P2: Salesforce-NetSuite Integration

Background

- SF-NS Integration is a component of a larger effort to integrate Sales, Finance, Entitlements and Product Provisioning as outlined here.
- In parallel, there was an effort undertaken by Finance to simplify product offering (Item/SKU simplification) and migrate all products from the NetSuite Contract Renewals module to NetSuite SuiteBilling (NS's recurring billing solution).

Goals related to NS-SF Integration

- Create software / data-driven Sales function.
- Move quoting into the Sales flow (from Finance) and automate quote generation and signing.
- Automate Sales fulfillment, entitlements, and subsequent billing from SF to NS / SuiteBilling

Guiding Principles

- NetSuite / Suite Billing must be the master for customer / product / subscription data.
- Avoid bi-directional synchronization of the same data records as much as possible.
- Use stock systems and processes in SF and NS as much as possible.
- Any Sales-specific process needs must be implemented outside of NS, not as NS customizations.
- Added in March, 2021: Implement the NS-SF integration using the upcoming DevFlows product as it is strategically important to us.

Concepts

- An integration project using an iPaaS can be thought of as a layered stack, similar to the OSI layers used in networking.

<table>
<tr><td>Specific Workflow Logic
The operations necessary to connect data and workflows across systems</td><td>• Leading vendors have some basic templates that do simple mapping of a few basic objects SF-NS.
• Celigo is the only vendor that supports key SF-NS workflows, with some coverage for SuiteBilling and CPQ.
• There is a difference in how companies choose to use SF and NS that go outside of the default workflows that prevent using many (usually more complex) templates OOTB.</td></tr>
<tr><td>Generic App Connectors
- Salesforce Connector
- NetSuite Connector</td><td rowspan="2">• Required capabilities provided by many iPaaS vendors:
- Boomi,
- MuleSoft,
- Tray.io,
- Celigo,
- JitterBit,
- Zapier, etc</td></tr>
<tr><td>Generic iPaaS Pattern
- Data Transfer and Mapping
- Constructs for process logic</td></tr>
</table>

[click on the image for Lucidchart link. Alternatively: high-res image]

- **Layer 1** - is the generic iPaaS function which contains the components necessary to handle data marshalling and transformation, process logic components, and error handling.
  - There are many vendors with solid solutions. We have a fairly basic integration case and fundamentally, for our purposes, most major vendors have the required capabilities.
  - The difference is in the ease of use and how close the solution is to the upcoming DevFlows.

- **Layer 2** - App-specific Connectors that enable CRUD on records, events, and execution of the APIs.
  - Most leading vendors have connectors for both Salesforce and NetSuite.
  - The difference is in how the connectors are implemented.
    - Some tools like MuleSoft, Jitterbit, Tray.io, Boomi will operate via the APIs.
    - Celigo, requires installing software packages into both NetSuite and Salesforce that enables higher-level integration, and partially replicates SuiteBilling data model in SF.
- **Layer 3** - Specific logic that integrates workflows and data records across systems
  - Most vendors that support NS-SF offer a few basic 1:1 integration templates for simple objects (Accounts, Contacts, Items). These are very basic and can be used as a "how-to" reference, rather than an OOTB solution.
  - Celigo has the most extensive library of SF-NS templates that provide integrations for key workflows, including support for SuiteBilling and some support for CPQ. These templates assume Celigo's opinion on the default use of the respective systems.
  - In fact, there is variation in **how** companies (including Trilogy) choose to use these systems which prevents using these templates OOTB. For example:
    - In NS we are grouping all sellable items into separate Subscription Plans consisting of the product Item and the success plan Item. This simplifies revenue allocation and reporting (see IDD 6). Therefore, instead of NS Item -> SF Product sync (as done by Celio), we map NS Subscription Plan -> SF Product.
    - We want to use NS as the product catalog and pricing master, but be able to generate accurate amendment quotes in the standard SF flow. To do this, we use the SF CPQ package, which includes a Subscription object. Celigo template does not assume CPQ, and creates a custom Subscription object which cannot be directly used by the SF quoting workflow.
  - **This is where the time is spent when developing the Trilogy NS-SF connector.**

Choice of an iPaaS

- We spent 3 weeks working with Celigo in March-April and documented the NS-SF templates to learn how it handles the integration. There are 3 template groups available:
  - Default NetSuite - Salesforce (Customer, Contact, Items, etc)
  - SuiteBilling - Salesforce (Subscriptions - but creates custom object in SF instead of using CPQ Subscription)
  - Salesforce CPQ -> NetSuite (additional flows for Contract Renewals and Amendments)
  - We would not be able to use Celigo templates OOTB and would need to modify them.
- In addition to Tray.io, we also did a short POC using Boomi (referencing the past attempt to integrate NS and Central SF), and also evaluated MuleSoft (due to it now being a part of the Salesforce ecosystem). Detailed capability comparison can be found here.
- **All have the needed capabilities to do the integration, but none will be OOTB.**
- The decision was finally based on 2 key parameters:
  - Simplicity of use.
  - Similarity (from the user and capabilities standpoint) to the upcoming DevFlows.
- We would learn how to solve the Layer 3 issues using the simplest and most similar (in terms of capabilities) tool, making it easy to recreate the connector logic in DevFlows

Criteria	Celigo	Tray.io	Boomi	MuleSoft
iPaaS technically supports required constructs	Yes	Yes	Yes	Yes
Simplicity of use	Difficult to customize and debug	Easy	More technically capable than others, with many options, but more difficult to use.	Easy
Similarity to DevFlows	Somewhat. Relies on installing adapter packages into target systems that go beyond basic data and API access.	Very similar	Similar, but unnecessarily more capable for our use-case.	Similar, but unnecessarily more capable for our use-case.
Other Considerations	Has pre-built templates for many NS-SF flows, but still will require customization. We documented and learned all we could from the templates.	If we build the integration using a similar tool and understand the IP that goes into Layer 3, we can easily port to DevFlows later.	Pre-existing Central SF-NS integration that can be referenced, but not with SuiteBilling.	Comes at 4x the cost. Additional features like API management not needed for our purposes.

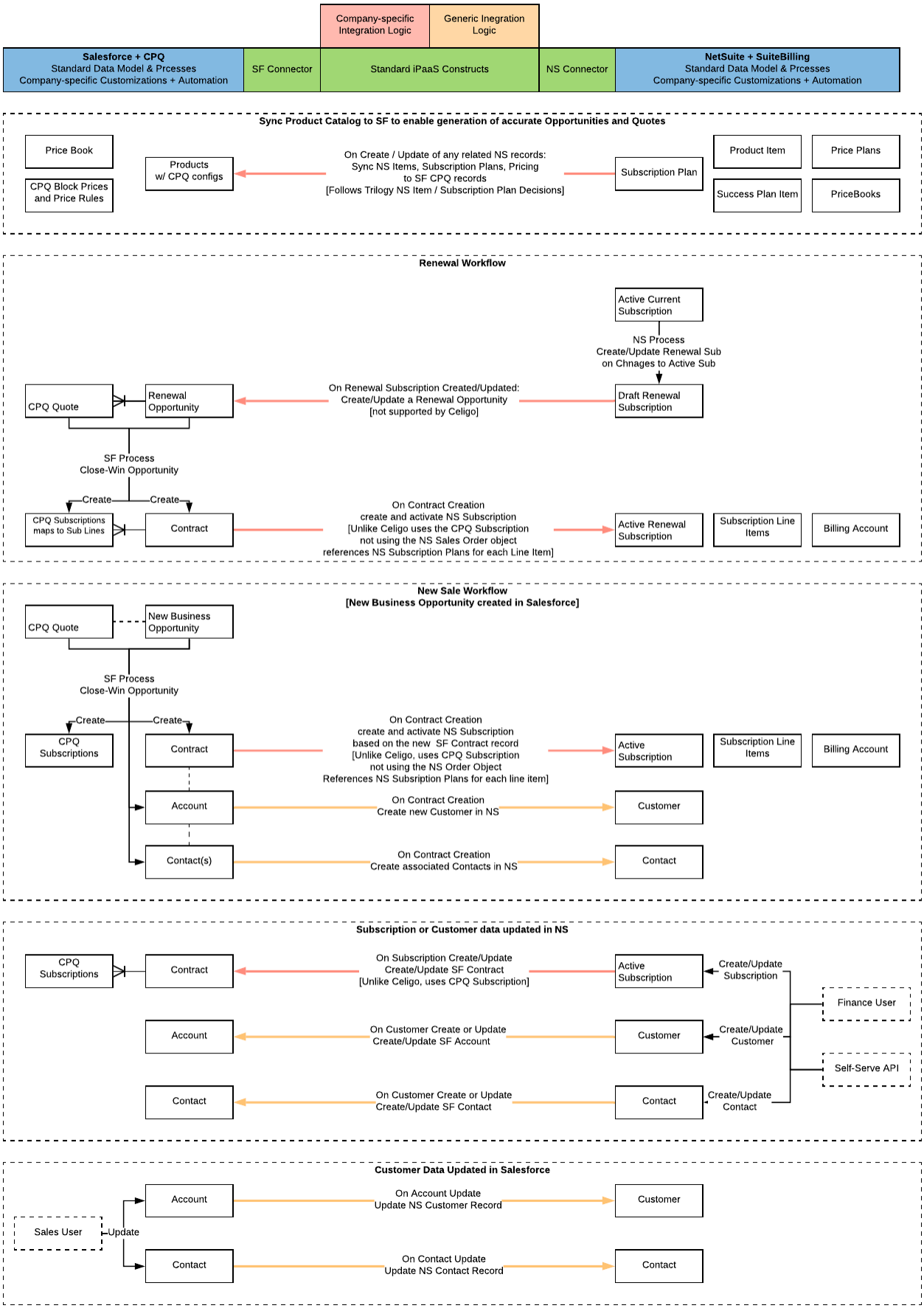
Integration

- We want to integrate both:
  - NS+SuiteBilling and SF data models - to the point where Sales have enough data to do their work independently in Salesforce and Finance can focus on their work in NetSuite.
  - The workflows between Sales and Finance to eliminate manual processes.
- **Data**

Data Objects Category		
Read/Write in SF, not sync'd to NS	Created and used exclusively by the sales process.	• Activities (Call Logs, Emails, Notes), Opportunities, Quotes, Leads, Account and Contact records that didn't complete a purchase.
Read/Write in NS, not sync'd to SF	Objects created/used exclusively by the finance process	• Financial Accounts, Charges, Invoices, etc. • Billing Accounts - created from a new SF Account, but then not used by SF.
Read/Write in NS, Read-Only in SF	Objects mastered in NS but sync'd to SF because it is used (but	• Product Catalog Items: ○ Items, Subscription Plans, Price Books, Price Plans

	not written) by sales-related modules or out-of-the-box workflows.	● Renewal Subscriptions ○ Used to create and update SF Renewal Opportunities.
Read/Write in SF, copied as Read-Only in NS	Objects mastered in SF but sync'd to NS because it is used (but not written) by finance-related modules or out-of-the-box workflows.	● Signed Quote/Order Form Document ○ Quote gets signed by Sales and becomes immutable. ○ The result gets propagated to NS to the Subscription record with respective line items and pricing.
Dual Master	Objects modified in both SF and NS	● SF Account/Contact - NS Customer/Contact ○ Accounts and Contacts are modified by Sales - creating new customers and contacts, modifying customer data. ○ The same records can also be created by Self-Serve directly in NetSuite or Customer and Contact information can be modified by Finance. ● SF Contract/CPQ Subscription - NS Subscription / Subscription Line Items ○ SF Contract is an equivalent object to the NS Subscription. It's needed in SF to track what the customer has purchased and to generate proper amendment quotes. ○ While NS is the system of record for Subscription data, the Subscriptions can be created and updated based on Renewals, New Business, and Upsell/Amendments performed by Sales in Salesforce, using Salesforce's workflows. ○ Subscription can also be created directly in NS via Self-Serve.

- **Workflows**
  - The integration can be broken down into several key workflows.
  - Each workflow is implemented by one or more iPaaS flows.
  - Some are simple,and can be easily templated (or work OOTB with Celigo)
    - Account-Customer, Contact-Contact
    - Others have company-specific business logic and data integrations requiring either customizing Celigo templates or implementing from scratch.



[click on the image for Lucidchart link. Alternatively: high-res image]