlassian.net/wiki/spaces/AIDM/pages/2241003524/IATA%2BData%2BExchange%2BStandards%2B-%2BDownload>
17. <https://n-flightplanning.support.skywise.com/support/solutions/articles/35000196341-flight-info-group-field-descriptions>
18. <https://www.lhsystems.com/solutions/operations-control-center/netline-crew>
19. <https://www.swiss-as.com/modules/planning>
20. <https://www.eurocontrol.int/publication/eurocontrol-specification-airport-collaborative-decision-making-cdm>
21. <https://www.asyncapi.com/docs/reference/specification/v3.1.0>
22. <https://json-schema.org/understanding-json-schema/reference/object>
23. <https://learn.microsoft.com/en-us/azure/architecture/databases/guide/transactional-out-box-cosmos>
24. <https://www.w3.org/TR/trace-context>
25. <https://docs.aws.amazon.com/prescriptive-guidance/latest/cloud-design-patterns/transactional-outbox.html>
26. <https://kafka.apache.org/41/getting-started/introduction>
27. <https://docs.cloud.google.com/pubsub/docs/ordering>
28. <https://www.rfc-editor.org/rfc/rfc9110.html>
29. <https://www.flightatom.com/solutions/airline-operations-control-software.html>

THE PUBLISHER

About LANDING.AERO

LANDING.AERO builds custom software for aviation operations teams at airlines, charter operators and flight departments. It is based in Montreal.

What it builds

Operations tools that reduce manual steps in day-to-day flight operations; integration of existing systems and data sources into one operational picture; workflow automation; real-time operational dashboards; bridges between legacy systems and modern applications; and mobile apps for staff in the field. Each application is designed around the operator's own workflows rather than sold as an off-the-shelf platform.

How it works

Work starts with discovery of the operator's workflows, continues with iterative development alongside the operator's team, and includes integration with the tools already in use and ongoing support.

Its own tools

LANDING.AERO also publishes small tools for aviation professionals, including UTC Alarm, which sets alarms in Zulu time, and NOTAM Notifier, announced for NOTAM change alerts on chosen airports.

Visit

See LANDING.AERO and its projects.

Articles published here explain general practice and cite their primary sources, naming the jurisdiction of any rule they describe. They are not operational, regulatory or legal advice, do not replace an operator's approved procedures or manuals, and requirements change.

Website: <https://www.landing.aero>

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.