MULESOFT ARCHITECTURE:

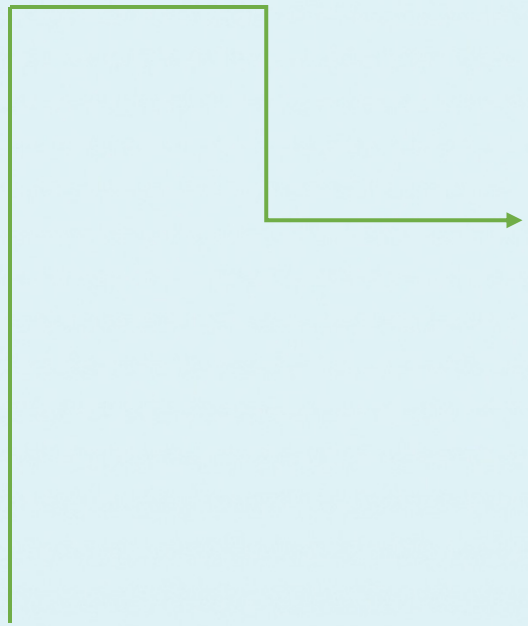
Integration Patterns Part 1

PRACTIC LEARNING

Integration Patterns Introduction

1. API-led connectivity
2. Messaging patterns
3. Reliability pattern
4. Publish and subscribe pattern

API-Led Connectivity



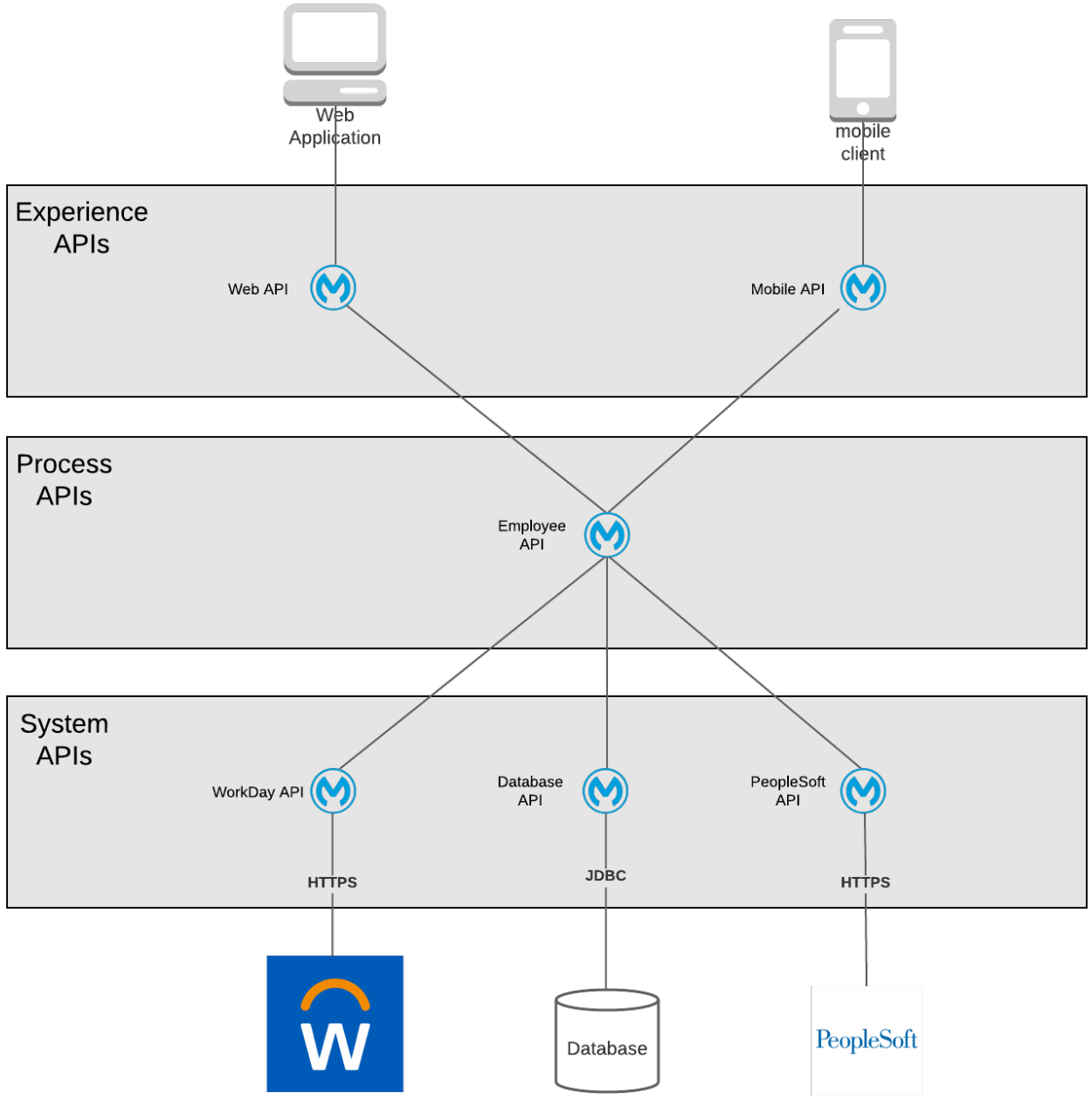
Integration Pattern: API-Led Connectivity

- API-led connectivity allows system to system connectivity through a network of reusable APIs
- API-led connectivity should be used wherever possible with synchronous processing
- Use API-led connectivity when:
  - At least one of the 3 API layers can be applied to fulfill requirements
  - API consumers can initiate the communication/process on demand
- Do you always need 3 layers of API connectivity?

API-Led Connectivity Scenario #1

Design an API strategy where a web application and a mobile application need to get employee information from 3 different HR systems: WorkDay, a custom database, and PeopleSoft.

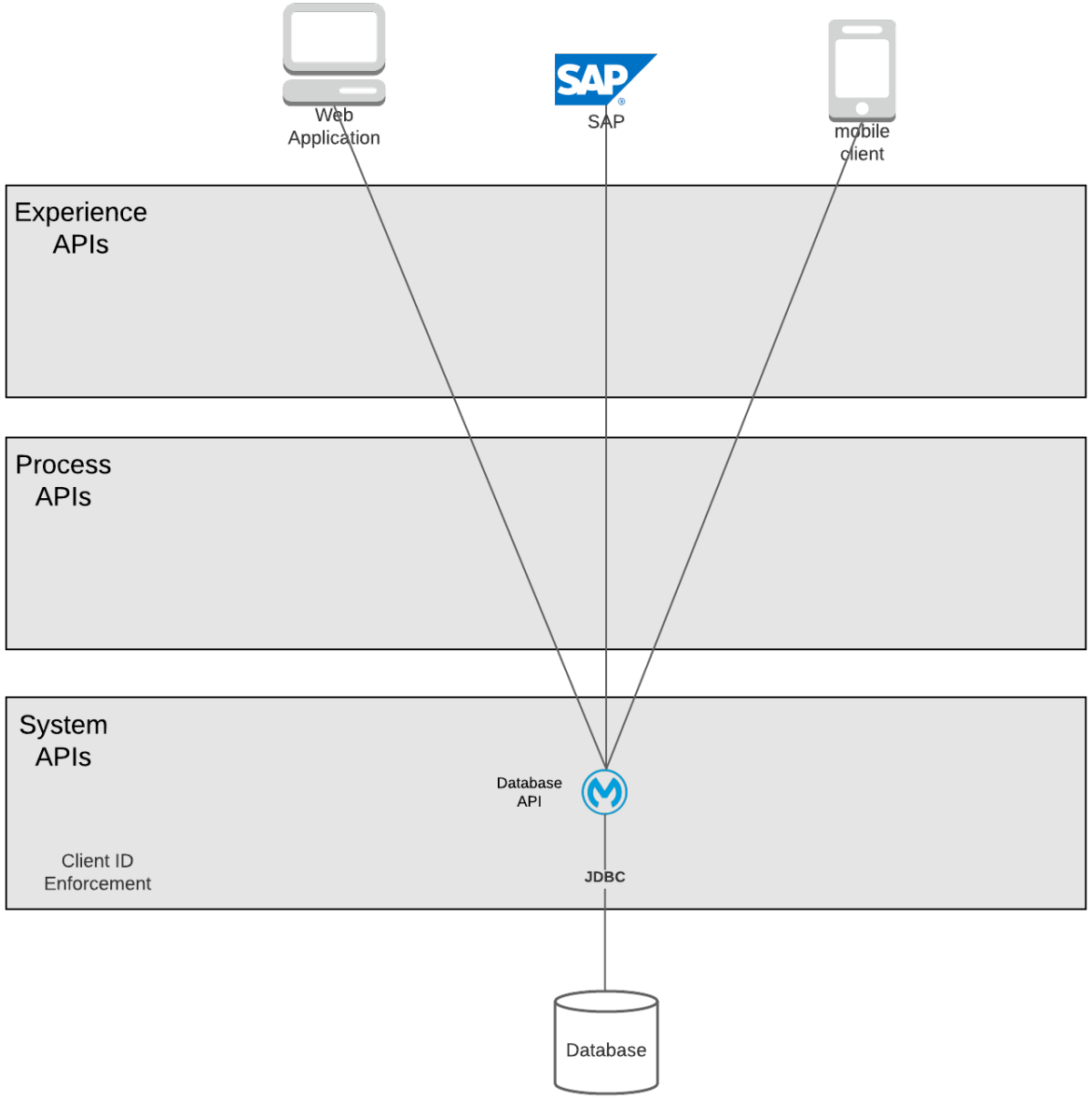
API-Led Connectivity Scenario #1 Architecture



API-Led Connectivity Scenario #2

Design an API strategy where various internal systems across the organization need to interact with a database. Little to no data transformation is required for consumers of the database data, and a single reusable API can be reused across the organization because the organization requires easy onboarding and minimal API governance.

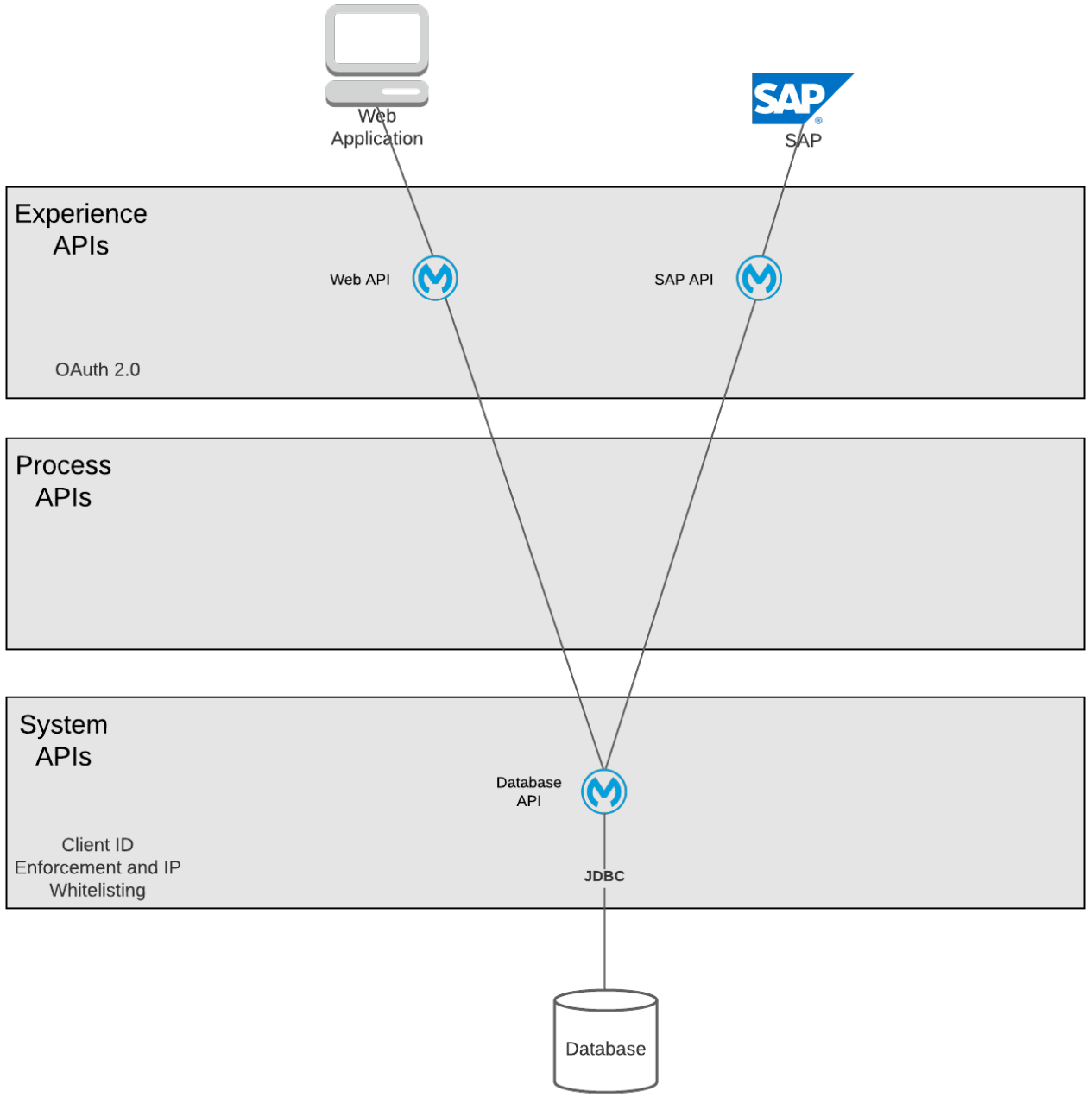
API-Led Connectivity Scenario #2 Architecture



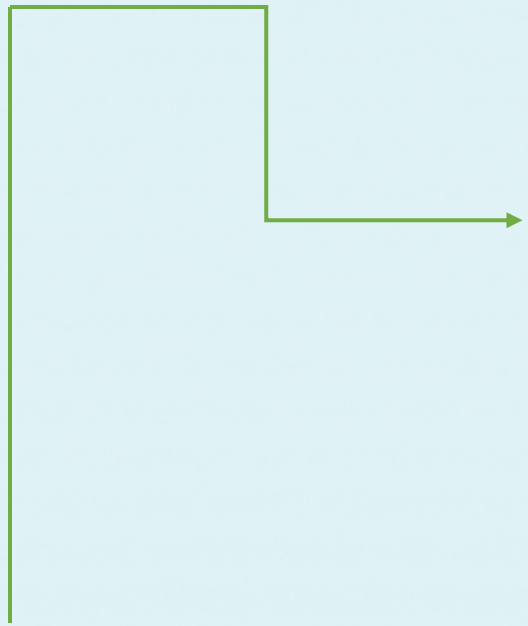
API-Led Connectivity Scenario #3

Design an API strategy where a web application and SAP require different responses from data that lives in a database. The organization deploys to CloudHub, but has a limited number of vCores left in Production so creating a solution that minimizes vCore usage is paramount.

API-Led Connectivity Scenario #3 Architecture



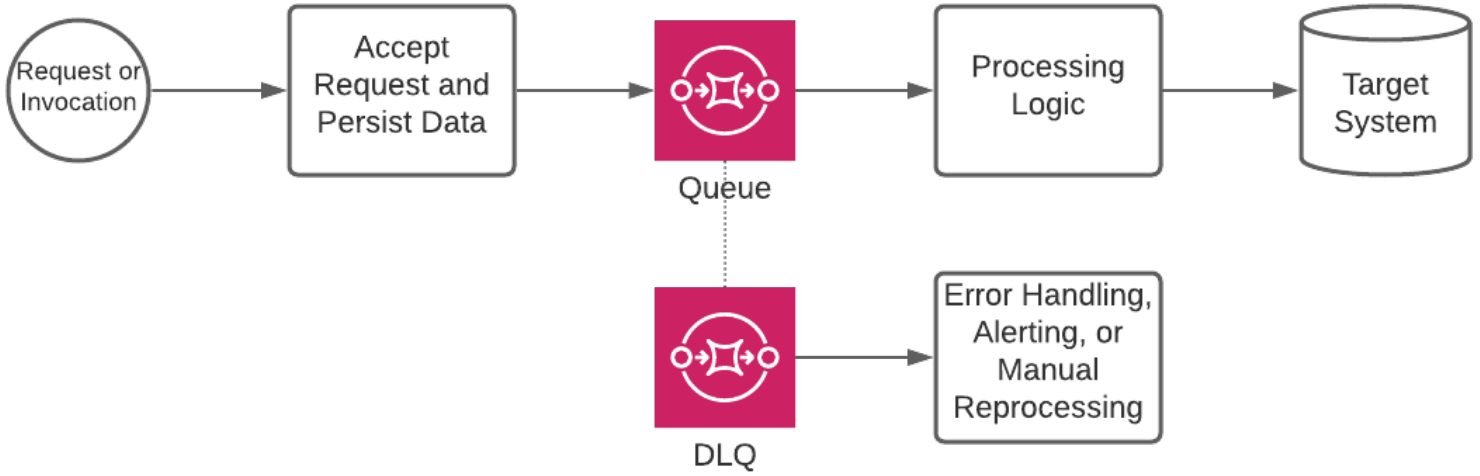
Messaging Integration Patterns



Integration Pattern: Reliability Pattern

- The reliability pattern uses a queue to guarantee delivery of messages to a destination
- The reliability pattern should be used when
  - Asynchronous processing is acceptable
  - Guaranteed delivery of data is a requirement
  - Message loss is unacceptable
- Combine API-led connectivity with the reliability pattern through HTTP 202 Accepted responses letting consumers know the request has been received
- Recommended to set retries for messages and leverage a dead letter queue (DLQ) when implementing the reliability pattern

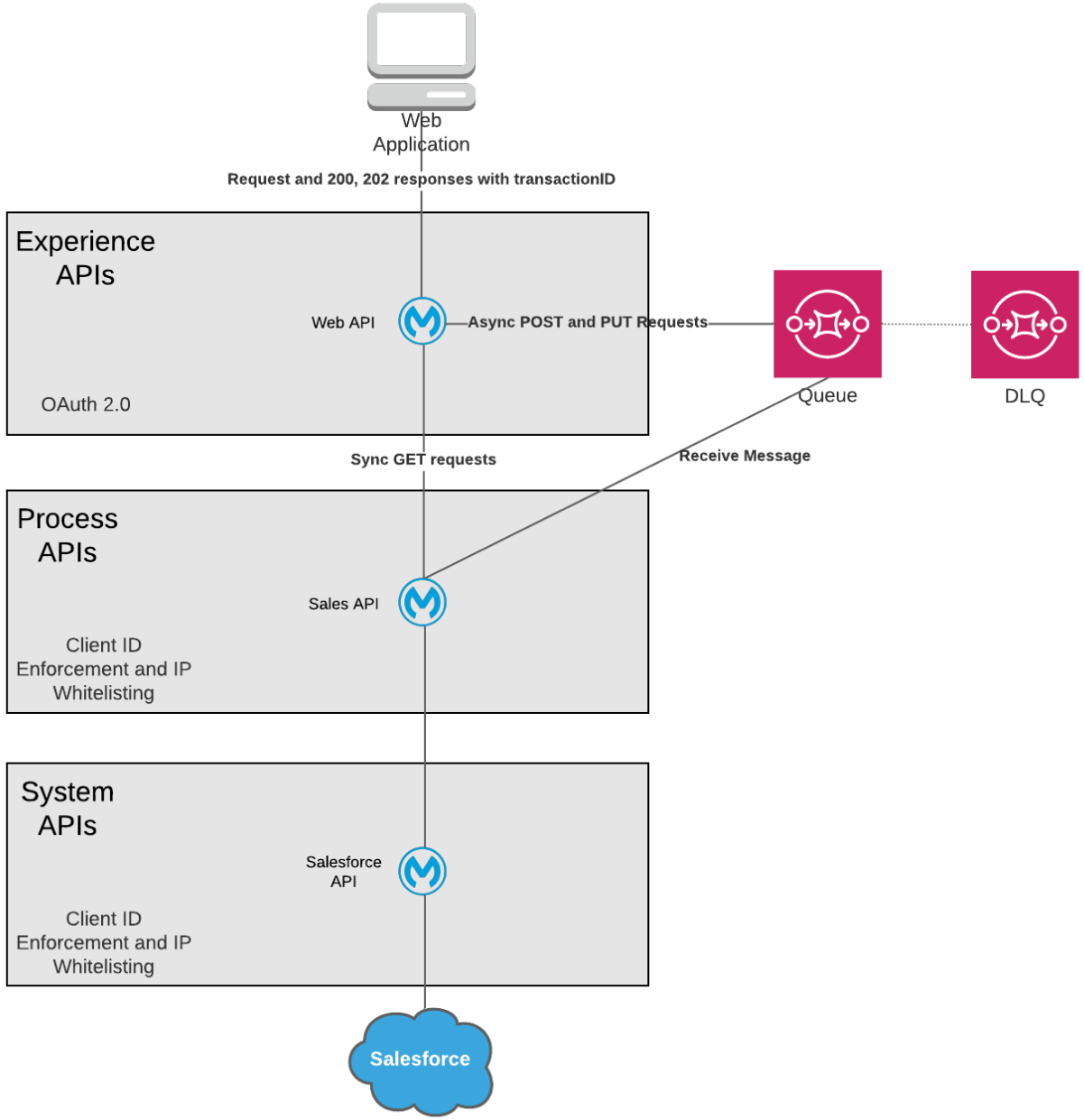
Reliability Pattern Architecture



Reliability Pattern Scenario #1

Design an API strategy where a web application needs to interact with Salesforce leads and opportunities data for an organization's sales team. The web application needs synchronous responses to get leads and opportunities on demand, but does not require synchronous responses to create and update leads and opportunities. When creating and updating leads and opportunities, it is crucial that the leads and opportunities are created or updated reliably with no loss of data.

Reliability Pattern Scenario #1 Architecture



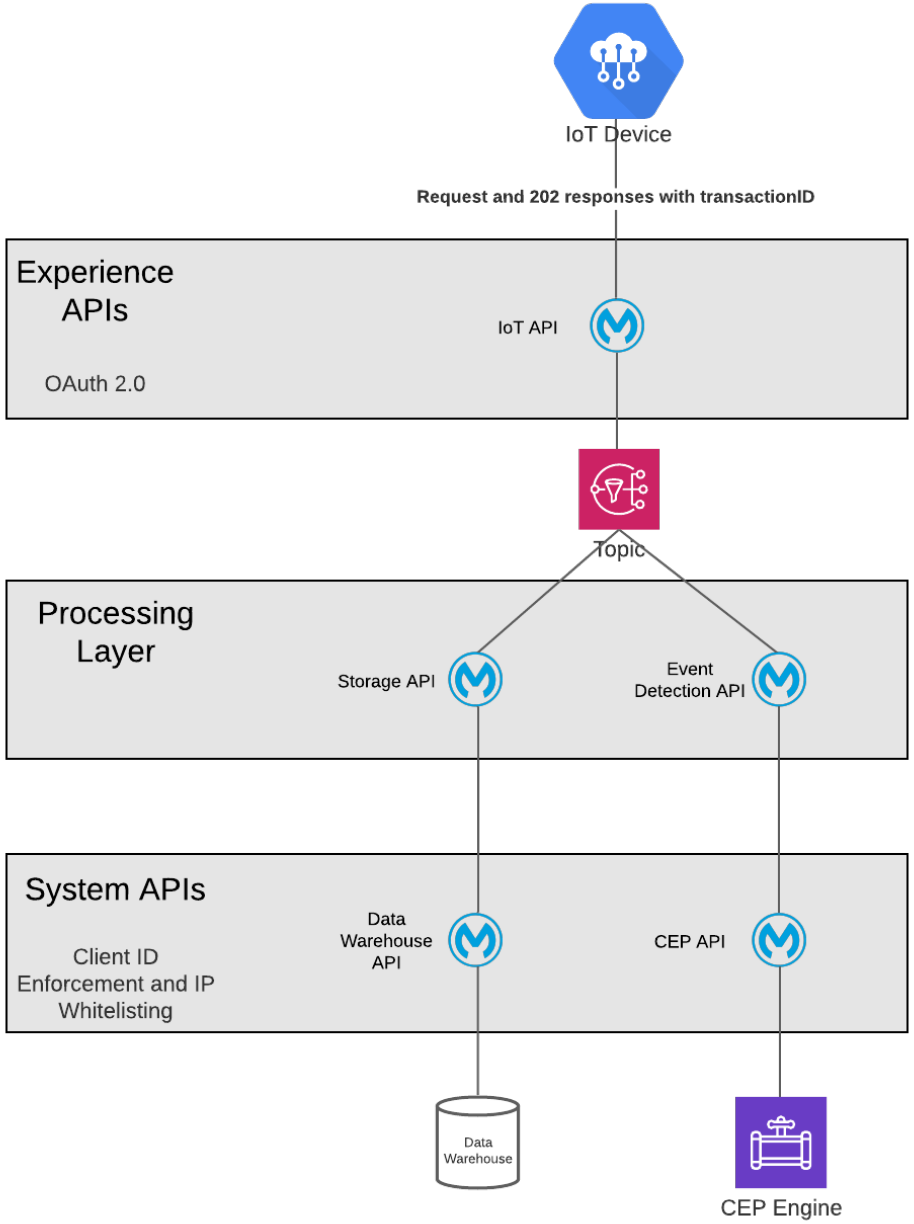
Integration Pattern: Publish and Subscribe (Pub/Sub)

- Pub/sub uses a topic to allow a single message to be sent to multiple subscribers
- Pub/sub should be used when
  - Asynchronous processing is acceptable
  - Many downstream destinations to send a single message to (fanout pattern)
  - There is a need to easily hook in multiple downstream targets
- Can combine API-led connectivity with pub/sub through HTTP 202 Accepted responses letting consumers know the request has been received
- Can combine reliability pattern with pub/sub to guarantee message delivery

Pub/Sub Scenario #1

Design an API strategy that allows for an IoT device to send API requests to a processing engine. The processing engine needs to send the IoT data to a data warehouse and a microservices layer for processing and filtering to power an event detection system.

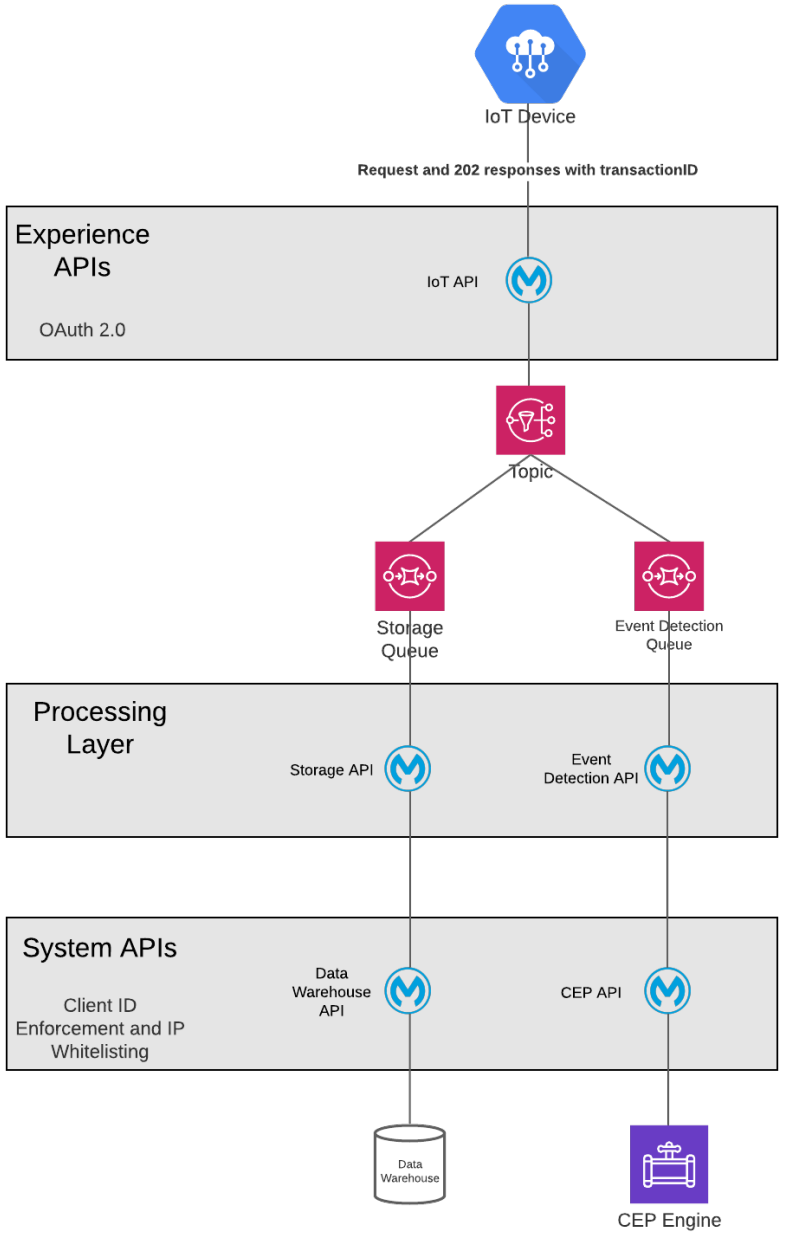
Pub/Sub Scenario #1 Architecture



Pub/Sub Scenario #2

Design an API strategy that allows for an IoT device to send API requests to a processing engine. The processing engine needs to send the IoT data to a data warehouse and a microservices layer for processing and filtering to power an event detection system. It is critical that the data is delivered to both the data warehouse and microservices layer.

Pub/Sub Scenario #2 Architecture



Integration Patterns Summary

- API-led connectivity uses, considerations, and designs
- Reliability pattern uses, considerations, and designs
- Pub/sub uses, considerations, and designs

Additional Reading

- https://blogs.mulesoft.com/learn-apis/api-led-connectivity/what-is-api-led-connectivity/
- https://docs.mulesoft.com/mule-runtime/4.3/reliability-patterns
- https://docs.aws.amazon.com/sns/latest/dg/sns-sqs-as-subscriber.html