International Journal of Multidisciplinary Research and Growth Evaluation
ISSN: 2582-7138
Received: 02-04-2020; Accepted: 18-04-2020
www.allmultidisciplinaryjournal.com
Volume 1; Issue 1; March-April 2020; Page No. 76-80

Salesforce integration best practices: A complete guide

Laxman Vattam [1], Kalpana Puli
[1] Independent Researcher, Washington, USA
[2] Independent Researcher, Texas, USA

Corresponding Author: **Laxman Vattam**

DOI: https://doi.org/10.54660/.IJMRGE.2020.1.1-76-80

Abstract
The growing need to connect data from various sources is essential for ensuring scalability, security, data consistency, and the long-term maintainability of an organization's technology infrastructure. Salesforce cannot act as the definitive source of truth for all data. The true strength of Salesforce lies in its capacity to integrate effortlessly with other systems, forming a cohesive business ecosystem. This guide outlines best practices for Salesforce integration, helping ensure your systems collaborate seamlessly and operate efficiently.

Keywords: Salesforce, integrations, platform events, rest API, Bulk API, SOAP API, External Objects, Long Polling

Introduction
If you're planning to adopt Salesforce-based applications or the Lightning Platform at scale, it's essential to familiarize yourself with the integration capabilities and options available. IT Team (Salesforce Architects and developers) should carefully consider these integration patterns and best practices during the design and implementation stages of any Salesforce integration project. Salesforces own consulting architects rely on these patterns during architectural reviews and are actively involved in refining and enhancing them. When implemented correctly, these patterns enable a rapid path to production, ensuring a stable, scalable, and low-maintenance set of applications.
While these patterns cover most integration scenarios, they don't encompass every possible use case. For instance, UI integrations, such as mashups, are outside the scope of this guide.

Pattern Selection Guide
The patterns are categorized using these dimensions.

- **Timing**

Synchronous—Blocking or near-real-time requests are request/response operations. The result is returned to the caller immediately via this operation.
Asynchronous—Nonblocking, queue, or message-based requests are invoked by a one-way operation. The results and any faults are returned by invoking other one-way operations. The caller therefore makes the request and continues without waiting for a response.

- **Type**

Process – Process-based integrations involve synchronizing the functional flow between two or more applications.
Data – Data integrations focus on connecting and exchanging the information used by different applications.
Virtual – Virtual integrations allow Salesforce to interact with data stored in an external system without the need to replicate that data within Salesforce.

- **Integrating Salesforce with Another System**

This table outlines the integration patterns and their key characteristics to help you identify the most suitable pattern for your needs when integrating Salesforce with another system.

Type	Timing	Key Pattern to Consider
Process Integration	Synchronous	Remote Process Invocation–Request and Reply
Process Integration	Asynchronous	Remote Process Invocation–Fire and Forget
Data Integration	Synchronous	Remote Process Invocation–Request and Reply
Data Integration	Asynchronous	UI Update Based on Data Changes
Virtual Integration	Synchronous	Data Virtualization

Fig 1: Patterns [1]

- **Integrating Another System with Salesforce**

This table outlines the integration patterns and their key characteristics to help you identify the most suitable pattern for your needs when integrating another system with Salesforce.

Type	Timing	Key Pattern to Consider
Process Integration	Synchronous	Remote Call-In
Process Integration	Asynchronous	Remote Call-In
Data Integration	Synchronous	Remote Call-In
Data Integration	Asynchronous	Batch Data Synchronization

Fig 2: Integrations [1]

Salesforce Integration Patterns

- **Request and Reply Pattern**

  The request and reply pattern in Salesforce is a powerful approach for integrating Salesforce with external systems. When an event occurs in Salesforce, this pattern facilitates triggering a process in an external system, transmitting the necessary data, receiving a response, and using that response to update Salesforce. It provides a structured mechanism for bi-directional communication, ensuring seamless data flow and process synchronization between systems.

When implementing the request and reply pattern, it is essential to choose the appropriate method based on the specific use case. External Services are ideal for simple, schema-driven integrations, while Lightning Components and Visualforce offer greater flexibility for custom UI-driven interactions. Triggers are effective for asynchronous, event-driven processes, and Batch Apex is best for large-scale operations. By carefully selecting and optimizing the integration approach, businesses can ensure secure, efficient, and reliable communication between Salesforce and external systems.

Triggers can be configured to make callouts to external systems when specific data changes occur in Salesforce. However, these callouts must be asynchronous, as Salesforce does not support synchronous callouts within triggers. While this approach is not well-suited for request-response interactions, it works effectively for fire-and-forget scenarios where a response is not immediately required. This makes it ideal for initiating processes or notifications in external systems without disrupting Salesforce operations.

Salesforce supports integration through Lightning Components or Visualforce pages by consuming WSDL files and generating proxy classes. These tools enable Salesforce to interact with external systems using SOAP or REST-based web services. Additionally, HTTP services are supported to perform operations such as GET, POST, PUT, or DELETE. This method is highly flexible and allows users to initiate external system interactions directly from custom user interfaces, making it ideal for real-time data exchange scenarios.

Salesforce's External Services feature allows for integration with external systems using point-and-click tools within Lightning Flow. This method requires the external system to provide an OpenAPI or Interagent schema to define the endpoints and operations. External Services support only primitive data types, making them suitable for simpler integrations where complex data structures are unnecessary. This approach empowers administrators and non-developers to build integrations without writing extensive code.

- **Fire-and-Forget Pattern**

  The fire-and-forget pattern in Salesforce is designed for scenarios where an event in Salesforce triggers a process in a remote system without requiring a response. This approach is suitable for asynchronous operations, ensuring that Salesforce processes continue seamlessly while initiating the desired external action. The fire-and-forget pattern is efficient for integrating loosely coupled systems and is commonly used for notifications or process initiation in external platforms.

Apex-based callouts offer a highly flexible method for fire-and-forget scenarios. Developers can write Apex code to make HTTP requests (e.g., POST or PUT) to remote systems when specific events occur in Salesforce. This approach is particularly useful for integrating with RESTful APIs or complex external services. Since these callouts are asynchronous, they do not wait for a response, ensuring minimal impact on Salesforce performance while triggering the external process.

Platform events provide a robust mechanism for achieving fire-and-forget integrations in Salesforce. In this method, an event is published when a specific process occurs, such as record creation or update. External systems subscribed to the event can consume the published data and initiate the necessary actions. Process Builder or Lightning Flow can be used to configure and automate these events, enabling administrators to create integrations without code. This method ensures scalability and real-time communication between systems.

For more complex scenarios, custom platform events can be designed to accommodate specific data structures and requirements. Developers can define custom events using the Salesforce schema and write Apex triggers to publish these events based on specific conditions. External systems can then subscribe to these events through APIs, ensuring they receive the required information. Customization-driven platform events are particularly useful for handling intricate data relationships and tailored workflows.

Outbound messages, configured through Salesforce workflows, provide a straightforward way to send data to

external systems. When a workflow rule is triggered, Salesforce sends an XML message to a specified endpoint, passing the necessary information. This method is easy to set up and does not require coding, making it ideal for simpler integrations. However, it requires the external system to be capable of consuming the XML message and processing it appropriately.

Selecting the appropriate fire-and-forget method depends on the complexity of the integration, the required level of customization, and the external system's capabilities. Platform events are ideal for scalable, real-time communication, while outbound messages suit simpler, pre-defined integrations. Apex-based callouts offer the greatest flexibility for complex scenarios. By understanding the unique requirements of each use case, businesses can effectively implement the fire-and-forget pattern to enhance their Salesforce integrations.

- **Batch Data Synchronization**

Batch data synchronization is a strategy for keeping data consistent between Salesforce and external systems by processing updates in batches. This method is particularly effective for scenarios involving large datasets or periodic updates. By synchronizing data in a structured and efficient manner, businesses can ensure seamless communication and data integrity across systems while minimizing performance impacts.

Change Data Capture (CDC) is a Salesforce feature that publishes change events to reflect record modifications, such as create, update, delete, or undelete actions. External systems subscribed to these change events can process the updates and synchronize their data accordingly. CDC is well-suited for real-time synchronization with minimal coding and allows organizations to track changes efficiently. It is especially valuable for maintaining data consistency in complex integration scenarios.

ETL (Extract, Transform, Load) tools play a central role in batch data synchronization by managing data movement between Salesforce and external systems. These tools extract data from the source system, transform it into the required format, and load it into the target system using Salesforce's Bulk API or SOAP API. ETL tools are highly scalable, allowing businesses to handle large data volumes effectively. This method is ideal for periodic or scheduled data synchronization tasks where real-time updates are not required.

In this approach, Salesforce and external systems make direct calls to each other whenever data changes. While this method provides flexibility, it can generate excessive traffic and lead to performance bottlenecks, especially when dealing with frequent updates or high data volumes. As such, manual remote calls are generally not recommended for batch data synchronization unless used in limited or highly specific scenarios.

To maximize the efficiency of batch data synchronization, businesses should prioritize methods that balance performance and reliability. Change Data Capture is ideal for real-time updates with minimal overhead, while ETL tools are best suited for periodic synchronization tasks involving large datasets. Manual remote calls should be used sparingly to avoid unnecessary traffic. By carefully evaluating the integration requirements, businesses can implement a synchronization strategy that meets their operational needs without compromising system performance.

Batch data synchronization ensures data consistency across systems, reduces manual data entry errors, and streamlines operations. By using Salesforce features like CDC and leveraging ETL tools, organizations can achieve reliable and scalable data synchronization. This capability supports better decision-making, improves customer experiences, and enhances the overall efficiency of integrated business processes.

Batch Apex provides a robust mechanism for making callouts to external systems, particularly for large datasets or bulk operations. The execute method within Batch Apex resets governor limits for each batch, enabling a higher number of callouts compared to standard operations. However, there are still limitations on the total number of callouts or the maximum duration of callouts within a single transaction. This approach is suitable for processing extensive data sets while maintaining adherence to Salesforce's governor limits.

- **Remote Call-In**

If data in Salesforce needs to be created, retrieved, updated or deleted by a remote system, we can consider this pattern

- SOAP & Rest API
- Apex based APIs

- **UI Update Based on Data Changes**

When an event occurs in Salesforce, if the user needs to be notified in the Salesforce user interface without having to refresh their screen and potentially losing work, this pattern can be considered.

Recommended solution is to use PushTopic with a query definition that allows you to:

- Specify what events trigger an update
- Select what data to include in the notification

- **Data Virtualization**

Salesforce accesses external data in real-time, eliminating the need to store data within Salesforce and reconcile it with the external system.

Use Salesforce Connect to access data from external sources, along with the Salesforce data. We can pull data from legacy systems such as SAP, Microsoft, and Oracle in real time without making a copy of the data in Salesforce.

Authorize Apps with OAuth

OAuth is an open standard protocol that enables a client application to securely access data from a protected resource by exchanging tokens. These tokens serve as authorizations, granting specific permissions to the client application.

1. OAuth Authorization Flows

OAuth authorization flows provide client applications with limited access to protected resources on a resource server. Each flow follows a unique process for granting access, but they generally involve three key steps. First, the client application requests access to a protected resource. Next, the authorization server issues access tokens to the client application in response. Finally, the resource server verifies the validity of these tokens and permits access to the protected resource.

- **OAuth 2.0 Web Server Flow**

  This flow allows external web apps to integrate with the Salesforce API using the OAuth 2.0 authorization code grant type. The server hosting the web app must secure the connected app's identity through the client ID and

client secret.

- **OAuth 2.0 User-Agent Flow**
  The user-agent flow enables users to authorize desktop or mobile apps to access data via an external or embedded browser. It is suitable for client apps running within a browser, such as those using JavaScript. This flow follows the OAuth 2.0 implicit grant type.
- **OAuth 2.0 Refresh Token Flow**
  The refresh token flow renews access tokens issued by either the OAuth 2.0 web server flow or the OAuth 2.0 user-agent flow.
- **OAuth 2.0 Token Exchange Flow**
  For architectures involving Salesforce and a central identity provider, the token exchange flow simplifies integration. It allows the exchange of tokens from external identity providers for Salesforce tokens, enabling access to Salesforce data.
- **OAuth 2.0 for Hybrid Apps**
  Hybrid apps often require complex session management. OAuth 2.0 hybrid app flows connect access and refresh tokens with web sessions, simplifying token tracking and session management for hybrid apps, unlike typical user-agent or refresh token flows.
- **OAuth 2.0 JWT Bearer Flow**
  The JWT bearer flow authorizes servers to access data without requiring interactive login. It uses a certificate to sign the JWT request and eliminates the need for user interaction, though prior client app approval is required.
- **OAuth 2.0 Client Credentials Flow**
  The client credentials flow facilitates direct data sharing between applications without user interaction. The client app exchanges its credentials (consumer key and secret) for an access token. It requires specifying an integration user and is a more secure alternative to the username-password flow.
- **OAuth 2.0 Device Flow**
  The device flow is designed for apps running on devices with limited input or display capabilities, such as IoT devices, Smart TVs, or command-line apps. Users can connect these devices to Salesforce by accessing a browser on a more advanced device.
- **OAuth 2.0 Asset Token Flow**
  This flow integrates IoT devices with the Salesforce API using asset tokens, which are JWT authentication tokens that secure and verify device requests. They link devices to Salesforce CRM data related to customers, accounts, or contacts.
- **Demo the Asset Token Flow**
  The Asset Token Explorer demo app simplifies the process of acquiring an access token and exchanging it for an asset token.
- **OAuth 2.0 Username-Password Flow**
  While the username-password flow allows authorization via a connected app using user credentials, it's generally not recommended due to security concerns with transmitting credentials. It's best used when there is a high degree of trust, and alternative grant types are unavailable.
- **OAuth 2.0 SAML Bearer Assertion Flow**
  In this flow, a client can use a signed SAML 2.0 assertion to request an OAuth access token. The digital signature authenticates the connected app, and the SAML assertion, issued by an identity provider, identifies the subject for security purposes.
- **SAML Assertion Flow**
  For organizations using SAML to access Salesforce, the SAML assertion flow provides a way to federate with the API, similar to how they federate with Salesforce for Web SSO. This flow can be used without a connected app.

2. OAuth Tokens and Scopes

OAuth tokens grant access to protected resources, allowing connected apps to receive tokens on behalf of a client after successful authorization. Scopes define the specific types of resources that the connected app is permitted to access. These scopes are assigned during the creation of the connected app and are included with the OAuth tokens as part of the authorization flow.

- **Authorization code**: The authorization server creates an authorization code, which is a short-lived token, and passes it to the client after successful authentication. The client sends the authorization code to the authorization server to obtain an access token and, optionally, a refresh token.
- **Access token:** After a client is authorized, Salesforce sends the client an access token. The client passes the access token to the resource server to request access to protected resources. The resource server validates the access token and additional permissions in the form of scopes before granting access to the client. The access token has a longer lifetime than the authorization code, usually minutes or hours. When an access token expires, attempts to use it fail, and the client must obtain a new access token by using a refresh token or reinitiating the authorization flow.
- **Refresh token**: Like a password, a refresh token can be used repeatedly by a client to gain access to the resource server. When a refresh token expires or a user revokes it outside of the client, the client requests a new access token, typically by implementing the authorization flow from the start. A refresh token can have an indefinite lifetime, persisting for an admin-configured interval or until explicitly revoked. The client can store a refresh token and use it to obtain new access tokens. For security, the client must protect a refresh token against unauthorized access.

Conclusion

The patterns above address the requirement to synchronize data that resides across two or more systems so that both systems always contain timely and meaningful information. When designing your integration approach, assess the complexity of your use cases and requirements as a business and tailor it specifically. Ensure data security compliance and comply with industry regulations during integration. Select an integration approach which.h can scale with future business growth. Implement trusted testing procedures in order to detect issues before they disrupt operations; Implement trusted testing procedures.

References

1. MuleSoft. Synchronous vs. Asynchronous Communication in Applications Integration. MuleSoft.com. [Online]. Available: https://www.mulesoft.com/resources/esb/applications-integration. [Accessed Nov. 22, 2019].

2. Microsoft Corporation. Integration Patterns (Patterns & Practices). Redmond, WA, USA: Microsoft Press; 2004. [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=12474. [Accessed Dec. 13, 2019].
3. Enterprise Integration Patterns. Hub and Spoke [or] Zen and the Art of Message Broker Maintenance. [Online]. Available: http://www.eaipatterns.com/ramblings/03_hubandspoke.html. [Accessed Dec. 1, 2019].
4. Enterprise Integration Patterns. Messaging Patterns. [Online]. Available: https://www.enterpriseintegrationpatterns.com/patterns/messaging/. [Accessed Dec. 13, 2019].
5. Salesforce. Integration Pattern Summary. [Online]. Available: https://developer.salesforce.com/docs/atlas.en-us.214.0.integration_patterns_and_practices.meta/integration_patterns_and_practices/integ_pat_pat_summary.htm?_ga=2.159182394.1420426287.1736288671-1033462729.1736288671. [Accessed Jan. 11, 2020].
6. Salesforce. REST API. [Online]. Available: https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/intro_what_is_rest_api.htm. [Accessed Feb. 23, 2020].
7. Salesforce. Platform Events. [Online]. Available: https://developer.salesforce.com/docs/atlas.en-us.platform_events.meta/platform_events/platform_events_intro.htm. [Accessed Mar. 25, 2020].