

Integration Architecture

[Author Name]

Université de Toulon, LIS UMR CNRS 7020

2026-10-03

Table des matières

1 Purpose of this Document	1
2 Integration Strategy	1
3 Integration Architecture Overview	1
4 System Interfaces	2
5 Interface Contracts	2
6 Integration Patterns	2
7 Message Formats	3
7.1 REST API Example	3
7.2 Event Message Example	3
8 Protocol Specifications	3
8.1 REST APIs	3
8.2 Message Queues	4
8.3 Events	4
9 Service Contracts	4
10 System Integration Inventory	4
10.1 External Systems Integration	4
10.2 Internal Integration Inventory	4
11 Interface Patterns	5
12 API Management	5
13 Integration Security	5
14 Integration Governance	5
15 Template Validation Checklist	6
16 Related Views	6

1 Purpose of this Document

This document defines the integration strategies, patterns, and specifications for [System Name], covering both internal component interactions and external system interfaces.

2 Integration Strategy

Integration Principles:

- [Key principle 1, e.g., "API-first design"]
- [Key principle 2, e.g., "Event-driven for asynchronous processes"]
- [Key principle 3, e.g., "Contract-driven development"]

Integration Types:

Synchronous:

- REST APIs: [When to use]
- GraphQL: [When to use]
- gRPC: [When to use]
- SOAP: [When to use, if applicable]

Asynchronous:

- Message Queues: [When to use]
- Event Streams: [When to use]
- Webhooks: [When to use]
- File Transfer: [When to use, if applicable]

3 Integration Architecture Overview

graph TD

```
A[Client Applications] --> B[API Gateway]
B --> C[Backend Services]
C --> D[Databases]
```

```

C --> E[Message Broker]
E --> F[Event Consumers]
B --> G[External Systems]
H[Partner Systems] --> B

```

4 System Interfaces

```

@startuml
package "Integration Layer" {
    interface "Public API" as api
    interface "Event Bus" as events
    interface "Message Queue" as queue

    component [API Gateway]
    component [Event Handler]
    component [Queue Manager]

    api - [API Gateway]
    events - [Event Handler]
    queue - [Queue Manager]
}

cloud "External Systems" {
    [System A]
    [System B]
}

[API Gateway] --> [System A]
[Event Handler] --> [System B]
@enduml

```

5 Interface Contracts

```

API Contracts:
  Version Control:
    Strategy: [API versioning approach]
    Deprecation: [Policy for deprecating APIs]

  Format:
    Primary: [OpenAPI/GraphQL/Proto/

```

RAML/etc.]

Documentation: [Link to main documentation]

Change Process: [How contracts are updated]

Standards:

Authentication: [Methods]

Rate Limiting: [Approach]

Error Handling: [Standard error format]

Pagination: [Standard approach]

Filtering: [Standard approach]

Event Contracts:

Version: [Versioning strategy]

Schema: [Avro/JSON Schema/Protocol Buffers/etc.]

Validation: [When/how validation occurs]

Routing: [Topic/routing patterns]

Delivery: [Guarantees]

6 Integration Patterns

Pat- tern	Use Case	Implementa- tion	Example
Re- quest/ Res- ponse	Syn- chro- nous opera- tions	REST/Graph- QL APIs	User au- then- tica- tion
Pu- blish/ Sub- scribe	Event notifi- cations	Message bro- ker	Order status up- dates
Event Sour- cing	State tra- cking	Event store	Audit log- ging
CQRS	Perfor- mance	Split read/ write models	Pro- duct

Pat-tern	Use Case	Implementa-tion		Example
	optimi-zation			cata-log
Saga	Distri-buted tran-sac-tions	Choreogra-phy/Orches-tration		Order pro-ces-sing
API Gate-way	Unified access	API	mana-gement plat-form	Client ap-plica-tions
BFF (Ba-ckend For Fron-tend)	UI op-timiza-tion	Purpose-built APIs		Mo-bile appli-cation
Web-hook	Exter-nal no-tifica-tions	HTTP	call-backs	Pay-ment noti-fica-tions

```
{
  "productId": "PROD-456",
  "quantity": 2,
  "price": 29.99
},
{
  "total": 59.98
}
```

7.2 Event Message Example

Sample Message Format:

```
header:
  version: "1.0"
  timestamp: "2023-06-15T14:22:00Z"
  correlation_id: "corr-123456"
  source: "order-service"
  type: "order.created"
payload:
  orderId: "ORD-12345"
  customerId: "CUST-789"
  items:
    - productId: "PROD-456"
      quantity: 2
      price: 29.99
  total: 59.98
```

7 Message Formats

7.1 REST API Example

```
{
  "id": "123",
  "timestamp": "2023-06-15T14:22:00Z",
  "type": "order.created",
  "data": {
    "orderId": "ORD-12345",
    "customer": {
      "id": "CUST-789",
      "name": "Example Customer"
    },
    "items": [
```

8 Protocol Specifications

8.1 REST APIs

- **Base URL Structure:** `https://api.example.com/v1/resources`
- **Authentication Method:** OAuth 2.0 with JWT
- **Rate Limiting:** 1000 requests per minute per client
- **Versioning Strategy:** URI path versioning (e.g., `/v1/resource`)
- **Standard Headers:**
 - Authorization: Bearer {token}
 - Content-Type: application/json
 - Accept: application/json

- X-Correlation-ID: {uuid}
- **Status Codes:**
 - 200: Success
 - 201: Created
 - 400: Bad Request
 - 401: Unauthorized
 - 403: Forbidden
 - 404: Not Found
 - 429: Too Many Requests
 - 500: Internal Server Error

8.2 Message Queues

- **Queue Naming Conventions:** {environment}.{domain}.{action}
- **Message Persistence:** 7 days
- **Delivery Guarantees:** At-least-once
- **Error Handling:** Dead letter queue after 3 retries
- **Message Size Limit:** 256KB
- **Partitioning Strategy:** By entity ID

8.3 Events

- **Event Schema Registry:** [URL or tool]
- **Naming Convention:** {domain}.{entity}.{action}
- **Event Versioning:** Schema versioning with backward compatibility
- **Retry Policies:** Exponential backoff with jitter
- **Consumer Groups:** One consumer group per service
- **Error Handling:** Error topic for failed processing

9 Service Contracts

```
@startuml
participant "Consumer" as C
participant "Provider" as P
database "Service Registry" as SR

C -> SR: Discover Service
SR --> C: Service Contract
```

```
C -> P: Request (per contract)
P --> C: Response
@enduml
```

10 System Integration Inventory

10.1 External Systems Integration

```
External Dependencies:
  APIs:
    - System: [External system name]
      Owner: [Company/team]
      Protocol: [REST/GraphQL/gRPC/SOAP/etc.]
      Purpose: [Business function]
      Data: [Key data exchanged]
      SLA: [Response time/availability/etc.]
      Authentication: [Method]
      Failure Impact: [Business impact]
      Fallback: [Alternative process]

  Events:
    - Stream: [Event stream name]
      Source: [Provider]
      Pattern: [pub/sub|queue]
      Volume: [Expected events/second]
      Latency: [Requirements]
      Data: [Key information]
```

10.2 Internal Integration Inventory

```
Service-to-Service:
  Synchronous:
    - Consumer: [Service name]
      Provider: [Service name]
      Pattern: [Request/Response]
      Protocol: [REST/GraphQL/gRPC]
      Purpose: [Function]
      Criticality: [High/Medium/Low]
```

Asynchronous:

- Publisher: [Service name]
- Consumers: [Service names]
- Pattern: [Pub/Sub]
- Protocol: [AMQP/Kafka/etc.]
- Events: [Event types]
- Purpose: [Function]

Documentation:

- API Reference
- Tutorials
- SDKs

Self-Service:

- API Key Management
- Usage Dashboard
- Sandbox Environment

11 Interface Patterns

Synchronous:

Pattern: Request/Response
Protocol: [REST/GraphQL/gRPC]
SLA:
Availability: 99.9%
Response Time: <500ms
Rate Limits: 1000 rps

Asynchronous:

Pattern: Pub/Sub
Protocol: [AMQP/Kafka/NATS/etc.]
Guarantees:
Delivery: At-least-once
Ordering: [Per partition/None]
Durability: [Persistent/Ephemeral]

13 Integration Security

Authentication:

External: [OAuth 2.0, API Keys, mTLS, etc.]
Internal: [Service accounts, mTLS, JWT, etc.]

Authorization:

Model: [RBAC, ABAC, etc.]
Enforcement: [API Gateway, Service Mesh, Code]

Data Protection:

Transport: [TLS 1.3]
Storage: [Encryption method]
PII Handling: [Approach]

12 API Management

API Gateway:

Implementation: [Product/platform]

Features:

- Authentication
- Authorization
- Rate Limiting
- Request Validation
- Response Transformation
- Analytics
- Caching

Developer Portal:

14 Integration Governance

Standards Enforcement:

- Contract Validation
- Security Review
- Performance Testing
- Compatibility Testing

Monitoring:

- API Usage Metrics
- Error Rates
- Response Times
- SLA Compliance

15 Template Validation Checklist

- ☐ Integration strategy defined
- ☐ Interface contracts documented
- ☐ Protocols specified
- ☐ Message formats defined
- ☐ Security controls identified
- ☐ External systems inventory complete
- ☐ Internal integration patterns documented
- ☐ Governance approach described
- ☐ Team review completed

16 Related Views

- System Context - Overall system context
- API Architecture - Detailed API design
- Component Architecture - Service details