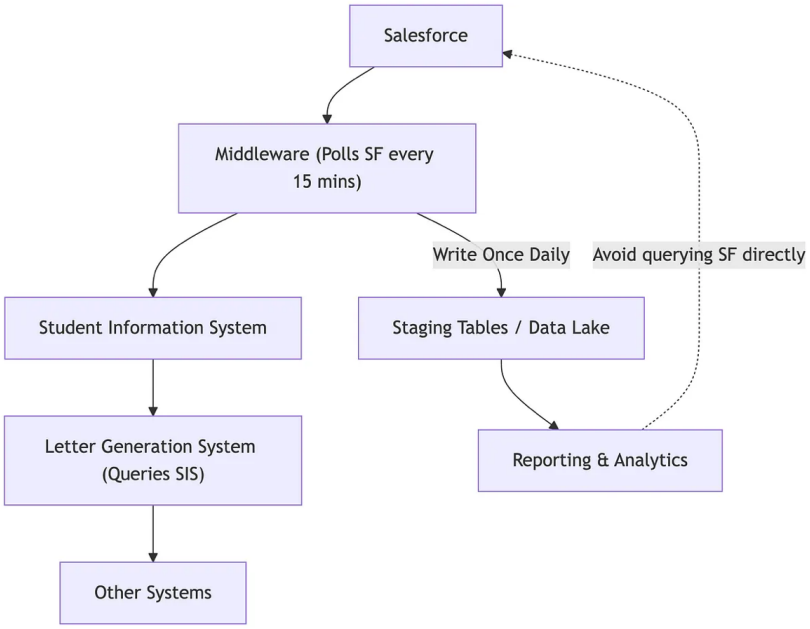
How We Built a Real-Time Integration Pipeline in Salesforce Using Platform Events, Channels, and Subscriptions

Pranav Nagrecha 5 min read · May 14, 2025

Say goodbye to 15-minute polling delays — here's how we reimagined SIS integration using event-driven architecture, native Salesforce tools, and Amazon EventBridge.

1. Background: The Problem We Were Solving



Our integration challenge began with a familiar scenario:

- Students submit applications via Salesforce Experience Cloud.
- Admission teams review and accept these applications within Salesforce.
- Accepted applications are then sent to the Student Information System (SIS).

Legacy model:

- The middleware polled Salesforce every 15 minutes.
- Queries ran at scheduled intervals, regardless of data changes.
- Applications accepted seconds after a poll were

missed for up to 15 minutes or lost entirely during high-load periods.

- This approach generated excessive API calls, introduced data inconsistencies, and risked duplicate or missing records.

We needed a solution that was:

- **Real-time:** event-driven rather than time-driven
- **Reliable:** guaranteed no dropped records or duplicates
- **Scalable:** easily extendable to additional systems (e.g., data lakes or notification services)

2. Designing the Real-Time Integration Pipeline

Before diving into implementation, it's important to understand the Salesforce-native components we used to architect this real-time solution:

1. What Are Platform Events?

Platform Events are Salesforce's publish-subscribe messaging feature. They let you define custom event schemas and publish events whenever specific changes occur in your org. External or internal systems can then subscribe to these events in real

time. Think of them as structured notifications that trigger integrations the moment something important happens — such as an application being accepted.

Key Traits:

- Defined in Setup: Configured like any custom object in Salesforce.
- Custom Fields: Support fields such as Student ID, Program, and Status.
- Publishable: Can be published from Apex, Flow, or the API.
- Subscribable: Consumable via CometD, the EMP Connector, or integrations like Amazon EventBridge.

How to create Platform Events — Trailhead & Youtube(must watch)

2. What Are Event Channels?

Event Channels are logical routes that group Platform Events and control external access. Instead of subscribing directly to `/event/Application_Approved__e`, you subscribe to a more flexible endpoint such as `/data/vXX.X/event/Admissions_Stream__chn`.

Why Use Channels?

- Unified Endpoint: Combine multiple Platform Events under a single URL.
- Traffic Management: Gain finer control over event throughput and access.

How to Create Event Channels — Salesforce Help & Youtube

3. What Are Channel Members?

Channel Members control who receives which events on a channel. Each member specifies:

- **Event Type:** the Platform Event to monitor
- **Filter Logic:** for example, `Program_Code = 'MBA'`
- **Consumer Credential:** the app or named credential authorized to consume the events

Benefits:

- **Targeted Delivery:** send different subsets of data to different subscribers
- **Noise Reduction:** filter out irrelevant events
- **Security Enforcement:** restrict access based on credentials and filters

How to create Channel Members — Youtube

Together, Platform Events, Channels, and Channel Members form the core of an **event-driven integration framework** in Salesforce.

3. Implementation Details

1. Publishing the Platform Event via Flow

We created a **Record-Triggered Flow** on the `Application` object:

- **Trigger:** After Update
- **Criteria:** `Status__c = 'Accepted'`
- **Action:** Create a new `Application_Accept__e` Platform Event with mapped fields:
- `Application_Id__c` `Student_Name__c` `Program_Code__c` `Start_Term__c` `Decision_Date__c` `Status__c`

2. Routing Through an Event Channel

To enable Amazon EventBridge to subscribe to events, Salesforce requires the use of an Event Channel and Event Relays.

We created:

- **Event Channel Name:** `ApplicationAccept__chn`

- **Platform Event Mapped:** `Application_Accept__e`
- **Event Relay:** This is our connection to AWS (Setup on AWS)

> Note: Salesforce Event Relay requires an Event Channel. It does not work with raw Platform Event endpoints and channel member endpoints like `/event/Application_Accept__e`

We also created a **Named Credential** that authenticates the Partner Event Source connection from AWS to Salesforce.

3. AWS EventBridge

In Amazon EventBridge:

1. We configured a **Partner Event Source** from Salesforce.

2. Accepted the event source in AWS Console.
3. Created an **EventBridge Rule** to send all events into an **Amazon SQS Queue**: `AdmissionsUpdateQueue`.

Why SQS?

- Boomi (Our middlewear) still processes SIS updates, but now consumes them from SQS instead of polling Salesforce.

SQS gives:

- Guaranteed delivery
- Buffering and retry support
- Controlled pace for downstream jobs

Adding SNS for Real-Time Notification Triggers

To make this pipeline even more extensible, we added a second EventBridge rule that routes events to an SNS Topic: `AdmissionsAlertTopic`.

This SNS topic supports:

- Real-time generation of admission confirmation packets
- Triggering email alerts to advisors or internal

teams

SNS is push-based and multi-subscriber — perfect for alerting, monitoring, or feeding downstream async systems like analytics or PDF generation.

4. Channel Members for Internal Consumers

We also created Salesforce **Channel Members** for internal systems:

- **Filtered Member:**
- Filter: `Program_Code__c = 'MBA'`
- Target: Internal Data Lake API
- **Unfiltered Member:**
- Target: Audit Logging Service

This allows internal Salesforce-connected systems to selectively consume events without going through AWS.

These were configured via Tooling API or Metadata API (not in Setup UI).

4. Lessons Learned

Implementing a real-time integration pipeline between Salesforce and SIS using Platform Events taught us far more than just technical architecture. Here are the key takeaways:

1. Design with the Consumer in Mind — Publishing events is only half the job — understand how each consumer (AWS, SIS, Data Lake, Audit, etc.) processes the data. Their latency tolerance, failure handling, and schema expectations all shape how you build your events.
2. Avoid Over-publishing — Just because Platform Events are easy to fire doesn't mean they should be triggered everywhere. Keep publishing logic tight and condition-driven to avoid flooding downstream systems with noise.
3. Use Event Channels Strategically — Salesforce Channels aren't just a technical requirement — they give you centralized routing, better governance, and clarity across integration points. Document what each channel is responsible for.
4. Channel Members Are Powerful for Internal Consumers — Channel Members allow filtered, secure data delivery without writing extra Flows or Apex. This is especially useful when multiple internal systems need only subsets of event data.

Salesforce | Integration | Data | Analytics | Amazon Web Services

Written by Pranav Nagrecha

Edit profile

51 followers · 43 following

Salesforce Administrator/Developer/Architect/Business Analyst/QA

No responses yet

Pranav Nagrecha he

What are your thoughts?

More from Pranav Nagrecha

Pranav Nagrecha

Boomi—How to convert Timezones in a Process

I use Google to the extreme, I know there are smarter peopl...

May 17, 2022

Pranav Nagrecha

Stop Guessing. Start Designing: Order of...

Your save isn't haunted—it's a kitchen line: before-save,...

Aug 26 2

Pranav Nagrecha

Roles, Profiles, and Permission Sets in...

Introduction

Aug 21 19

Pranav Nagrecha

Flow Control: Navigating Salesforce...

Flows are the lifeblood of many business processes. Bu...

Sep 25, 2023 9

See all from Pranav Nagrecha

Recommended from Medium

Vaishnavi R

Database. Stateful in Salesforce

If you have ever worked with batch classes in Apex, you...

Jul 23 2

Musa Ndlala

Understanding Standard Objects and...

💭 The Moment You Realize Salesforce Isn't Just Data—It'...

Sep 15 5

Mani

Optimizing Long-Running Callouts in...

Jun 24 1

Pedro Távora Santos

Why Your Agentforce POC Will Fail: An...

If you're reading this, you've probably sat through the...

Nov 18

Raja Saha

Generating Unique Identifiers in...

In Salesforce, unique identifiers play a vital role in...

Jun 30

Jeremy Carmona

How I Use Salesforce Templates to Save...

Third in the "Salesforce, Simplified" series

Jun 12 12.1K 22

See more recommendations