Salesforce API Playbook

Best Practices for Integrating with the Salesforce Platform

Created by Irina Saavedra

For Salesforce Architects, Partners & Innovators

2025

“Passionate about enterprise integration.

Banking brain, tech heart — building toward Salesforce Solution Architecture one API at a time.”

This playbook reflects the thinking and habits I’ve developed on my architect journey.

Created to help others learn smart, not hard — just like I’ve been doing.

Jump to: Glossary A | Glossary B | Glossary C | API Deep Dives | Decision Tree | Architecture Diagrams

🌐 Book a Discovery Call

📄 Download Full Playbook PDF

This matrix helps you decide **which Salesforce API or Integration Pattern** to use based on **real-world business scenarios**. It's designed to help architects and developers think in terms of **problem first, tool second**.

📘 Glossary + Best Practice Callouts

SOAP ACK Response
Acknowledgement message returned by an external system in response to a Salesforce Outbound Message. This response confirms successful receipt of the message. If not returned correctly, Salesforce will retry delivery for up to 24 hours. A valid ACK response typically includes an `<Ack>true</Ack>` element within a properly structured SOAP envelope.

Bayeux Protocol
A protocol used for publish-subscribe messaging over HTTP, enabling asynchronous, event-driven communication. It underlies Salesforce's Streaming, CDC, and Platform Events implementations via CometD.

CometD
A scalable web messaging library implementing the Bayeux protocol, used by Salesforce clients to subscribe to event channels (CDC, Platform Events, Streaming API).

PushTopic
A configuration in Salesforce used by the Streaming API to define which record changes should trigger events. Built using a SOQL query to filter relevant data changes.

Generic Streaming
A variant of the Streaming API that lets you define custom event channels without relying on object-specific SOQL filters.

📘 Glossary + Best Practice Callouts

Runtime Data vs Metadata

- **Runtime Data** refers to actual Salesforce records like Accounts, Leads, Contacts, etc. Accessed via REST, Bulk, or SOAP API.
- **Metadata** refers to the structure and configuration of the org — like objects, fields, flows, or page layouts. Managed using Metadata API or Tooling API.

Small Batch (Composite API)
A practical limit of up to 25 subrequests in a single Composite API call, with a total payload of ~6MB. Ideal for inserting small groups of related records (e.g., Account + Contacts) using reference chaining.

Composite API
Salesforce REST API feature that allows multiple requests in a single HTTP call. Includes endpoints like /composite, /composite/tree, and /composite/batch. Enables atomic operations with parent-child dependencies using referenceId.

Upsert
A combination of update and insert. If a record with a given Id or External ID exists, it is updated. If not, a new record is created.

Back to Top

📘 Glossary + Best Practice Callouts

Complex Upsert
Upserts that involve dependent or related records, such as parent-child inserts, conditional logic, or multi-step upsert flows. Not supported in Bulk API due to lack of reference chaining.

Best Practice
Use Composite API when:

- You need to insert related records in one call
  You want atomic (all-or-nothing) behavior
  You're working with low-volume, high-integrity data operations

Transform Logic
Composite API does not allow field mapping or value transformation inside the request. You must handle this in:

- Middleware (e.g., MuleSoft using DataWeave)
  Client-side logic
- Pre-processing in Apex (if triggered from Salesforce)

Back to Top

Salesforce API Use Case Decision Matrix (v1)

Use Case	Recommended API/Pattern	Why This Works	Notes / Watchouts
Sync external CRM leads into Salesforce	**REST API**	Lightweight, real-time, supports JSON	Use Composite if sending leads + related contacts in one call
Import 100k+ records from ERP system (Products, Orders)	**Bulk API v2**	Handles large volumes asynchronously	v2 is simpler, auto-chunks data, use CSV for performance
Salesforce pushes new Opportunities to an external system	**Platform Events**	Decouples systems, supports async delivery	Subscribers must be online or use durable replay
Keep external data warehouse updated with Salesforce changes	**Change Data Capture (CDC)**	Sends changes (Create/Update/Delete) as events	Great for audit trails or synced reporting layers

Salesforce API Use Case Decision Matrix (v1)

Use Case	Recommended API/Pattern	Why This Works	Notes / Watchouts
Call external address validation service on record save	**Apex HTTP Callout** + **Flow Orchestration**	Real-time response needed before commit	Use Queueable if async is acceptable
Orchestrate multi-step process across systems (e.g. Order > Fulfillment > Billing)	**Middleware** or **Orchestrator + Platform Events**	Tracks process state, retries, failure handling	MuleSoft, Boomi, or Flow Orchestrator preferred for complex chains
Admin wants to automate nightly sync of pricing data	**Scheduled Flow** or **Scheduled Apex + Callout**	Declarative-first approach for recurring jobs	Use Named Credentials for authentication
Send real-time alerts to Slack when Case escalated	**Outbound Messaging** or **Platform Events**	Lightweight, fast, decoupled	Outbound Messaging is SOAP only; Slack needs webhook adapter

Salesforce API Use Case Decision Matrix (v1)

Use Case	Recommended API/Pattern	Why This Works	Notes / Watchouts
Allow external app to upsert Contacts	**REST API** (with upsert key)	Simple, secure if scoped	Use Named Credential + External ID field
Provide external vendors with read-only access to Salesforce data	**Salesforce Connect** or **Custom API Gateway**	Real-time access without data replication	Salesforce Connect works best with OData/SQL-compatible sources
Mass update Contacts based on external campaign results	**Batch Apex** or **Bulk API**	Handles transformation + volume efficiently	Use External IDs to match records safely
Provide a flexible, client-defined data fetch pattern (query and fields on demand)	**GraphQL API**	Declarative, client-controlled query structure	Ideal for front-end apps, requires precise query discipline

API-by-API Deep Dive

📘 API-by-API Deep Dive — Starting with REST API

🔹 What It Is

The Salesforce REST API is a lightweight, HTTP-based interface that allows external systems to interact with Salesforce using standard web protocols. It supports CRUD operations on records, metadata access, and simple query execution.

🔹 Best Use Cases

- Real-time integrations
  Single record or small batch processing
- Mobile or web apps calling Salesforce
- Rapid prototyping or testing (e.g., Postman)

🔹 Real-World Example

An external web form sends lead data to Salesforce via REST API upon submission, and receives the Salesforce ID in return.

API-by-API Deep Dive

🔹 **Strengths & Limitations**

✅ **Strengths:**

- Easy to use and understand
  JSON format, ideal for web services
  Full support for standard/custom objects, queries, and metadata

⚠️ **Limitations:**

- Not optimized for high-volume data (use Bulk API instead)
- Subject to strict governor limits per transaction
- Limited transactional behavior (no auto rollback across requests)

🔹 **Integration Design Tips**

- Use Upsert with External IDs to reduce pre-checks
  Use the /composite endpoint when inserting multiple related records
  Always implement retry logic and error handling

🔹 **Troubleshooting Tips**

- 401 Unauthorized? Check your access token
  400 Bad Request? Validate data types and field names

API-by-API Deep Dive — Bulk API vs Composite API

🔹 **Bulk API**

What It Is:
Bulk API is built for processing massive volumes of records asynchronously. It's optimized for performance using CSV files and server-side chunking (especially in Bulk API v2).

Best Use Cases:

- Large-scale data migrations
- Scheduled ETL pipelines
- Data lake → Salesforce ingestion

Strengths:

- Handles millions of records efficiently
- Supports retry and monitoring
  Bulk v2 simplifies job management

Limitations:

- No parent-child record creation
- No cross-record logic or referencing
  No immediate ID return (asynchronous only)

API-by-API Deep Dive — Bulk API vs Composite API

🔹 **Composite API**

What It Is:
Composite API allows multiple REST calls in one HTTP request, supporting reference IDs and record chaining. Great for maintaining data integrity in small payloads.

Best Use Cases:

- Creating related records in a single request
  Submitting small batches of upserts with dependencies
  Real-time web or mobile app sync

Strengths:

- Reference chaining across subrequests
  All-or-nothing atomicity
  Reduces API call count

Limitations:

- Max 25 subrequests per call
  Not suitable for large volumes
  No server-side logic or branching allowed

API-by-API Deep Dive cont.

🔹 **Tooling API**

What It Is:
Enables developer tooling to access org metadata (Apex, triggers, debug logs) programmatically.

Best Use Cases:

- Querying Apex Class metadata
  CI/CD pipeline diffs
- Automated code analysis

Strengths:

- Fast, lightweight
  Works great with IDEs and scripts

Limitations:

- Only a **subset** of metadata types supported
- No full deploy support (use Metadata API)

Sample Endpoint:
/services/data/vXX.0/tooling/query/?q=SELECT+Name+FROM+ApexClass

API-by-API Deep Dive cont.

🔹 **Metadata API**

What It Is:
Enables **deployments, backups, and config syncs** using XML-based packaging.

Best Use Cases:

- Deploy changes between sandboxes/orgs
- Git-based source control
- Metadata snapshot/versioning

Strengths:

- Full coverage of declarative components
  Works with DevOps tools (SFDX, Copado, Gearset)

Limitations:

- Not real-time
  Slower and more complex than Tooling API

Typical Flow:

1. Retrieve → 2. Store → 3. Validate → 4. Deploy

API-by-API Deep Dive cont.

🔹 **Change Data Capture (CDC)**

What It Is:
Sends real-time events when records are created, updated, deleted, or undeleted.

Best Use Cases:

- Sync Salesforce → data lake
  Trigger updates in external systems
- Compliance/audit tracking

Strengths:

- Delivers changed fields + replay ID
- Works with all standard/custom objects
  Real-time stream, scalable

Limitations:

- Requires middleware listener (CometD)
- Events expire after 72 hrs unless stored

Sample Payload:
Includes Change Event Header + modified fields.

API-by-API Deep Dive cont.

🔹 **Platform Events**

What It Is:
Custom-defined async events for business processes (not tied to record CRUD).

Best Use Cases:

- Publish Order Created events
  Trigger billing/invoicing downstream
  Decouple Salesforce from external chains

Strengths:

- Custom payload schema
  Supports Apex, Flow, external CometD clients
  Real-time and replayable

Limitations:

- Short retention (1–3 days)
  Must be explicitly published

API-by-API Deep Dive cont.

🔹 **Streaming API**

What It Is:
Legacy real-time push notifications for filtered record updates using **PushTopics** or **Generic Streaming**.

Best Use Cases:

- Live dashboard updates
  Monitoring record state transitions
- Web app integration

Strengths:

- Efficient event delivery
  Filter with SOQL
  Good for UI apps

Limitations:

- Requires open CometD connection
- No field-level granularity like CDC
  Rigid schema — requires re-creation on change

API-by-API Deep Dive cont.

🔹 **SOAP API**

What It Is:
Standards-based, XML-heavy API for systems requiring WSDL-based contract (e.g., Java, .NET).

Best Use Cases:

- Legacy systems
  Enterprise WSDL for strict schemas
  Long-lived sessions with login()

Strengths:

- Strong typing via WSDL
- Full CRUD + metadata access
  Works with WS-Security

Limitations:

- Verbose XML
- Client stubs need regeneration if schema changes

WSDL Types:

- **Enterprise WSDL:** Strongly typed per-org

API-by-API Deep Dive cont.

🔹 **Outbound Messaging**

What It Is:
No-code SOAP notifications triggered by Workflow Rule criteria.

Best Use Cases:

- Notify middleware or 3rd-party on Case creation
  Trigger basic external workflows
  Systems requiring ACK responses

Strengths:

- Retries up to 24h
- No Apex required
  Reliable delivery (if ACKed)

Limitations:

- SOAP only
  Payload can't be transformed
  No support for record hierarchy

API-by-API Deep Dive cont.

🔹 **Salesforce Connect**

What It Is:
Lets Salesforce **display** external system data using **External Objects** — no import/export required.

Best Use Cases:

- Real-time visibility of SAP or legacy ERP data
- Avoiding data duplication
  Vendor catalog or inventory views

Strengths:

- No storage footprint in Salesforce
  OData/SQL connectors supported
  Supports lookups, layouts, reporting

Limitations:

- Slower UX (depends on external system)
- Limited write support
  Not for high-volume reporting

API-by-API Deep Dive cont.

🔹 **GraphQL API**

What It Is:
Salesforce's new API offering allowing clients to query data using **GraphQL** syntax — flexible, powerful, and front-end friendly.

Best Use Cases:

- Headless UI apps
- Dynamic queries from front-end
  Fetch only fields you need — no over-fetching

Strengths:

- Fully declarative queries
- Flexible nesting and filters
- Single endpoint: /services/data/vXX.0/graphql

Limitations:

- Requires strict query validation
  Not designed for bulk loads or DML

API-by-API Deep Dive — Bulk API vs Composite API

When to Use Which?

Scenario	Use Bulk API	Use Composite API
Migrating 100k+ records	✅ Yes	❌ No
Inserting parent and related children	❌ No	✅ Yes
Real-time response needed	✅ Yes	❌ No
Reference ID chaining required	❌ No	✅ Yes
Mass nightly upsert by external system	✅ Yes	❌ No
Conditional logic across records	❌ No	❌ Use Apex or Middleware

Comparison Table: Composite API vs Apex vs Bulk API vs Middleware

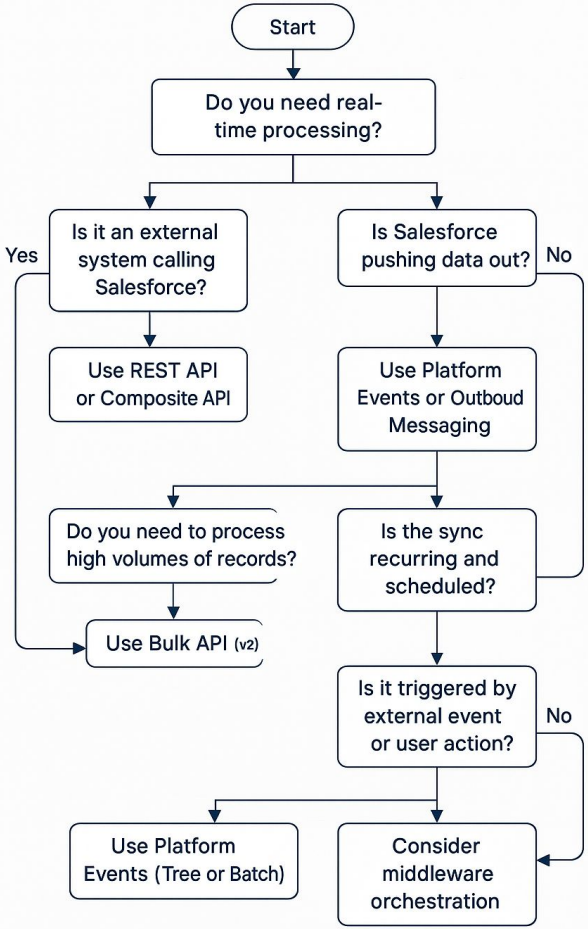
Capability	Composite API	Apex	Bulk API	Middleware (e.g., MuleSoft)
Multi-object creation	✅	✅	❌	✅
Upsert with reference	✅	✅	❌	✅
Record volume handling	⚠️ (25 subrequests)	✅ (Batchable)	✅ (Millions)	✅
Complex logic / conditionals	❌	✅	❌	✅
Transformation support	❌	✅	❌	✅ (via DataWeave, etc.)
Retry / error management	⚠️ Minimal	✅	✅	✅
Real-time capable	✅	✅	❌	✅ (depends on orchestration)
Easily monitored/debugged	✅	✅ (Debug Logs)	✅ (Monitoring UI)	✅ (Dashboards, Alerts)

Comparison Table: Composite API vs Apex vs Bulk API vs Middleware

🧠 **Notes:**

- Composite API is fast, but **not flexible** — use for atomic reference-based insertions.
- Apex gives control, but you **own error handling** and scalability logic.
- Bulk API is **volume king**, but not relationship-friendly.
- Middleware is best for **orchestration, retry queues, error routing**, and transforms.

API Decision Tree Diagram



Architecture Diagram