Decision Reference Guide -- Integration Patterns (Version 1.0)

Author: Enterprise Architecture Group

Purpose: Provide guidelines for selecting the appropriate integration pattern across cloud and on-premises systems.

1. Event-Driven Integration

Description

Event-driven integration is recommended when systems communicate asynchronously using messages, events, or notifications rather than direct request/response.

Use When

- Latency requirements are near-real-time.
- Workload type is high-volume streaming or event-based triggers.
- Systems need loose coupling.
- Consumers may appear or disappear dynamically.
- The producing system must not block waiting for a reply.

Avoid When

- Strong consistency is required.
- A synchronous response is needed.
- Transaction boundaries must span multiple systems.

Typical Technologies

- Azure Event Hubs
- Azure Service Bus Topics
- Kafka

2. API-Based Synchronous Integration

Description

Direct request/response via HTTP APIs is suitable when consumers need immediate results or must validate a process step before continuing.

Use When

- Latency must be sub-second or human-level (<2s).
- Consumers require a deterministic response.
- Workload is moderate in volume.
- Strong or medium consistency is required.

Avoid When

- Producers cannot handle load spikes.
- Workload is high-volume or streaming.
- The process can tolerate eventual consistency.

Typical Technologies

- Azure API Management
- Azure Functions (HTTP trigger)
- Logic Apps (synchronous workflows)

3. ETL / Data Pipeline Integration

Description

Bulk data movement across systems where the objective is to transform, enrich, and load data into analytics platforms or operational stores.

Use When

- Data volume is large or very large.
- Processing can occur in batch (hourly or nightly).
- Transformations involve joins, aggregations, or cleansing.
- Latency requirements allow multi-minute delays.

Avoid When

- Real-time or near-real-time behavior is required.
- Source systems cannot handle extraction load.
- Data quality must be guaranteed per record.

Typical Technologies

- Azure Data Factory
- Azure Synapse Pipelines
- Azure Databricks

4. File-Based Integration

Description

A pattern where systems exchange files (CSV, XML, JSON) for interoperability or regulatory reporting.

Use When

- Systems support only file interfaces.
- Volumes are medium to large.
- Latency is not strict.
- Batch processes are acceptable.

Avoid When

- Real-time behavior is required.
- File corruption risk is unacceptable.
- Data must be strongly consistent at transaction level.

Typical Technologies

- Azure Blob Storage
- SFTP Gateways
- Managed File Transfer systems

5. RPA-Based Integration

Description

Robotic Process Automation is used when systems do not expose APIs or integration mechanisms, and UI automation is the only option.

Use When

- Legacy system lacks API or integration extensibility.
- Volume is low or medium.
- Consistency requirements are flexible.
- The process must mimic human UI actions.

Avoid When

- High availability or mission-critical reliability is needed.
- Transaction volume is high.
- System UI changes frequently.

Typical Technologies

- Power Automate Desktop
- UiPath
- BluePrism

Summary Table

Pattern	Best For	Avoid If	Latency	Data Volume
Event-Driven	Streaming, async workflows	Need sync response	Near-real-time	High
API Sync	Validation, immediate response	Heavy load, async	Sub-second	Medium
ETL / Data Pipeline	Warehousing, analytics	Real-time needs	Minutes--hours	Large--Very Large
File-Based	Legacy systems	Need strong consistency	Minutes--hours	Medium--Large
RPA	Legacy UI	Needs reliability	Seconds	Low--Medium