Architecture — Event Bus Map

LumisCare Event Bus — Current State Map

Date: 2026-04-04 **Author:** Martin Kleppmann (Distributed Systems Audit) **Method:** Static analysis of all publisher and listener classes in backend/services/ **Source of truth:** Actual Java source files, not design documents

---

Architecture

Azure Service Bus Premium tier, topics/subscriptions model. Each service owns its own topic and publishes events to it. Consumers subscribe to specific topics with SQL filters on `event_type`.

The design document (`Service-Bus-Subscriptions-Design.md`) specifies three topics: `visit-events`, `schedule-events`, `notification-events`. The actual implementation has diverged — publishers use five distinct topic names across six services, and the notification-service listener is wired to a different topic (`domain-events`) than what the visit and scheduling publishers write to.

Event Flow Diagram

```
graph LR
  subgraph WIRED["FULLY WIRED"]
    HR["hr-service<br/>(HrEventPublisher)"]
    SCHED["scheduling-service<br/>(AppointmentEventPublisher)"]
    SCHED_AVAIL["scheduling-service<br/>(HrAvailabilityUpdatedEventHandler)"]
    NOTIF_PUB["notification-service<br/>(NotificationEventPublisher)"]
  end

  subgraph PARTIAL["PARTIALLY WIRED — publisher exists, consumer absent or mis-wired"]
    VISITS["visits-service<br/>(VisitEventPublisher)"]
```

```
    INCIDENTS["incidents-service<br/>(EventPublisher)"]
    SAFETY["safety-service<br/>(SafetyEventPublisher)"]
    NOTIF_LISTEN["notification-service<br/>(ServiceBusEventListener)"]
    SCHED_CAREPLAN["scheduling-service<br/>(CarePlanPublishedEventHandler)"]
  end

  subgraph BROKEN["BROKEN — no publisher class exists"]
    ASSESS["assessment-service"]
    CAREPLAN["careplan-service"]
    FINANCE["finance-service<br/>(OutboxEvent entity only)"]
  end

  HR --
>|"employee.created/updated<br/>availability.updated<br/>leave.approved<br/>certification.expi
ring<br/>topic: domain-events"| SCHED_AVAIL
  HR -->|"leave.approved<br/>topic: domain-events"| NOTIF_LISTEN
  VISITS -->|"visit.started/completed/missed<br/>visit.task.completed<br/>topic: visits-
events"| NOTIF_LISTEN
  VISITS -->|"visit.completed<br/>topic: visits-events"| FINANCE_STUB["finance-service<br/>(NO
listener class)"]
  SCHED -->|"schedule.published/updated<br/>appointment.created/updated/cancelled<br/>topic:
scheduling-events"| NOTIF_LISTEN
  NOTIF_PUB -->|"notification.sent/failed/acknowledged<br/>topic: notification-events"|
ANALYTICS_STUB["analytics-service<br/>(NOT IMPLEMENTED)"]
  INCIDENTS -->|"incident.created<br/>topic: incidents-events"| NOTIF_LISTEN
  SAFETY -->|"lwp.escalation.*<br/>topic: safety-events"| NOTIF_LISTEN
  CAREPLAN_STUB["careplan-service<br/>(NO publisher)"] -.->|"careplan.published — MISSING"|
SCHED_CAREPLAN
  ASSESS -.->|"assessment.completed — MISSING"| CAREPLAN_STUB

  style WIRED fill:#1a4731,color:#fff
  style PARTIAL fill:#713f12,color:#fff
  style BROKEN fill:#7f1d1d,color:#fff
```

Critical Architectural Finding: Topic Name Mismatch

The notification-service `ServiceBusEventListener` subscribes to topic `domain-events` (default configured at `ServiceBusConfig.java:29`):

```
azure.servicebus.topic-name:domain-events
```

The visits-service `VisitEventPublisher` publishes to topic `visits-events` (default configured at `VisitEventPublisher.java:45`):

```
azure.servicebus.visits-topic-name:visits-events
```

The scheduling-service `AppointmentEventPublisher` publishes to topic `scheduling-events` (default configured at `AppointmentEventPublisher.java:45`):

```
azure.servicebus.scheduling-topic-name:scheduling-events
```

The HR-service `HrEventPublisher` publishes to topic `domain-events` (default configured at `HrEventPublisher.java:40`):

```
azure.servicebus.topic-name:domain-events
```

This means: **the notification-service listener will receive HR events but will silently miss all visit and schedule events at runtime unless the application properties are overridden in deployment configuration.** There are no application.properties or application.yml files with explicit topic overrides found in the source tree — only Spring `@Value` defaults. This must be validated against the actual Azure Container Apps environment variables before treating any visit-to-notification flow as working.

Event-by-Event Status

Event	Topic (default)	Producer Service	Publisher Class	Consumer Service	Listener Class	Status
`employee.created`	`domain-events`	hr-service	`HrEventPublisher`	scheduling-service	`HrAvailabilityUpdatedEventHandler`	WIRED — both sides present
`employee.updated`	`domain-events`	hr-service	`HrEventPublisher`	scheduling-service	`HrAvailabilityUpdatedEventHandler`	WIRED — both sides present

Event	Topic (default)	Producer Service	Publisher Class	Consumer Service	Listener Class	Status
`availability.updated`	`domain-events`	hr-service	`HrEventPublisher` (via `AvailabilityPreferencesService`)	scheduling-service	`HrAvailabilityUpdatedEventHandler` (listens for `hr.availability.updated`)	WIRED — subject used in code is `hr.availability.updated`, design doc says `availability.updated` — schema mismatch risk
`leave.approved`	`domain-events`	hr-service	`HrEventPublisher` (via `LeaveManagementService`)	notification-service	`ServiceBusEventListener` (case `leave.approved`)	WIRED — both sides on same topic
`certification.expiring`	`domain-events`	hr-service	`HrEventPublisher` (via `CertificationExpiryScheduler`)	notification-service	`ServiceBusEventListener` (case `certification.expiring`)	WIRED — both sides on same topic
`visit.started`	`visits-events`	visits-service	`VisitEventPublisher.publishVisitStarted()`	notification-service	`ServiceBusEventListener` (case `visit.checkedin` — event name mismatch)	BROKEN — publisher emits `visit.started`, listener handles `visit.checkedin`; different names
`visit.completed`	`visits-events`	visits-service	`VisitEventPublisher.publishVisitCompleted()`	notification-service	`ServiceBusEventListener` (case `visit.completed`)	TOPIC MISMATCH — publisher on `visits-events`, listener on `domain-events`
`visit.completed`	`visits-events`	visits-service	`VisitEventPublisher.publishVisitCompleted()`	finance-service	No listener class found	NOT WIRED — finance has no Service Bus consumer
`visit.missed`	`visits-events`	visits-service	`VisitEventPublisher.publishVisitMissed()`	notification-service	No explicit case in listener	NOT HANDLED — listener has no `visit.missed` case
`visit.task.completed`	`visits-events`	visits-service	`VisitEventPublisher.publishTaskCompleted()`	notification-service	No explicit case in listener	NOT HANDLED

Event	Topic (default)	Producer Service	Publisher Class	Consumer Service	Listener Class	Status
`checkin.timeout`	`visits-events`	visits-service	NOT FOUND in `VisitEventPublisher` methods	notification-service	`ServiceBusEventListener` (case `checkin.timeout`)	PUBLISHER ABSENT — listener expects this event but publisher has no `publishCheckinTimeout()` method
`assistance.requested`	`visits-events`	visits-service	NOT FOUND in `VisitEventPublisher` methods	notification-service	`ServiceBusEventListener` (case `assistance.requested`)	PUBLISHER ABSENT — listener expects this event but publisher has no method
`task.flagged`	`visits-events`	visits-service	NOT FOUND in `VisitEventPublisher` methods	notification-service	`ServiceBusEventListener` (case `task.flagged`)	PUBLISHER ABSENT — listener expects this but only `visit.task.completed` is published
`redflag.detected`	unknown	unknown	NOT FOUND	notification-service	`ServiceBusEventListener` (case `redflag.detected`)	PUBLISHER ABSENT — no publisher produces this event
`visit.reassigned`	unknown	unknown	NOT FOUND	notification-service	`ServiceBusEventListener` (case `visit.reassigned`)	PUBLISHER ABSENT
`schedule.published`	`scheduling-events`	scheduling-service	`AppointmentEventPublisher.publishSchedulePublished()`	notification-service	`ServiceBusEventListener` (case `schedule.published`)	TOPIC MISMATCH — publisher on `scheduling-events`, listener on `domain-events`
`schedule.updated`	`scheduling-events`	scheduling-service	`AppointmentEventPublisher` (no explicit `publishScheduleUpdated` found — has `publishScheduleApproved`)	notification-service	`ServiceBusEventListener` (case not found)	PARTIAL — method naming diverges from design doc

Event	Topic (default)	Producer Service	Publisher Class	Consumer Service	Listener Class	Status
`careplan.published`	`careplan-events`	careplan-service	NO PUBLISHER CLASS	scheduling-service	`CarePlanPublishedEventHandler` (subscribes to `careplan-events`)	BROKEN — listener fully implemented, publisher does not exist
`careplan.updated`	unknown	careplan-service	NO PUBLISHER CLASS	notification-service	`ServiceBusEventListener` (case `careplan.updated`)	BROKEN — no publisher
`assessment.completed`	unknown	assessment-service	NO PUBLISHER CLASS	careplan-service	NO LISTENER CLASS	NOT WIRED — entire chain absent
`incident.created`	`incidents-events`	incidents-service	`EventPublisher`	notification-service	`ServiceBusEventListener` — no `incident.created` case found in routing switch	TOPIC MISMATCH + case absent — publisher on `incidents-events`, listener on `domain-events`
`lwp.escalation.level1/2/3/4`	`safety-events`	safety-service	`SafetyEventPublisher.publishLWPEscalation()`	notification-service	`ServiceBusEventListener` (cases `lwp.escalation.level1` through `level4`)	TOPIC MISMATCH — publisher on `safety-events`, listener on `domain-events`
`invoice.created`	none	finance-service	NO PUBLISHER — only `OutboxEvent` entity + repository	notification-service	`ServiceBusEventListener` (no case found)	NOT WIRED — outbox pattern incomplete, no dispatcher daemon
`invoice.paid`	none	finance-service	NO PUBLISHER	analytics-service	NOT IMPLEMENTED	NOT WIRED
`timesheet.created`	none	finance-service	NO PUBLISHER — `TimesheetBatchJobProcessor` writes `OutboxEvent` but no Service Bus dispatch	notification-service	`ServiceBusEventListener` (no case found)	NOT WIRED — outbox records written to DB but no relay to Service Bus
`notification.sent`	`notification-events`	notification-service	`NotificationEventPublisher.publishNotificationSent()`	analytics-service	NOT IMPLEMENTED	PUBLISHER WIRED, consumer service does not exist

Event	Topic (default)	Producer Service	Publisher Class	Consumer Service	Listener Class	Status
`notification.failed`	`notification-events`	notification-service	`NotificationEventPublisher.publishNotificationFailed()`	analytics-service	NOT IMPLEMENTED	PUBLISHER WIRED, consumer service does not exist
`daily.summary`	unknown	unknown	NOT FOUND	notification-service	`ServiceBusEventListener` (case `daily.summary`)	PUBLISHER ABSENT

Broken Chains

Chain 1: Assessment to Care Plan AI Generation (Clinical Safety Risk)

The primary clinical workflow trigger is broken at its first link.

- `assessment-service` has no event publisher class anywhere in its source tree. When an assessment is completed, no `assessment.completed` event is emitted.
- `careplan-service` has no event listener class. Even if an event were emitted, nothing would receive it.
- Both sides of the Assessment → CarePlan trigger are absent.

Where it breaks: `assessment-service/src/main/java/` — no `*Publisher*.java` file exists in this directory.

Chain 2: Care Plan to Scheduling (Scheduling Cannot Start Automatically)

The `CarePlanPublishedEventHandler` in scheduling-service is fully implemented (300+ lines, idempotency logic, recurring schedule creation). It is waiting for a `careplan.published` event on topic `careplan-events`. That event is never published.

- `careplan-service` has no publisher class. The service can mark a care plan as published in the database (`CarePlan.publishedAt`, `CarePlan.publishedBy` fields exist) but does not emit the event.

Where it breaks: `careplan-service/src/main/java/` — no `*Publisher*.java` file exists. The handler waiting for the event is at `scheduling-service/src/main/java/com/lumiscare/scheduling/event/CarePlanPublishedEventHandler.java:101`.

Chain 3: Visit Check-in to Notification (Event Name Mismatch)

The design document specifies `visit.checkedin`. The `VisitEventPublisher` publishes `visit.started` (set at `VisitEventPublisher.java:127`). The `ServiceBusEventListener` routes on `visit.checkedin` (at `ServiceBusEventListener.java:207`). These names do not match. Even if the topic mismatch were fixed, this chain would still silently fail.

Where it breaks:

- Publisher sets name: `VisitEventPublisher.java:127` — `"visit.started"`
- Listener routes on: `ServiceBusEventListener.java:207` — `case "visit.checkedin"`

Chain 4: Visit Completed to Finance (Finance Cannot Generate Invoice Line Items)

`VisitEventPublisher.publishVisitCompleted()` publishes to topic `visits-events`. The finance-service has no `ServiceBusProcessorClient`, no listener class, and no subscription configuration. The billing data embedded in the `visit.completed` payload (rates, task categories, mileage) is never consumed. Invoice line item generation is not automated.

Where it breaks: `finance-service/src/main/java/` — no Service Bus consumer class of any kind. Finance only writes `OutboxEvent` records to its own database for the timesheet batch job, which itself has no Service Bus dispatcher.

Chain 5: Critical Safety Alerts Silently Dropped (Topic Mismatch)

`SafetyEventPublisher` publishes to topic `safety-events`. `ServiceBusEventListener` in notification-service subscribes to topic `domain-events`. The LWP escalation events (`lwp.escalation.level1` through `level4`) will never reach the notification service unless an application property override is set in deployment. For lone worker protection, this is a CQC compliance risk.

Where it breaks: `SafetyEventPublisher.java:39` (topic: `safety-events`) vs `ServiceBusConfig.java:29` (subscribed to: `domain-events`).

Chain 6: Visit Events Unreachable by Notification Service (Topic Mismatch)

`VisitEventPublisher` publishes to `visits-events`. `AppointmentEventPublisher` publishes to `scheduling-events`. `ServiceBusEventListener` subscribes to `domain-events`. Unless deployment environment variables override these defaults, all visit and schedule events are invisible to the notification service. This silences checkin.timeout, task.flagged, assistance.requested, and schedule.published notifications.

Where it breaks: Topic default misalignment across three services — see Critical Architectural Finding section above.

Chain 7: Events Listened For But Never Published

The `ServiceBusEventListener` routing switch includes cases for events that have no publisher anywhere in the codebase:

Event	Missing Publisher	Listener Location
`checkin.timeout`	No publisher method in `VisitEventPublisher`	`ServiceBusEventListener.java:207`
`assistance.requested`	No publisher method in `VisitEventPublisher`	`ServiceBusEventListener.java:211`
`redflag.detected`	No publisher anywhere in any service	`ServiceBusEventListener.java:215`
`visit.reassigned`	No publisher anywhere in any service	`ServiceBusEventListener.java:220`
`task.flagged`	No publisher method in `VisitEventPublisher` (publishes `visit.task.completed` instead)	`ServiceBusEventListener.java:224`
`daily.summary`	No publisher anywhere in any service	`ServiceBusEventListener.java:249`

Chain 8: Notification Outbound Events Have No Consumers

`NotificationEventPublisher` publishes `notification.sent`, `notification.failed`, and `notification.acknowledged` to topic `notification-events`. The `analytics-service` and `audit-service` that should consume these are not implemented as standalone services. The events will be published to a topic with no subscribers, age to TTL, and be discarded.

Fix Priority

Ordered by clinical and operational impact.

Priority 1 — Fix Topic Name Coherence (All Services)

Impact: Unblocks all visit and schedule notifications in one config change.

The notification-service `ServiceBusConfig` must subscribe to multiple topics, or all publishers must align on a shared topic name. The least-invasive fix is to add per-topic subscription processors in `ServiceBusConfig.java` for `visits-events` and `scheduling-events` in addition to `domain-events`. Alternatively, if the design intent is a single `domain-events` topic, all publishers must update their `@Value` defaults to `domain-events`.

This is a configuration and wiring change, not a logic change. No new business logic required.

Files to change:

- `notification-service/src/main/java/com/lumiscare/notification/config/ServiceBusConfig.java`
- OR all publisher `@Value` defaults (4 publisher files)

Priority 2 — Fix visit.started / visit.checkedin Event Name

Impact: Allows check-in notifications to reach care managers.

Either rename the event type in `VisitEventPublisher.java:127` from `"visit.started"` to `"visit.checkedin"`, or update the listener case to `"visit.started"`. The design document and listener both say `visit.checkedin` — the publisher is the outlier.

File to change: `visits-service/src/main/java/com/lumiscare/visits/event/VisitEventPublisher.java:127`

Priority 3 — Add Missing Publisher Methods to VisitEventPublisher

Impact: Enables checkin.timeout (clinical safety), task.flagged (care quality), assistance.requested (lone worker safety).

`VisitEventPublisher` needs three new publish methods. The listener routing and notification handlers for these events already exist. This is purely adding publisher-side methods and calling them from the appropriate visit execution service layer.

Events to add: `checkin.timeout`, `task.flagged`, `assistance.requested` **File to change:** `visits-service/src/main/java/com/lumiscare/visits/event/VisitEventPublisher.java`

Priority 4 — Add careplan.published Publisher to careplan-service

Impact: Allows scheduling to automatically create recurring visits when a care plan is approved. This is the central automation trigger for care delivery.

The `CarePlanPublishedEventHandler` in scheduling-service is complete and production-ready. Only the publisher side is missing. Add a publisher class to careplan-service that emits `careplan.published` on topic `careplan-events` when a care plan transitions to published status (the state transition already exists in `CarePlanMapper.java`).

Files to create/change:

- Create: `careplan-service/src/main/java/com/lumiscare/event/CarePlanEventPublisher.java`
- Wire it into the publish endpoint in the careplan service controller

Priority 5 — Add assessment.completed Publisher to assessment-service

Impact: Triggers automatic AI care plan generation after assessment completion.

Create an event publisher in assessment-service and publish `assessment.completed` when an assessment reaches a completed status. The careplan-service needs a corresponding listener (this does not yet exist either — both sides must be built).

Files to create:

- `assessment-service/src/main/java/.../event/AssessmentEventPublisher.java`
- `careplan-service/src/main/java/.../event/AssessmentCompletedEventListener.java`

Priority 6 — Add Finance Service Bus Consumer for visit.completed

Impact: Automates invoice line item generation from completed visits.

Add a `ServiceBusProcessorClient` to finance-service that subscribes to the `visits-events` topic with filter `event_type = 'visit.completed'`. The billing data payload already contains all required fields (`billing_data` section with visit_type, care_types, day_type, travel_time_minutes, mileage_miles).

Files to create:

- `finance-service/src/main/java/.../event/VisitCompletedEventListener.java`
- Wire to existing rate calculation and invoice generation logic

Priority 7 — Complete the Finance Outbox Dispatcher

Impact: Enables `timesheet.created` and `invoice.created` notifications.

`TimesheetBatchJobProcessor` writes `OutboxEvent` records to the database but the comment in the code (`TimesheetBatchJobProcessor.java:35`) refers to an "external outbox-processor daemon" that does not exist. Either implement the outbox polling scheduler within the finance-service or replace the outbox pattern with a direct `ServiceBusSenderClient` call inside the existing transaction.

Files to change:

- `finance-service/src/main/java/com/lumiscare/icon/finance/service/TimesheetBatchJobProcessor.java`
- Optionally create a dedicated outbox publisher component

Priority 8 — Resolve hr.availability.updated vs availability.updated Event Name

Impact: Prevents silent message drops when carer availability changes.

`HrAvailabilityUpdatedEventHandler` expects message subject `hr.availability.updated` (at `HrAvailabilityUpdatedEventHandler.java:94`). `HrEventPublisher` publishes with event type `availability.updated` (called at `AvailabilityPreferencesService.java:161`). This is a subject field mismatch. The handler will silently complete all messages that do not match its expected subject, meaning availability changes will not trigger schedule recomputation.

Files to change: Align the event type string in `AvailabilityPreferencesService.java:161` to `hr.availability.updated`, or update the handler constant at `HrAvailabilityUpdatedEventHandler.java:94`.

Summary Counts

Category	Count
Fully wired event chains (publisher + consumer on matching topic)	3 (employee.created, employee.updated, leave.approved/certification.expiring via domain-events)
Partially wired (publisher exists, consumer absent or topic mismatch)	8
Events listener handles but no publisher produces	6
Events completely absent (no publisher, no consumer)	3 (assessment.completed, invoice.created, invoice.paid)
Publisher classes that exist	6 (HrEventPublisher, VisitEventPublisher, AppointmentEventPublisher, NotificationEventPublisher, SafetyEventPublisher, incidents EventPublisher)
Publisher classes missing	2 (assessment-service, careplan-service)
Services with no Service Bus integration at all	3 (assessment-service, careplan-service, finance-service consumer side)
Topic name mismatches between publisher default and notification-service subscription	3 (visits-events, scheduling-events, incidents-events, safety-events vs domain-events)

The HR to Notification chain (leave.approved, certification.expiring) is the only end-to-end path that works without deployment configuration overrides. Everything that involves visit execution events, schedule events, or safety escalations depends on either a topic name fix or missing publisher code.

Revision #2
Created 2026-04-05 21:40:49 UTC by John
Updated 2026-05-31 20:05:33 UTC by John