# Azure Messaging Technical Specification

**Project**: IoT Gateway – Health, Metrics, Security Warnings, Alerts & Errors

**Goal**: Send device health status, per‑service status & fault reasons, metrics, and security alerts from a small IoT device to Azure **every 5 minutes and on any status change**. Provide resilient offline buffering, duplicate safety, TTLs, and clear topic/subscriber strategy. Decide the best Azure message broker.

## 1. Scope & Assumptions

* Single device class (“gateway”), many tenants possible later.
* Uplink only for routine data (D2C); cloud→device (C2D) reserved for commands (e.g., software update start, config refresh).
* Message paths must work when network is intermittent; device OS offers stable clock within ±60 s.
* No giant binaries on the broker; software packages live in Blob Storage (signed URL). Broker carries **control** only.

## 2. Message Catalog

### 2.1 Health (D2C)

* **When**: every **5 min** and on change (online/partial/offline).
* **Size**: 1–5 KB. **Semantics**: latest-wins.
* **Ordering**: per device helpful. **Criticality**: high, must reach.

### 2.2 Metrics (D2C)

* **When**: every 5 min (batch) and on change if threshold exceeded.
* **Size**: 0.5–50 KB per batch (compressible JSON); **Throughput**: low/medium for a small device footprint.
* **Ordering**: per device preferred but not strictly required.

### 2.3 Service Status + Fault Reason (D2C)

* **When**: included in health payload, plus separate **event on change** (serv1/serv2…)
* **Payload**: service name, protocol statuses (HTTP/Broker/GGG/HHH as applicable), fault code/message.

### 2.4 Security Warnings / Alerts / Errors (D2C)

* **When**: on detection (CVE, tamper, policy drift, auth failures).
* **Priority**: highest; never dropped if buffer not full (preempt metrics if necessary).

### 2.5 Software Update Signals (C2D/D2C)

* **C2D**: notify gateway of update (version, signed URL, SHA‑256, rollout window).
* **D2C**: ack + progress + result (idempotent, small payloads).

## 3. Brokers Considered

| Capability | Azure **Service Bus** | Azure **Event Hubs** | Azure **Event Grid** |
| --- | --- | --- | --- |
| Protocol | AMQP 1.0, HTTPS | AMQP 1.0 (Kafka API), HTTPS SDK | HTTPS/Webhooks (CloudEvents) |
| Delivery | Queues & **Topics**, at‑least‑once, **DLQ**, **deferral**, **sessions**, **transactions**, **scheduled** messages | Append-only partitions; at‑least-once via consumer checkpoints; very high throughput | Push to endpoints with retries; simple filtering |
| Per‑device ordering | ✔ via **Sessions** or PartitionKey | Per‑partition only | ✖ |
| Rich filtering | ✔ SQL filters on subscriptions | Consumer-side | Basic attributes only |
| De‑duplication | ✔ built-in (time window) | Consumer-side | ✖ |
| TTL / Lock management | ✔ | Basic retention per Hub | Limited |
| Best for | **Control, health, security, commands** | Firehose telemetry | Webhook fan‑out |

**Fit**: For a small device sending periodic + on‑change control-ish data, **Azure Service Bus Topics** are the best default. Use Event Hubs only if metrics volume becomes very high.

## 4. Protocol, Auth, and Security

* **Primary wire protocol**: **AMQP 1.0** (efficient, bi‑directional, settlement acks).
* **Fallback**: HTTPS (when AMQP unavailable). Both must be supported by client library.
* **Identity**: Device X.509 cert stored in secure store (TPM/HSM if available).
* **Auth to broker**: Microsoft **Entra ID** client‑credentials (cert) or short‑lived **SAS** minted by a **Custom Auth Broker (CAB)**.
* **Token lifetime**: 15–60 min; rotate seamlessly before expiry; clock skew tolerance ±2 min.
* **Confidentiality**: TLS 1.2+. Optional payload encryption (JWE) for sensitive security events.
* **Authorization**: Least privilege; per‑topic/per‑subscription rights scoped by tenant/device.

## 5. Topic & Subscription Design (Service Bus)

### 5.1 Topic Strategy

**Option A – Single topic with filtering (recommended)** - Topic: d2c.gateway.events - Application properties: type ∈ {health,metrics,serviceStatus,security,sw\_ack}, plus tenantId,deviceId,severity,schemaVer,traceId,dedupeKey. - Subscriptions with SQL filters: - sub\_health: type = 'health' - sub\_metrics: type = 'metrics' - sub\_security: type = 'security' - sub\_serviceStatus: type = 'serviceStatus' - sub\_all: no filter (raw archive/ELT)

**Option B – One topic per feature** - Topics: d2c.health, d2c.metrics, d2c.security, d2c.serviceStatus. - Pros: hard isolation, per‑feature quotas. Cons: more entities to manage, client routing logic.

**Default**: **Option A** (single topic + filters). Add per‑tenant subscriptions if needed for isolation.

### 5.2 Command Channel (C2D)

* Topic: c2d.gateway.commands
* Subscriptions per rollout wave/region: update\_wave\_apac, update\_wave\_emea, etc.

### 5.3 Number of Subscribers (initial)

* sub\_health → 1–2 consumers (realtime status + dashboard cache)
* sub\_metrics → 1–n (metrics processor; optional lake ingestion)
* sub\_security → 1–2 (SOAR/SIEM pipeline)
* sub\_all → 1 (raw archiver)
* Scale out via competing consumers per subscription.

## 6. Message Semantics & Policies

### 6.1 Idempotency & De‑duplication

* **Headers**: messageId (UUID), deviceId, ts, seq.
* **Dedup window** (Service Bus): 10 minutes (configurable). Device must **retry with same messageId** until broker settlement is received.
* Server handlers must be **idempotent** on (deviceId,ts,seq|messageId).

### 6.2 Ordering

* **Sessions** with SessionId = deviceId for health/security/serviceStatus to preserve per‑device order.
* Metrics may skip sessions; use PartitionKey = deviceId for locality.

### 6.3 TTL

* Health: **1 hour** (stale health is useless)
* Service status: **4 hours** (for recent history)
* Security: **7 days** (operations may need replay)
* Metrics: **24–72 hours** (depends on analytics SLA)
* SW acks: **24 hours**

### 6.4 Delivery ACK Strategy

* Device uses AMQP **settlement ack** (receiver acceptance) as success. Do **not** block waiting for a business response.
* Retries until ack or local policy expires. Exponential backoff with jitter.

### 6.5 Batching

* Metrics batched by **time (5 min)** or size (e.g., 200 records/250 KB), compressed (gzip/deflate) when beneficial.

## 7. Offline / Connectivity Failure Behavior

* **Store‑and‑forward queue** on device (disk‑backed). Capacity: target 72h at expected rate.
* **Priority lanes**: Security > Health/ServiceStatus > Metrics. If storage full, drop oldest **metrics** first; never drop security.
* **Retry policy**: exponential backoff (e.g., 1s → 60s cap) with **jitter**. Preserve messageId across retries.
* **Recovery**: upon reconnect, flush in order (by timestamp) per lane. Health should publish **current** snapshot first, followed by backlog (mark as late:true).
* **Clock drift handling**: include ts (ISO 8601) and monotonic seq; server accepts up to ±10 min skew.

## 8. Error Handling & DLQ

* Use **dead‑letter queues (DLQ)** for poison messages (exceed max delivery count or deserialization error).
* DLQ monitored by ops; automated reprocess job checks and republishes after fix.
* Include errorCategory and parseError details when NACKing internally.

## 9. Payload Schemas (examples)

**Health**

{  
 "schemaVer":"1.0",  
 "deviceId":"gw-001",  
 "ts":"2025-10-22T12:00:00Z",  
 "status":"online",   
 "summary":"all services healthy",  
 "services":[  
 {"name":"serv1","status":"green","fault":null},  
 {"name":"serv2","status":"partial","fault":{"code":"BROKER\_TIMEOUT","msg":"Broker not reachable"}}  
 ]  
}

**Metrics (batch)**

{  
 "schemaVer":"1.0",  
 "deviceId":"gw-001",  
 "ts":"2025-10-22T12:00:00Z",  
 "metrics":[  
 {"name":"cpu","val":0.42,"u":"ratio","t":173, "tags":{"core":"total"}},  
 {"name":"mem\_mb","val":512,"u":"MB","t":173}  
 ]  
}

**Service Status Change**

{  
 "schemaVer":"1.0",  
 "deviceId":"gw-001",  
 "ts":"2025-10-22T12:03:12Z",  
 "service":"serv1",  
 "protocols":[  
 {"name":"HTTP","status":"Up"},  
 {"name":"Broker","status":"Down","reason":"timeout"}  
 ],  
 "overall":"partial"  
}

**Security Alert**

{  
 "schemaVer":"1.0",  
 "deviceId":"gw-001",  
 "ts":"2025-10-22T12:04:55Z",  
 "event":"cve\_detected",  
 "cve":["CVE-2025-12345"],  
 "severity":"high",  
 "details":"openssl 1.1.1x vulnerable"  
}

**Software Update (C2D) → Ack (D2C)**

{  
 "schemaVer":"1.0",  
 "commandId":"cmd-7f2a",  
 "deviceId":"gw-001",  
 "state":"in\_progress",   
 "progressPct":42,  
 "message":"download 420/1000 MB"  
}

## 10. Cloud Consumers & Scaling

* Each subscription processed by a **consumer group** (competing consumers) for scale-out.
* Consumers are **idempotent**; de‑duplicate on (deviceId,ts,seq|messageId).
* Expose **prometheus/AppInsights** metrics: success rate, retry counts, DLQ depth, processing lag, handler latency P50/P95/P99.

## 11. Monitoring & SLOs

* **Device→Broker P95**: Health/security ≤ 5 s; metrics ≤ 10 s.
* Alerts on: auth failures, throttling, DLQ growth, session lock lost, partition/key skew.
* Trace using traceId and correlationId (commands ↔ acks).

## 12. DR & Compliance

* **Namespace geo-pair** with **alias** for failover (Service Bus Premium recommended for mission‑critical).
* Optional active/active with multi‑region write if tenant split is region‑scoped.
* **Data residency** tags; customer-managed keys if required.

## 13. Answering Your Design Questions (explicit)

1. **What happens when connectivity fails?**  
   Store‑and‑forward to disk with priority queues; exponential backoff; retry with same messageId; flush in order on reconnect; publish a **fresh health snapshot first**.
2. **Message recovery process?**  
   DLQ for poison; reprocess job; idempotent handlers; replay safe due to dedupe keys.
3. **Message duplicates?**  
   Use Service Bus **duplicate detection** + idempotent consumers. Device retries reuse same messageId.
4. **Message TTL?**  
   Health 1h; Service Status 4h; Security 7d; Metrics 24–72h; SW acks 24h (adjust per SLA).
5. **How many subscribers?**  
   Start with 4 logical subscriptions (health, metrics, security, all). Scale consumers horizontally; add per‑tenant subs if needed.
6. **One topic per feature vs one topic for all?**  
   Default to **one topic + filtered subscriptions**. Switch to per‑feature topics if isolation/quotas/audits demand it.
7. **Should the device wait for a cloud response?**  
   No. Treat AMQP settlement as delivery confirmation. Only commands require app‑level replies (acks/progress), sent asynchronously.
8. **Should the broker support both HTTP and AMQP?**  
   Yes—**AMQP** primary for efficiency & features; **HTTP** fallback for constrained networks. Service Bus supports both.

## 14. Final Decision & Variants

* **Primary Recommendation**: **Azure Service Bus Topics (Premium)** for all D2C control messages (health, metrics (low/med), security, service status) and C2D commands.
* **Variant A (High‑volume metrics)**: keep health/security/serviceStatus on Service Bus; route **metrics** to **Event Hubs** (partition key = deviceId) with a separate ingestion pipeline.
* **Variant B (Webhook fan‑out)**: raise selected security events from Service Bus to **Event Grid** to notify external systems.

## 15. Minimal Entity Blueprint (Service Bus)

**Topics** - d2c.gateway.events (Requires Sessions=true for health/security/serviceStatus; DedupWindow=10m; MaxDeliveryCount=10) - c2d.gateway.commands (Requires Sessions=true; ScheduledMessages=enabled)

**Subscriptions & Filters** - sub\_health: type = 'health' - sub\_metrics: type = 'metrics' - sub\_serviceStatus: type = 'serviceStatus' - sub\_security: type = 'security' - sub\_all: no filter (archival)

**Headers (Application Properties)** - type, tenantId, deviceId, schemaVer, severity, traceId, correlationId, seq, dedupeKey (== messageId by default)

## 16. Sequence (PlantUML)

@startuml  
actor Device  
participant "Azure Service Bus" as SB  
participant "Metrics Processor" as MP  
participant "Status Service" as SS  
participant "Security Handler" as SH  
  
== Periodic & On-Change Send ==  
Device -> SB : Send D2C (health/metrics/security) [AMQP]  
SB --> Device : Ack (settlement)  
  
== Parallel Consumption ==  
SB ->> SS : sub\_health/serviceStatus  
SB ->> MP : sub\_metrics  
SB ->> SH : sub\_security  
  
== Connectivity Loss ==  
Device -> Device : Store-and-forward to disk  
Device -> Device : Retry with backoff, same messageId  
  
== Recovery ==  
Device -> SB : Flush backlog (ordered by deviceId session)  
SB ->> Consumers : Deliver (dedupe, DLQ on poison)  
@enduml

## 17. Non-Functional Requirements

* **Latency**: P95 ≤ 5s (health/security), ≤10s (metrics).
* **Availability**: 99.9%+ (Premium tier, zone redundant where available).
* **Cost controls**: batch metrics; compress; avoid oversized messages; right-size tiers.
* **Telemetry**: device + broker + consumer tracing with traceId.

## 18. Next Steps

1. Provision Service Bus (Premium) with the topics/subscriptions above.
2. Implement device AMQP client with store‑and‑forward, retries, and headers.
3. Stand up consumers (Functions/Workers) per subscription, idempotent with DLQ workflows.
4. Add dashboards/alerts for DLQ, throttle, auth failures, lag.
5. Decide whether metrics volume needs Variant A (Event Hubs).

*This spec is written to be implementation-ready; we can append Bicep/Terraform and AMQP (Java) code snippets on request.*