Cross-System Data Integration — Custom Component on Litify (Salesforce)

Business Analyst case study: bridging a custom Salesforce component with a law firm's external processing servers

Business Analysis | **Salesforce / Litify** | **System Integration** | **XML/JSON Data Mapping**

Overview

The client, a law firm, runs its legal case management, data, and operations on Litify — the standard Salesforce managed package used across the legal industry. The engagement scope was to build a custom Salesforce component within the client's Litify org that could capture case data and attachments (up to 10MB per file), and reliably hand that data off to the client's own external servers for downstream processing.

The Challenge

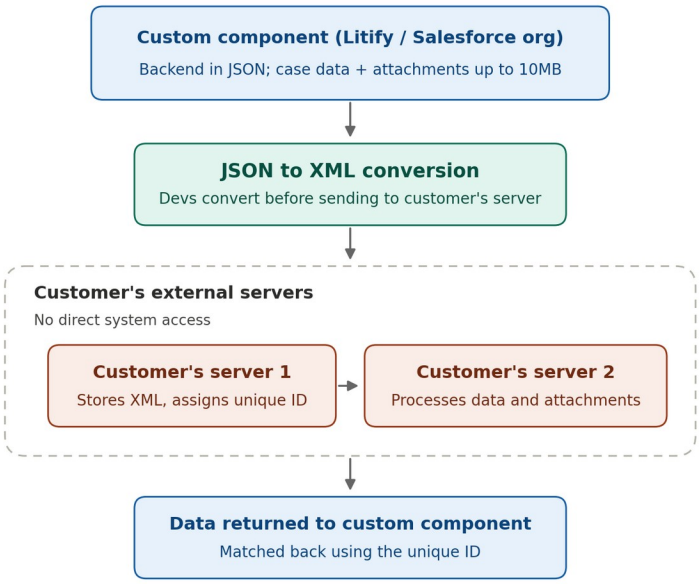
- The custom component's backend logic was written natively in JSON, but the client's external servers would only accept data in XML.
- The client's two external servers handled all downstream data and attachment processing, but the project team had no direct access to either system.
- Data had to travel one-directionally through two external systems before returning to the Salesforce component — with no way to query either server directly to confirm status.
- The component needed to reliably carry file attachments up to 10MB alongside the structured case data, without loss across the hand-off.

My Role

I acted as the functional liaison between the client's technical stakeholders and the development team. I elicited the client's API requirements and constraints, translated them into a clear technical specification the developers could build against, and coordinated the end-to-end data flow design so that every handoff between systems was explicit and testable.

The Solution

Working with the developers, we designed a conversion and hand-off layer that translated the custom component's native JSON output into the client's required XML format, tagging each payload with a unique identifier before it left Salesforce. That identifier became the thread connecting every step of a process we couldn't directly observe past the first hop:



- The custom component's backend generates case data and attachments (up to 10MB) natively in JSON.
- Developers convert the JSON payload to XML and tag it with a unique identifier before sending it to the client's first server.
- The first server hands the data and attachments to a second external server for processing.
- The component polls using the same unique identifier to detect when processed data becomes available on the second server, then retrieves and stores it back against the case record.

Outcome & Impact

0	10MB	2
Direct access to client servers — yet full data traceability	Max attachment size supported per case record	External servers bridged through one identifier-based contract

The identifier-based hand-off pattern gave the team a reliable way to confirm delivery and completion across systems neither side could directly inspect, without requiring any change to the client's existing external infrastructure. The approach became a reusable integration pattern for later phases of the engagement, and the structured requirement-elicitation process reduced rework across the broader implementation.

Skills Demonstrated

- Requirement elicitation and translation into functional specifications
- Cross-functional liaison between client stakeholders and engineering
- XML/JSON data flow mapping across disconnected systems
- Custom Salesforce component development within a Litify legal case management org
- Designing for systems with no direct visibility or access

Based on a Salesforce Business Analyst engagement, KCloud Technologies Pvt Ltd.