API Integration and Enterprise Integration Patterns Course

26 – 30 April 2027
London (UK)

Dubai - UAE: +971 4 450 5697
Istanbul - Türkiye: +90 539 599 1206
Amman - Jordan: +962 79 712 3347

MERCURY

API Integration and Enterprise Integration Patterns Course

Ref: 121362_1029451 **Date:** 26 – 30 April 2027 **Location:** London (UK) **Fees:** 5900 Euro

API Integration and Enterprise Integration Patterns Course is a 1-week course for integration engineers, architects and developers who connect business systems that were never designed to talk to each other. You compare APIs, messaging, events and file transfer, choose the right pattern for each flow and take back an Integration Design Blueprint for one of your own interfaces. Mercury Training Center delivers this API integration and enterprise integration patterns course.

About This Course

Point-to-point links pile up until one change in an ERP or CRM breaks five other systems. To follow the course you should already build, support or specify interfaces between applications. You practice on integration scenarios, draw message flows, choose patterns and record each design decision with its trade-offs.

Who It Is For

- Engineers who build and support connections between enterprise applications.
- Architects who decide how systems exchange data across cloud and on-premises estates.
- Developers who expose or consume services from core business systems.
- Middleware and platform staff who run service buses, brokers or iPaaS tenants.
- Analysts who map business processes to the data flows behind them.

This course is not for readers who mainly need to run APIs as products, with gateways, developer portals and monetization, who suit an API management course, nor for business users who want a non-technical overview, who suit a digital transformation awareness course.

Competencies You Will Build

- Integration Style Selection: you choose between synchronous APIs, messaging, events and batch for each flow and justify the choice.
- Pattern Application: you name and apply enterprise integration patterns such as content-based router, aggregator and translator.
- Event-Driven Design: you define topics, event schemas and consumers so systems stay loosely coupled.
- Reliable Delivery: you design retries, idempotent receivers and dead letter handling so messages are not lost or duplicated.
- Platform Choice: you compare service bus, iPaaS and broker options against your estate and skills.
- Interface Contracts: you write OpenAPI and AsyncAPI contracts that teams can build and test against.

Dubai - UAE: +971 4 450 5697
Istanbul - Türkiye: +90 539 599 1206
Amman - Jordan: +962 79 712 3347

MERCURY

What You Will Be Able to Do

By the end of the course you will be able to:

- From an inventory of current interfaces, map point-to-point links and mark the ones to consolidate first.
- Given an order-to-cash process, draw the message flow and place the right routing and transformation patterns on it.
- Using orchestration and choreography, design a long-running transaction with compensating steps.
- Given a legacy system with no API, choose an adapter, change data capture or file approach and record the trade-offs.
- From security and audit needs, specify OAuth 2.0 scopes, encryption and logging for each integration flow.
- Using monitoring data, set latency, error and backlog thresholds for an integration flow.

Course Content

Day 1: Integration Styles and Landscape

- File Transfer, Shared Database, Remote Procedure and Messaging Styles Compared
- Interface Inventory and Point-to-Point Dependency Mapping
- Synchronous Versus Asynchronous Communication for Business Flows
- API-Led Connectivity Layers for System, Process and Experience APIs
- Integration Decision Records for Recording Trade-Offs

Day 2: Messaging and Enterprise Integration Patterns

- Message Channels, Point-to-Point Queues and Publish-Subscribe Topics
- Content-Based Router, Splitter and Aggregator Patterns in Order Processing
- Message Translator and Canonical Data Model for Cross-System Mapping
- Guaranteed Delivery, Idempotent Receiver and Dead Letter Channel
- Correlation Identifiers and Message Tracing Across Systems

Day 3: Event-Driven and Microservices Integration

- Event Notification Versus Event-Carried State Transfer
- AsyncAPI Contracts and Event Schema Versioning
- Orchestration Versus Choreography for Long-Running Processes
- Saga Pattern With Compensating Transactions for Distributed Updates
- Transactional Outbox and Change Data Capture From Legacy Databases

Dubai - UAE: +971 4 450 5697
Istanbul - Türkiye: +90 539 599 1206
Amman - Jordan: +962 79 712 3347

Day 4: Platforms, APIs and Operations

- Enterprise Service Bus, iPaaS and Message Broker Selection Criteria
- OpenAPI Contracts and Versioning Rules for System APIs
- OAuth 2.0 Scopes, Encryption and Audit Logging for Integration Flows
- Hybrid Integration Between Cloud Services and On-Premises Applications
- Monitoring Latency, Error Rates and Queue Backlogs

Day 5: Practice Building the Integration Design Blueprint

- Exercise: Pattern Map for an ERP to E-Commerce Order Flow
- Exercise: Saga Design for a Payment and Inventory Update
- Exercise: Platform Choice for a Hybrid Cloud Estate
- Exercise: Recovery Plan for a Failed Message Batch
- Completing the Integration Design Blueprint for Your Own Interface

Case Studies and Exercises

These are suggested activities.

- Case study: a retailer's online store, warehouse system and ERP disagree on stock levels; you redesign the flow with events and an idempotent receiver.
- Case study: a bank must expose customer data held in a mainframe to mobile channels; you choose between an adapter, change data capture and a system API.
- Exercise: a hospital links laboratory, billing and patient record systems; you draw the message flow and pick routing and translation patterns.
- Exercise: a logistics operator moves from nightly files to near real-time tracking events; you write the AsyncAPI contract and error handling.

What You Take Back

The Integration Design Blueprint is the design you complete for one of your own interfaces during the course. In your first month back, you use it to review that interface with your team, agree the contracts with the systems involved and plan its move away from point-to-point links.

- A current-state interface map with dependencies.
- A target message flow annotated with integration patterns.
- OpenAPI or AsyncAPI contracts and error handling rules.
- Platform choice, security settings and monitoring thresholds.

Dubai - UAE: +971 4 450 5697
Istanbul - Türkiye: +90 539 599 1206
Amman - Jordan: +962 79 712 3347

Quick Answers FAQ

What should I know before API integration and enterprise integration patterns training?

You should already build, support or specify interfaces and be able to read JSON or XML messages. Programming experience helps but no specific language or platform is required.

How does enterprise integration patterns training differ from an API management course?

It looks at how whole systems exchange data, with APIs as one style beside messaging, events and files. API management courses focus on running APIs as products through gateways, portals, usage plans and analytics.

When should an integration use events instead of a synchronous API call?

Use events when several systems react to the same change, when the sender should not wait, or when a receiver may be offline. Use a synchronous call when the caller needs an immediate answer to continue.

What do I take back from the API Integration and Enterprise Integration Patterns Course?

You take back the Integration Design Blueprint for one of your interfaces: a current-state map, a pattern-annotated target flow, contracts and error rules, and platform, security and monitoring choices.

For Your Manager

After the course, the team member can choose integration styles, apply enterprise integration patterns, design reliable event flows and specify contracts and monitoring. They return with an Integration Design Blueprint for one live interface, first used to review that interface with the team and plan its move off fragile point-to-point links.