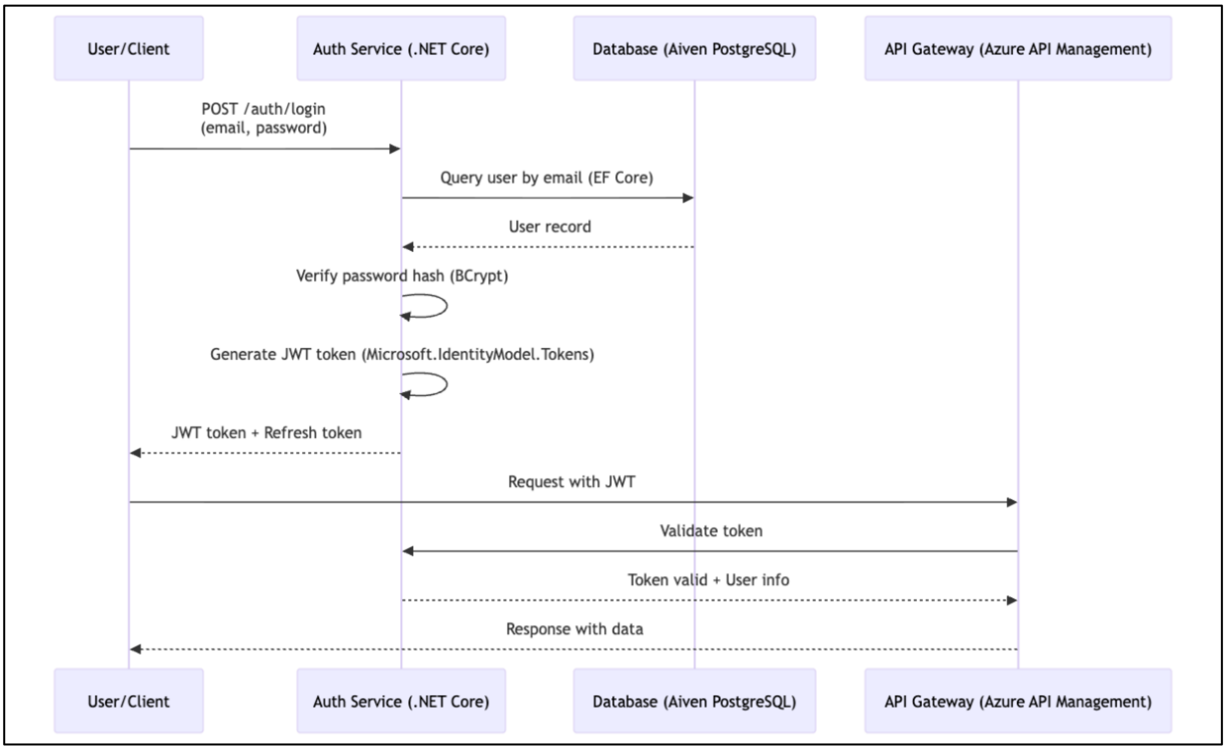
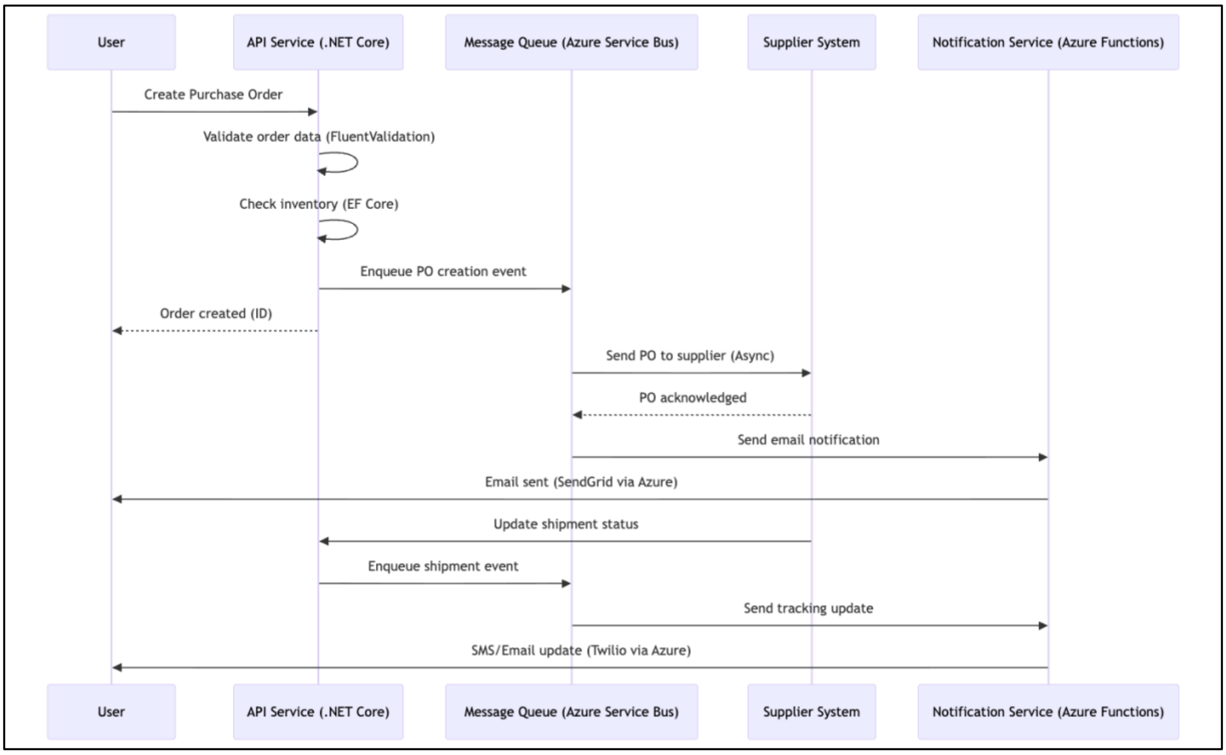
IMSOP - API Flows & Sequences

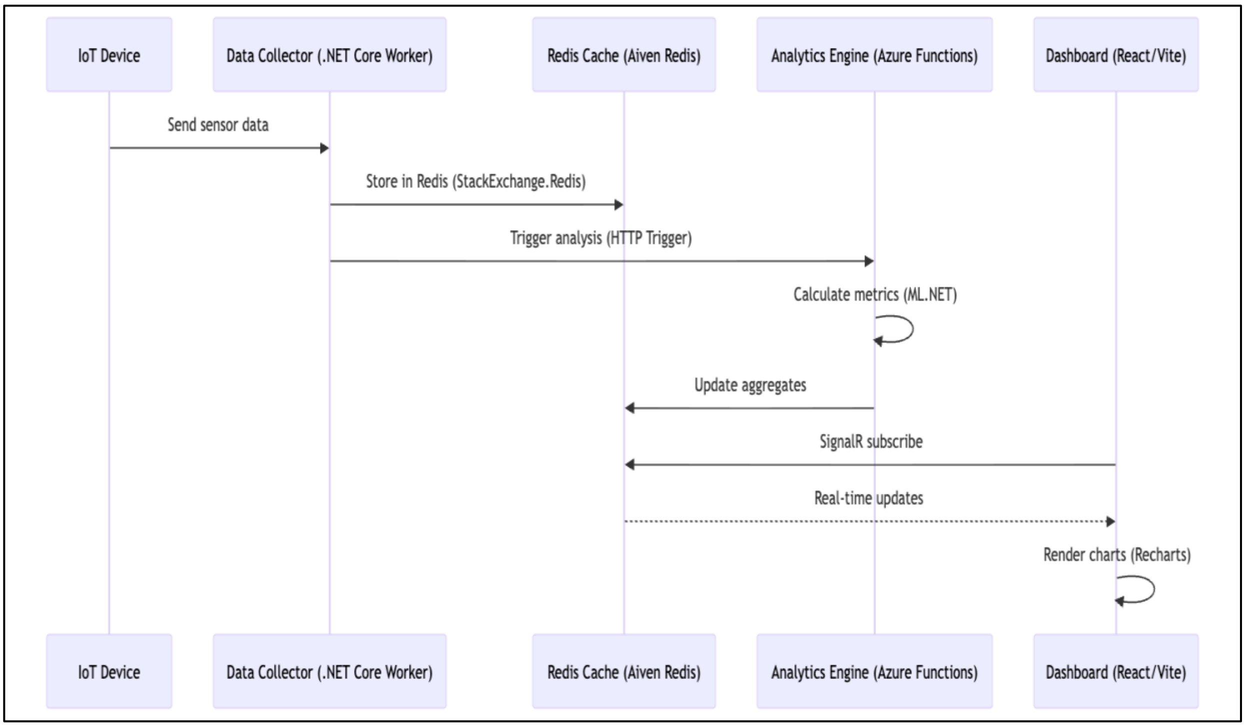
Authentication Flow


User/Client
Auth Service (.NET Core)
Database (Aiven PostgreSQL)
API Gateway (Azure API Management)
POST /auth/login
(email, password)
Query user by email (EF Core)
User record
Verify password hash (BCrypt)
Generate JWT token (Microsoft.IdentityModel.Tokens)
JWT token + Refresh token
Request with JWT
Validate token
Token valid + User info
Response with data
User/Client
Auth Service (.NET Core)
Database (Aiven PostgreSQL)
API Gateway (Azure API Management)

Supply Chain Order Flow

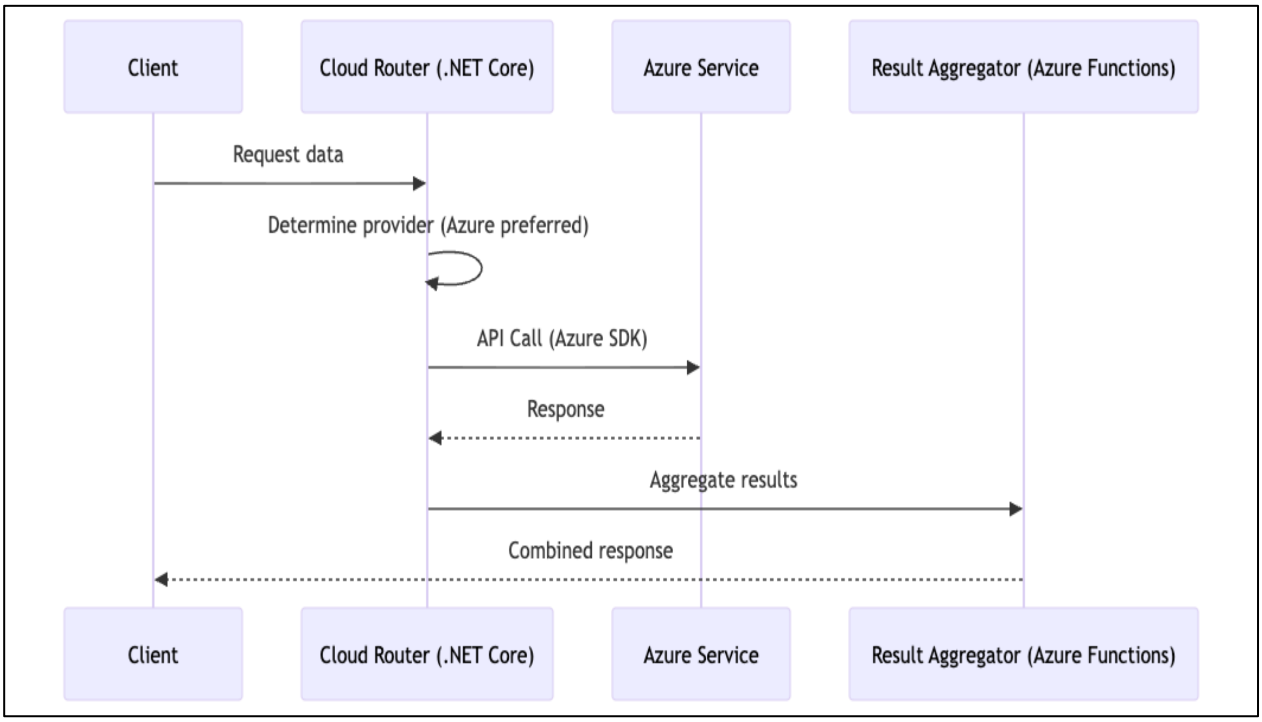

User
API Service (.NET Core)
Message Queue (Azure Service Bus)
Supplier System
Notification Service (Azure Functions)
Create Purchase Order
Validate order data (FluentValidation)
Check inventory (EF Core)
Enqueue PO creation event
Order created (ID)
Send PO to supplier (Async)
PO acknowledged
Send email notification
Email sent (SendGrid via Azure)
Update shipment status
Enqueue shipment event
Send tracking update
SMS/Email update (Twilio via Azure)
User
API Service (.NET Core)
Message Queue (Azure Service Bus)
Supplier System
Notification Service (Azure Functions)

Real-time Analytics Flow

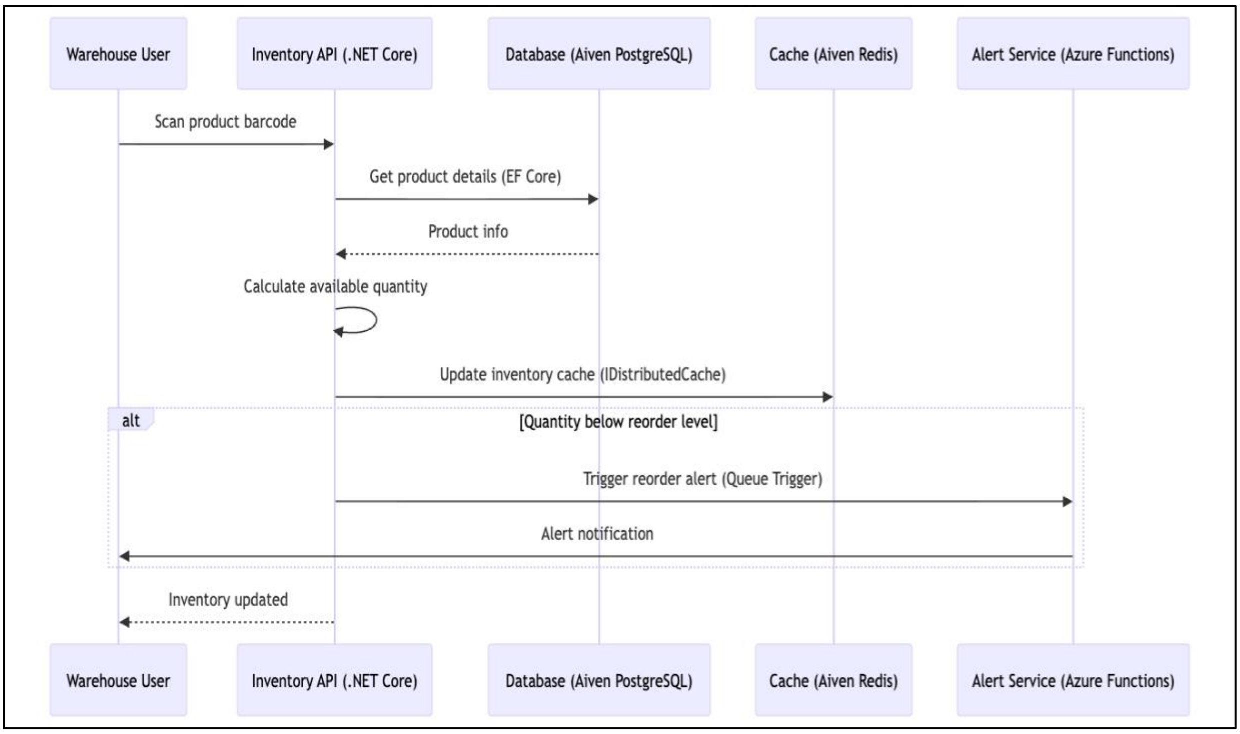

IoT Device
Data Collector (.NET Core Worker)
Redis Cache (Aiven Redis)
Analytics Engine (Azure Functions)
Dashboard (React/Vite)
Send sensor data
Store in Redis (StackExchange.Redis)
Trigger analysis (HTTP Trigger)
Calculate metrics (ML.NET)
Update aggregates
SignalR subscribe
Real-time updates
Render charts (Recharts)
IoT Device
Data Collector (.NET Core Worker)
Redis Cache (Aiven Redis)
Analytics Engine (Azure Functions)
Dashboard (React/Vite)

Multi-Cloud Integration Flow

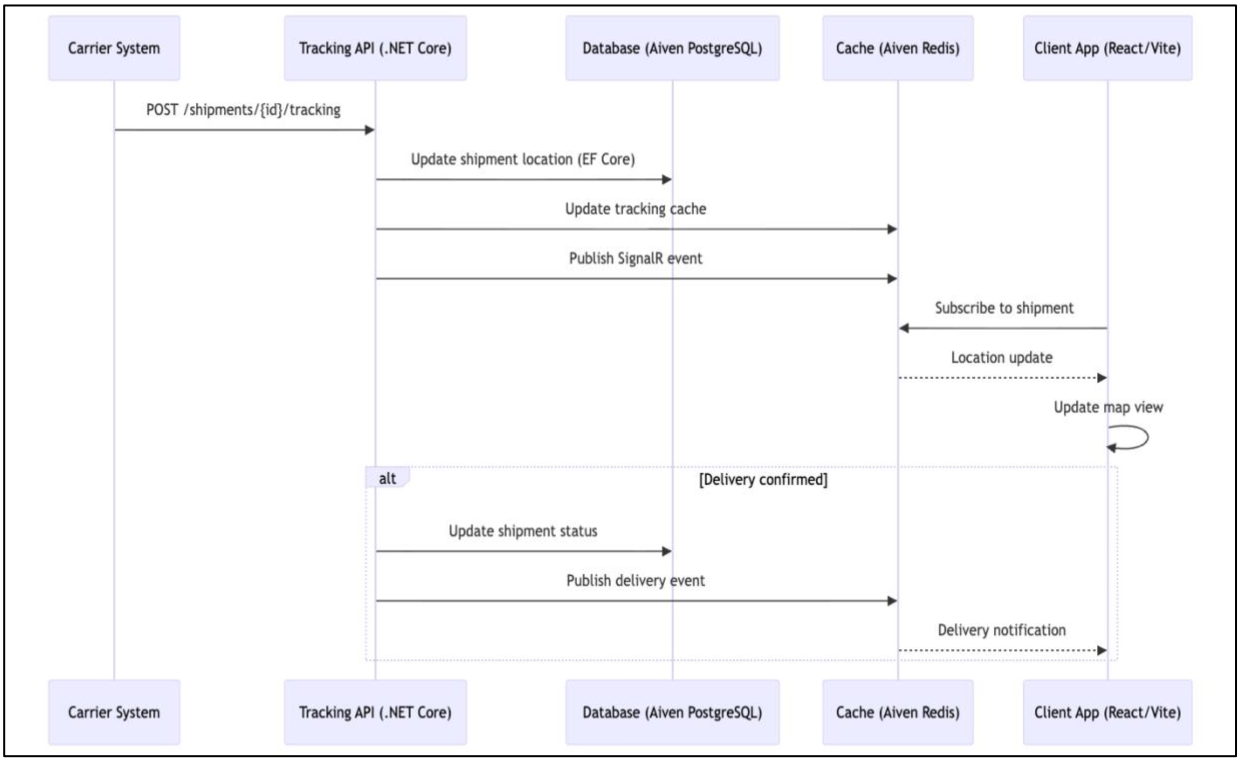
Note: Shifted to Azure-centric with optional multi-cloud via Azure Arc.



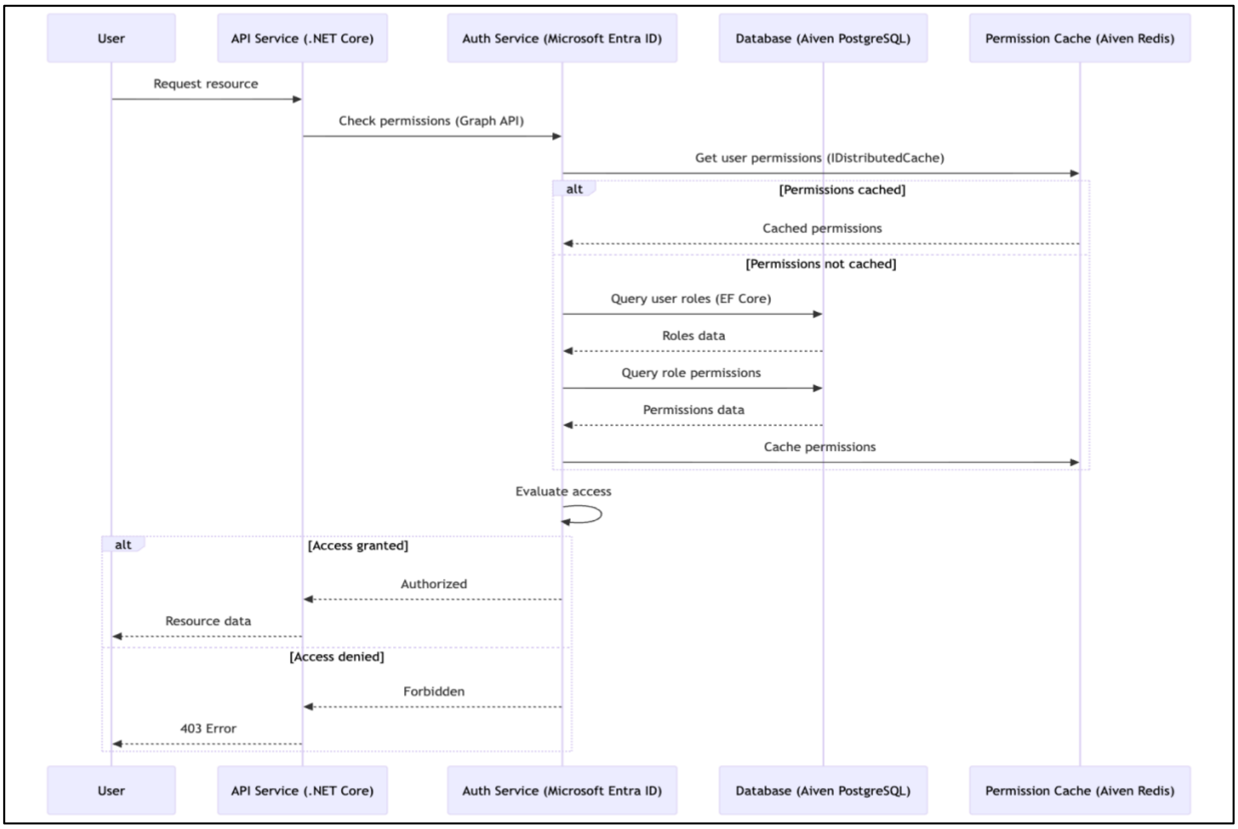
Inventory Management Flow


Warehouse User
Inventory API (.NET Core)
Database (Aiven PostgreSQL)
Cache (Aiven Redis)
Alert Service (Azure Functions)
Scan product barcode
Get product details (EF Core)
Product info
Calculate available quantity
Update inventory cache (IDistributedCache)
alt
[Quantity below reorder level]
Trigger reorder alert (Queue Trigger)
Alert notification
Inventory updated
Warehouse User
Inventory API (.NET Core)
Database (Aiven PostgreSQL)
Cache (Aiven Redis)
Alert Service (Azure Functions)

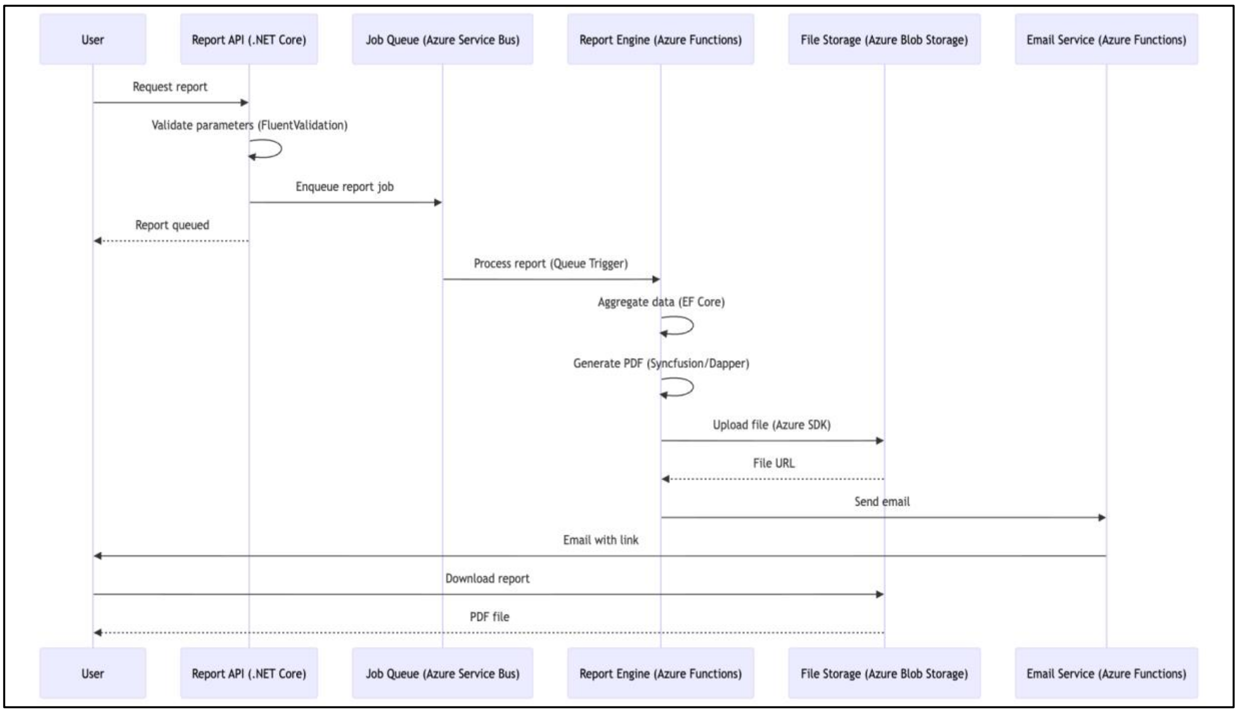
Shipment Tracking Flow



User Permission Flow



Report Generation Flow


User
Report API (.NET Core)
Job Queue (Azure Service Bus)
Report Engine (Azure Functions)
File Storage (Azure Blob Storage)
Email Service (Azure Functions)
Request report
Validate parameters (FluentValidation)
Enqueue report job
Report queued
Process report (Queue Trigger)
Aggregate data (EF Core)
Generate PDF (Syncfusion/Dapper)
Upload file (Azure SDK)
File URL
Send email
Email with link
Download report
PDF file
User
Report API (.NET Core)
Job Queue (Azure Service Bus)
Report Engine (Azure Functions)
File Storage (Azure Blob Storage)
Email Service (Azure Functions)

API Response Formats

Success Response

JSON

```json
{
  "success": true,
  "data": {
    "id": "uuid",
    "name": "Example",
    "createdAt": "2024-01-10T10:00:00Z"
  },
  "meta": {
    "timestamp": "2024-01-10T10:00:00Z",
    "version": "1.0"
  }
}
```

Error Response

JSON

```json
{
  "success": false,
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Invalid input parameters",
    "details": [
      {
        "field": "email",
        "message": "Invalid email format"
      }
    ]
  },
  "meta": {
    "timestamp": "2024-01-10T10:00:00Z",
    "requestId": "req-12345"
  }
}
```

Paginated Response

JSON

```json
{
  "success": true,
  "data": [
    { "id": "1", "name": "Item 1" },
    { "id": "2", "name": "Item 2" }
  ],
  "pagination": {
    "page": 1,
    "pageSize": 20,
    "total": 100,
    "totalPages": 5,
    "hasNextPage": true,
    "hasPreviousPage": false
  }
}
```

Rate Limiting Configuration

Implementation Strategy

To enforce these limits within your **.NET 8** and **Azure** ecosystem, you can utilize the following components:

- **ASP.NET Core Rate Limiting Middleware:** Introduced in .NET 7/8, this allows you to define policies (Fixed Window, Sliding Window, or Token Bucket) directly in your `Program.cs`.
- **Redis-based Rate Limiting:** Since you are using **Aiven Redis**, you can implement a distributed rate limiter. This ensures that if you have multiple instances of your microservices running in **AKS (Azure Kubernetes Service)**, the request count is synchronized across all nodes.
- **Azure API Management (APIM):** If you scale further, placing APIM in front of your services allows you to handle "Throttling" at the gateway level before the request even reaches your compute layer.

Endpoint Type	Requests/Minute	Burst
Authentication	5	10
Read Operations	60	100
Write Operations	30	50
Bulk Operations	10	20
Search	30	50

Implementation Guide

To enforce these timeouts across your enterprise stack, you should apply settings at multiple layers:

1. **EF Core (Database):** Set the command timeout in your `DbContext` configuration:
   - `options.UseNpgsql(connectionString, o => o.CommandTimeout(10));`
2. **HttpClient:** For external API calls, ensure the client is configured to cancel after 30 seconds to avoid thread pool starvation.
3. **SignalR (WebSockets):** Configure the `ClientTimeoutInterval` and `KeepAliveInterval` in your Hub options to match the 30s idle requirement.
4. **Azure Functions:** For reports, use the `function.json` or `host.json` to extend the `functionTimeout` attribute to 10 minutes.

Operation	Timeout
API Request	30s
Database Query	10s
File Upload	5m
Report Generation	10m
WebSocket Connection	30s (idle)

API Versioning Strategy

URL-based Versioning

```
/api/v1/orders
/api/v2/orders
```

Header-based Versioning

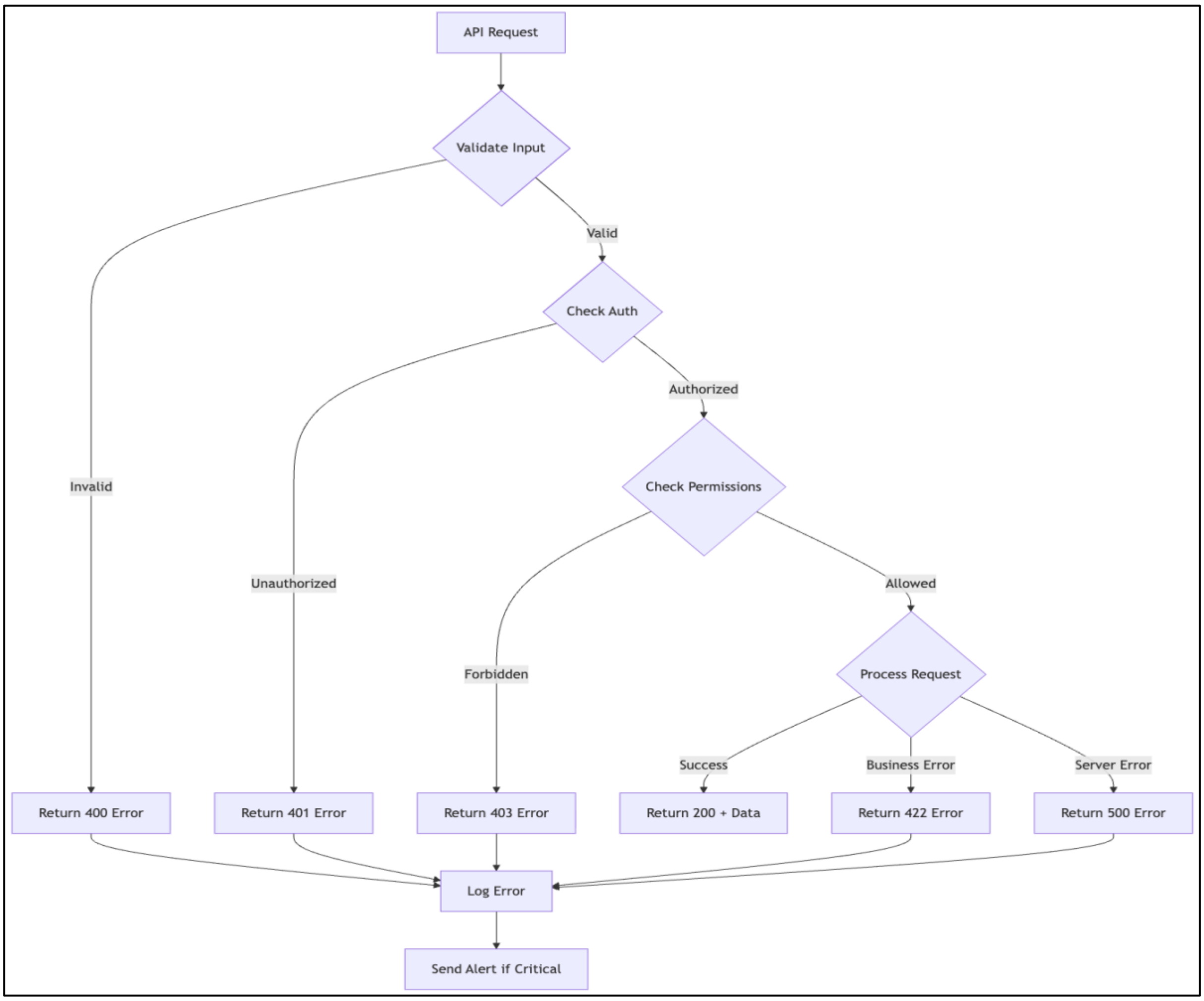
text

```
Accept: application/vnd.imsop.v1+json
```

Deprecation Policy

- Announce deprecation 6 months in advance
- Maintain deprecated version for 12 months
- Provide migration guide
- Support both versions during transition

Error Handling Flow



WebSocket Connection Management

Connection Flow

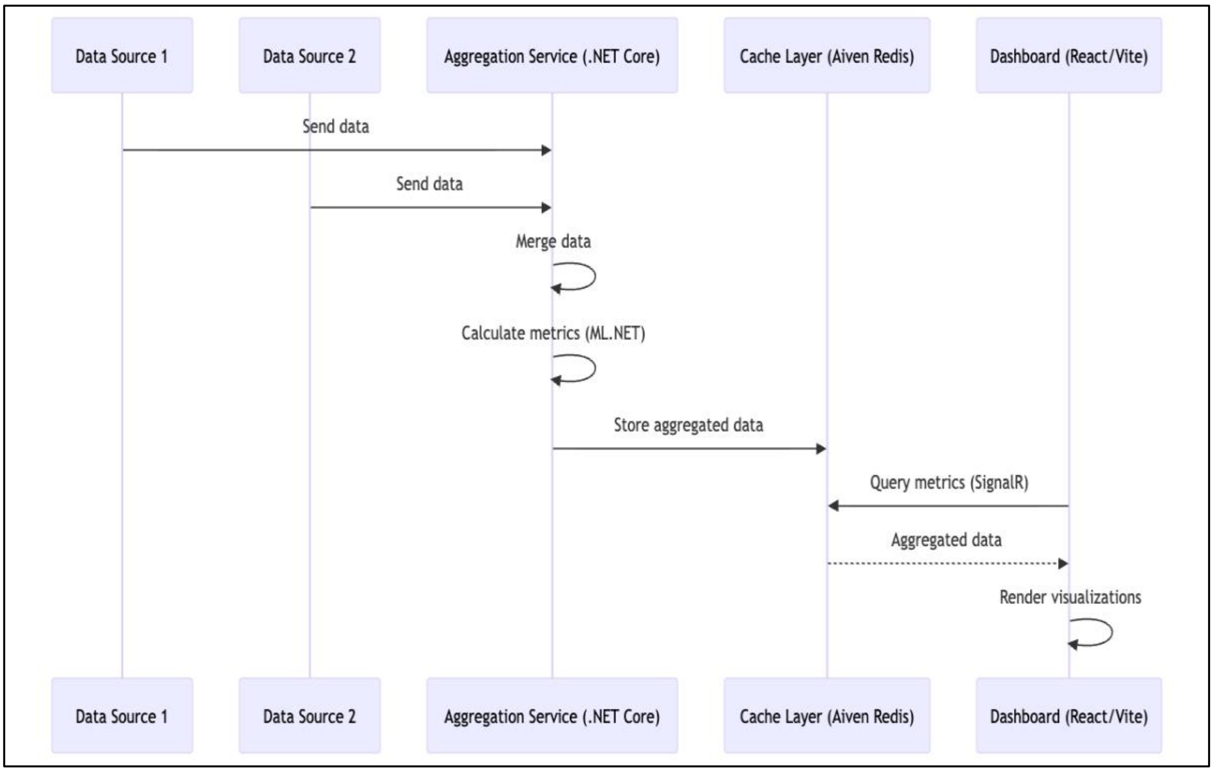
```
1. Client initiates SignalR connection
2. Server validates JWT token (Microsoft Entra ID)
3. Server subscribes client to channels
4. Server sends initial state
5. Client receives real-time updates
6. Connection maintained with heartbeat
7. Client disconnects or timeout
```

Message Format

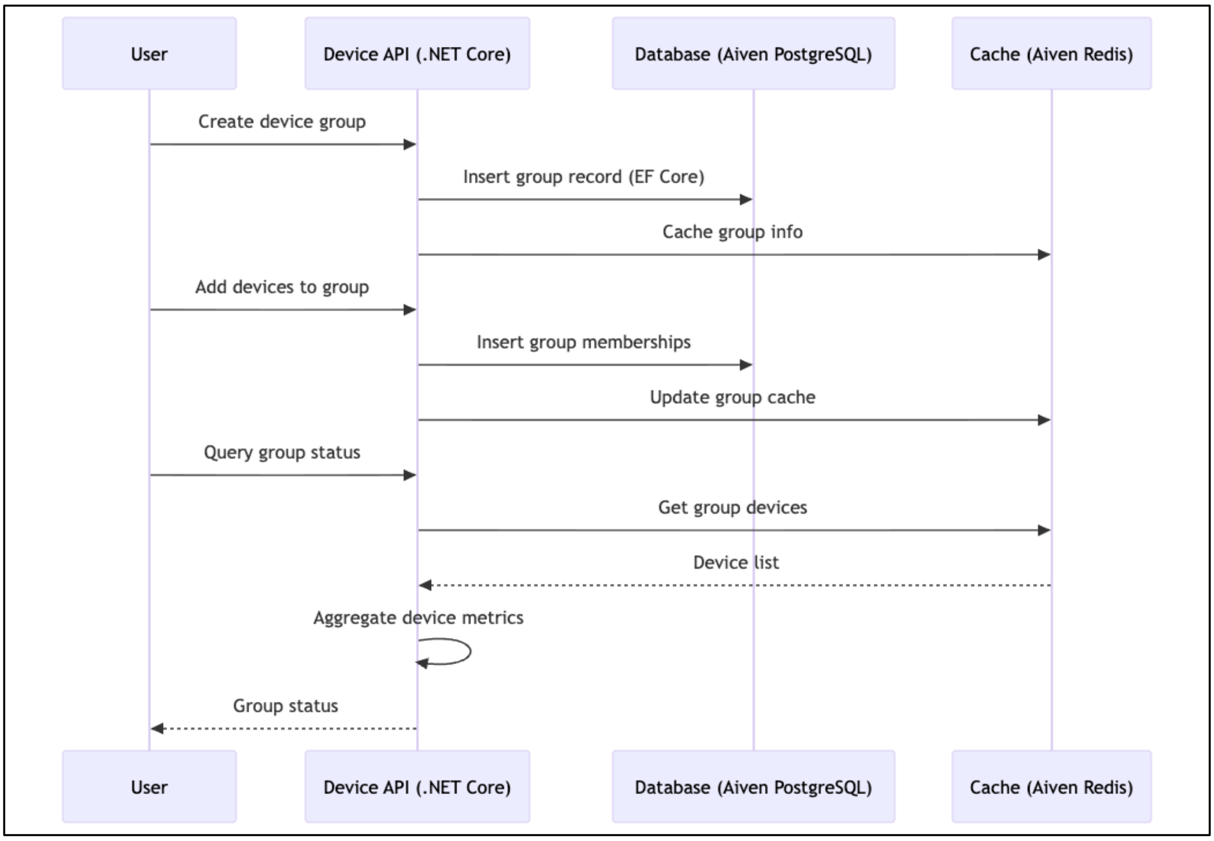
JSON

```json
{
  "type": "update",
  "channel": "shipments:123",
  "event": "status_changed",
  "data": {
    "shipmentId": "123",
    "status": "in_transit",
    "location": "New York, NY"
  },
  "timestamp": "2024-01-10T10:00:00Z"
}
```

Data Aggregation Flow



Multi-Device Grouping Flow



Performance Optimization Strategies

Caching Strategy

- Cache frequently accessed data (5 min TTL) using Aiven Redis
- Cache user permissions (10 min TTL)
- Cache product catalog (1 hour TTL)
- Cache organization settings (1 day TTL)

Query Optimization

- Use database indexes for common queries (EF Core LINQ)
- Implement query result pagination
- Use projection to select only needed fields
- Batch related queries together (EF Core)

Async Processing

- Queue long-running operations (Azure Service Bus)
- Process reports asynchronously (Azure Functions)
- Send notifications asynchronously
- Archive data asynchronously

Load Distribution

- Distribute requests across multiple servers (Azure App Service)
- Use message queues for async tasks
- Cache responses at CDN level (Azure CDN)
- Implement request batching