SalesforceとMuleSoftによるシステム連携入門

API連携の基本と実践



Start Challenge Consulting co., ltd.

2026年8月発行

Confidential｜本資料の無断転載・二次利用を禁じます

Start Challenge
Consulting co., ltd.
挑む力が、世界を照らす

はじめに｜システム連携が重要になっている背景

Salesforce導入後も顧客情報や契約情報が基幹システムとSaaSに分散し、CSVによる手作業と部門ごとの重複入力が残っている。本書はこの状態を前提に、APIを軸とした連携の考え方と進め方を整理する。

現状（As-Is）｜連携が課題になる背景

- Salesforceを導入しても、顧客情報や契約情報が基幹システムに分散している
- 複数のSaaS導入によって、データの保存場所が増えている
- CSVによる手作業のデータ連携が残っている
- 営業・サービス・経理部門で同じ情報を重複入力している
- システムごとに個別連携を開発し、保守負担が増えている

本書で分かること

01 SalesforceとMuleSoftの基本的な役割

02 APIを活用したシステム連携の考え方

03 代表的な連携方式と選び方

04 導入プロジェクトの進め方

05 設計・運用で失敗しないためのポイント

補足：本書は入門者向けに、Salesforce（顧客接点の管理）とMuleSoft（API統合基盤）の役割分担を前提として構成しています。

注記　事実：提供資料に記載の背景・目的／解釈：データ分散と手作業の残存が連携基盤検討の起点／提案：本書の5論点に沿って検討を進める。 出典：提供構成案（全15ページ）

企業が抱えるシステム連携の課題と影響の構造

手作業による転記と複数システムへの重複入力は、システムごとに個別連携を積み上げてきた構造から生じており、業務品質の低下だけでなく保守コスト増加とデータ活用の遅れという経営課題に連鎖している。

発生事象	直接原因	構造的原因	業務への影響	経営への影響
Salesforceと基幹システムで顧客情報が一致しない	CSVの出力・加工・取込作業が属人化している	システム間を1対1で直接接続している	入力ミスやデータ不整合の発生	IT部門の運用・保守コスト増加
受注内容を会計・販売管理システムへ再入力している	営業・サービス・経理で同じ情報を重複入力している	連携方式・データ形式の標準が定まっていない	顧客対応の遅延	データを活用した意思決定の遅れ
在庫や納期の確認に時間がかかる	連携処理が要件ごとに個別開発されている	システム改修のたびに複数の連携処理を変更する必要がある	営業担当者の生産性低下	新規サービス・チャネル追加のリードタイム長期化
夜間バッチが失敗しても翌朝まで気づけない		データの正本と更新権限が未定義	月次締め作業の長期化	

補足：個々の事象への対処（担当者の増員・作業手順の整備）では構造的原因が残るため、連携方式そのものの見直しが必要となる。

注記　事実：提供資料の「よくある課題」「経営・業務への影響」／解釈：事象・原因・影響を5階層に構造化／提案：構造的原因への対処を優先する。 出典：提供構成案（P3）

Salesforceの役割と連携対象

Salesforceは顧客接点(リード～契約・問い合わせ)を管理する領域を担い、受注後の請求・出荷・入金は別システムで処理されるため、Salesforce単体では業務が完結せず外部システムとの連携が前提となる。

Salesforceで管理する代表的な情報

リード	取引先
取引先責任者	商談
見積	契約
問い合わせ	活動履歴

営業、マーケティング、カスタマーサービスなどの顧客接点を管理するクラウド型CRM。

主な連携対象システム

販売管理	在庫管理
ERP・会計	ECサイト
MAツール	コールセンター
人事・勤怠	DWH・BI

接続方式・データ形式・更新頻度は対象ごとに異なる。

Salesforce単体で完結しにくい理由

01 基幹データをSalesforceへ集約する必要がある

02 受注後の請求・出荷・入金処理は別システムで行われる

03 外部システムごとにデータ形式や接続方式が異なる

補足:連携対象の棚卸しでは「どの情報を、どちらのシステムを正として持つか」を先に決めることが後続設計の前提となる(詳細はP12)。

注記 事実:提供資料の管理情報・連携対象一覧／解釈:Salesforceの担当範囲は顧客接点に限定される／提案:連携対象を業務プロセス単位で棚卸しする。 出典:提供構成案(P4)

MuleSoftの基本と役割

MuleSoftはAPIを通じてSalesforce・基幹システム・クラウドサービス・データベースを接続する統合プラットフォームであり、設計から実行・公開・監視までを同一基盤上で管理できる点が個別開発との差異である。

主な機能

- APIの設計・開発
- システム間のデータ連携
- データ形式の変換
- 連携処理の実行・制御
- APIの公開・管理
- 認証・アクセス制御
- ログ・稼働状況の監視

Anypoint Platformの主な構成要素

構成要素	主な役割
Design Center	APIの設計
Exchange	APIや部品の共有・再利用
Runtime Manager	実行環境の管理
API Manager	APIポリシーやアクセス制御
Monitoring	処理状況や障害の監視
Studio	連携処理の開発

補足：設計（Design Center）と再利用（Exchange）を組織のルールとして運用できるかが、API資産化の分岐点となる。

注記　事実：提供資料の機能・構成要素一覧／解釈：設計～監視を単一基盤で管理できる点が個別開発との差異／提案：構成要素ごとの管理責任者を定める。 出典：提供構成案（P5）

SalesforceとMuleSoftを組み合わせるメリット

システム間を個別に直接接続する形から、MuleSoftを介したAPI接続へ整理することで、リアルタイム性・再利用性・変更耐性・可視性が同時に改善され、保守性と拡張性の向上につながる。

1. 連携の標準化

システムごとの個別連携を、API中心の共通方式へ整理できる。連携仕様の重複を削減。

2. リアルタイム反映

商談の受注、契約変更、在庫更新などを即時に連携できる。

3. API・部品の再利用

一度作成した顧客APIや商品APIを、複数の業務・チャネルで利用できる。

4. 変更影響の抑制

接続先の変更をAPI層で吸収し、Salesforce側の改修範囲を限定できる。

5. 一元的な監視

処理件数、エラー、応答時間などを管理し、障害を早期に検知できる。

導入前後の比較

導入前（As-Is）

システム間を個別に直接接続（N対Nの配線）

▶

導入後（To-Be）

MuleSoftを介してAPIで接続（ハブ型）

効果：**保守性・拡張性・可視性**の向上

注記　事実：提供資料の5つのメリットと導入前後比較／解釈：効果は再利用と可視化の運用定着に依存する／提案：小規模連携から共通APIを積み上げる。 出典：提供構成案（P6）

代表的な連携ユースケース

連携の起点は「受注」「在庫照会」「入金確認」「会員統合」の4類型に集約され、いずれもSalesforce画面上で外部システムの情報を参照・更新できる状態を目標とする。

連携先	主な処理の流れ	業務上の狙い
基幹システム 受注連携	Salesforceで商談を受注 → 顧客・商品・受注情報を基幹システムへ連携 → 基幹システムで受注番号を採番 → Salesforceへ受注番号と処理結果を返却	受注後の再入力を排除し、受注番号を単一の管理キーにする
在庫管理システム 在庫・納期照会	商品の在庫数をSalesforce画面に表示 → 商談・見積作成時に在庫と納期を確認 → 在庫不足時に営業担当者へ通知	照会待ちを解消し、見積段階で納期回答を可能にする
会計システム 請求・入金連携	請求・入金情報をSalesforceへ反映 → 営業担当者が顧客の未入金状況を確認 → 入金完了後に案件ステータスを更新	与信・督促対応を営業側で把握し、月次締めを短縮する
ECサイト 会員・購買連携	EC会員をSalesforceの顧客情報へ統合 → 購買履歴を連携 → 問い合わせ履歴と購買履歴を一画面で確認	オンラインと対面の顧客像を一元化する

補足：4類型はいずれも「顧客」「商品」「受注」の共通APIを共有するため、着手順に関わらず先に共通APIを整備する方が総コストを抑えやすい。

注記　事実：提供資料の4ユースケース／解釈：共通APIの重複が大きく、先行整備が有効（仮説）／提案：受注連携を起点に共通APIを設計する。 出典：提供構成案（P7）

API-led Connectivityの基本｜3層でAPIを設計する

APIをSystem／Process／Experienceの3層に分けることで、システム固有の接続方式・業務ルール・チャネル要件を切り離し、再利用性を確保しながら変更影響を各層に閉じ込められる。

利用チャネル側 ← → 接続先システム側

Experience API

Web・モバイル・Salesforce画面など
チャネル別に最適化

- 必要な項目だけを返す
- 画面やチャネルごとの形式へ変換する
- Salesforce画面向けの情報を提供する

Process API

複数システムのデータを
業務処理として提供

- 顧客情報と契約情報を統合する
- 受注登録や在庫確認などの業務ルールを実行する
- データ変換や入力チェックを行う

System API

基幹システム・DB・
Salesforceのデータへアクセス

- システム固有の接続方式を隠す
- データ取得・登録・更新を共通化する
- Salesforce API、ERP API、DB接続などを集約する

3層化するメリット

APIを再利用しやすい

システム変更の影響を限定できる

責任範囲を明確にできる

APIの乱立を防ぎやすい

注記　事実：提供資料の3層定義と各層の役割／解釈：層ごとに変更理由が異なるため分離が有効／提案：層の境界を設計標準として文書化する。 出典：提供構成案（P8）

データ連携方式の選び方

方式は業務が求める即時性とデータ量で決まる。照会系はリアルタイム、大量同期はバッチ、後続処理はイベント駆動、API未提供の旧システムはファイル連携が適し、単一方式への統一は避けるべきである。

リアルタイム連携

ユーザー操作やデータ更新を契機に、その場で処理する。

適した業務

在庫・納期照会／顧客情報の参照／受注登録／契約状況の確認

注意点

応答時間／接続先の停止／タイムアウト／同時アクセス数

バッチ連携

一定時間ごとに複数データをまとめて処理する。

適した業務

売上実績の夜間連携／商品マスタの一括更新／日次・月次の集計／大量データの同期

注意点

処理時間／差分抽出／再実行範囲／SalesforceのAPI制限

イベント駆動型連携

データの変更や業務イベントを契機に非同期で処理する。

適した業務

商談受注後の後続処理／契約更新の通知／大量処理の負荷分散／複数システムへの同時連携

注意点

処理順序／重複到達／結果通知の設計／障害時の再送

ファイル連携

CSVなどのファイルを共有領域やSFTP経由で受け渡す。

適したケース

APIを提供していない旧システム／日次で十分な業務／一時的な移行・連携

注意点

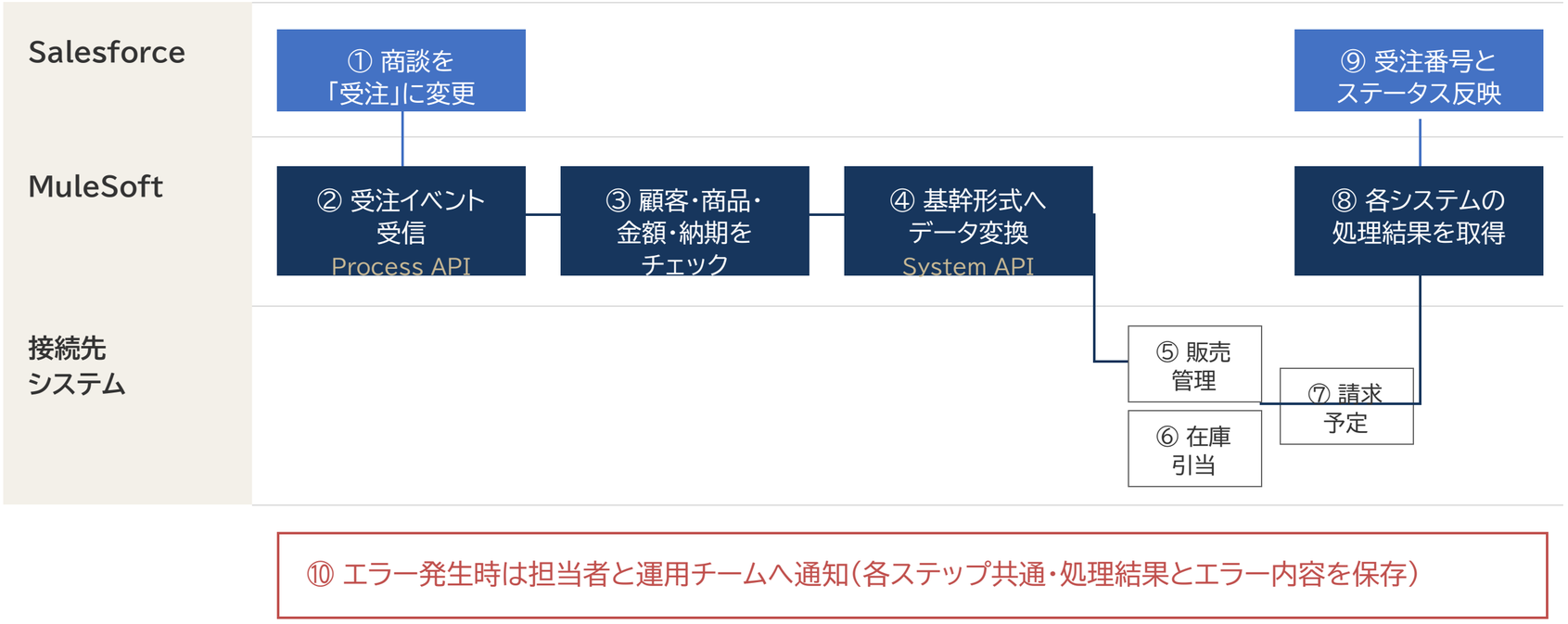
文字コード／連携遅延／授受の確認手段／属人化の再発

補足：方式選定は「即時性が業務要件として必要か」を先に確認する。必要でない処理をリアルタイム化すると、応答時間とAPI制限の制約を不必要に受ける（P14 失敗1）。

注記　事実：提供資料の4方式と適用業務・注意点／解釈：即時性とデータ量が選定軸／提案：業務単位で方式を割り当て、単一方式への統一は避ける。 出典：提供構成案（P9）

連携アーキテクチャの設計例｜受注情報の3システム連携

Salesforceの受注をMuleSoftが受け、販売管理・在庫管理・会計へ振り分けて結果をSalesforceへ戻す構成では、正本の定義と部分失敗時の戻し方を先に決めておくことが設計の要点となる。



補足：⑤〜⑦は非同期・並列実行を前提とし、⑨の反映は全システムの結果確定後に行う。想定シナリオは提供構成案の例示。

設計時の検討事項

どのシステムを正とするか

同期処理と非同期処理の使い分け

部分的に処理が失敗した場合の戻し方

同じ受注を二重登録しない仕組み

障害復旧後の再実行方法

注記　事実：提供資料の処理フロー10ステップと検討事項／解釈：レーン別に責務を整理／仮説：⑤〜⑦の並列化が処理時間短縮に有効（仮）／提案：部分失敗時の補償処理を先に定義する。 出典：提供構成案（P10）

導入プロジェクトの進め方｜6ステップ

現状調査から運用開始までを6ステップで進める。特にStep2の連携要件（件数・応答時間・エラー時の業務対応）とStep5の異常系テストが、本番稼働後の安定性を左右する。

Step 1 現状調査	Step 2 連携要件の整理	Step 3 API・アーキテクチャ設計	Step 4 開発・単体テスト	Step 5 結合・性能・障害テスト	Step 6 本番移行・運用開始
対象システムと責任者を整理する	連携するデータ項目	System／Process／Experience APIの切り分け	接続設定	システム間のデータ整合性	リリース手順
現在の連携処理を一覧化する	送信元・送信先	同期・非同期・バッチ方式の選定	データ変換	大量データ処理	切り戻し手順
CSVや手作業を含めて業務フローを確認する	連携タイミング／処理件数	APIの命名規則とバージョン管理	業務ロジック	タイムアウト	初期データ移行
データ量、処理件数、実行時間を確認する	許容応答時間	認証・認可方式の決定	例外処理	接続先停止	監視設定
	エラー時の業務対応		ログ出力	重複送信	運用引き継ぎ
	セキュリティ要件		テストデータ作成	再実行	障害連絡体制

プロジェクト開始前に確定すべき事項　①連携対象業務と責任者　②データの正本　③連携方式（同期／非同期／バッチ）　④エラー時の業務対応と運用体制

注記　事実：提供資料のStep1～6と各作業項目／解釈：Step2・Step5が稼働後の安定性に直結／提案：Step1完了時点で正本と責任者を確定させる。 出典：提供構成案（P11）

データ設計・エラー処理のポイント

外部IDと採番ルールの統一、および項目単位での正本の定義が連携品質の前提となる。エラーは分類して扱い、冪等性を確保した再実行設計を併せて用意する。

データ設計

- Salesforce IDだけに依存しない外部IDを設定する
- 顧客・商品・契約コードの採番ルールを統一する
- 必須項目と初期値を定義する
- 日付、通貨、文字コードを統一する
- コード値の変換表を管理する

データの正本を決める（例）

データ種別	正本となるシステム
顧客基本情報	Salesforce
商品・価格情報	基幹システム
在庫情報	在庫管理システム
請求・入金情報	会計システム

エラー処理 | 6分類で扱う

入力エラー	認証エラー	接続エラー
タイムアウト	業務エラー	システムエラー

分類ごとに通知先・自動リトライの可否・業務側の対応手順を定義する。

再実行設計

- 自動リトライの回数と間隔
- 手動再実行の方法
- 再実行対象データの特定
- 二重登録を防ぐ冪等性
- 処理履歴とエラー内容の保存

補足：正本の定義は項目単位で行う。同一オブジェクトでも項目ごとに更新元が異なる場合があるため、システム単位の定義では運用時に判断が分かれる。

注記　事実：提供資料のデータ設計・正本例・エラー6分類・再実行設計／解釈：冪等性の確保が再実行運用の前提／提案：正本は項目単位で定義する。 出典：提供構成案（P12）

セキュリティと運用監視

認証・権限・暗号化をAPI層で統一的に適用し、稼働状況とSalesforce API使用量を継続監視することで、障害の早期検知と整合性確認までを定型手順として運用できる。

セキュリティ設計

- OAuthなどを利用した認証
- APIごとのアクセス権限
- IPアドレス制限
- 通信の暗号化
- 個人情報や機密情報のマスキング
- 認証情報の安全な保管
- 操作・アクセスログの保存

運用監視項目

- APIの稼働状況
- 成功・失敗件数
- 応答時間
- タイムアウト数
- バッチ処理時間
- 再実行件数
- **Salesforce API使用量**

障害発生時の流れ

1. エラーを検知
2. 運用担当者へ通知
3. 原因と影響範囲を確認
4. 自動または手動で再実行
5. 接続先との整合性を確認
6. 障害記録と再発防止策を登録

補足：Salesforce API使用量は組織単位の上限があるため、監視項目として常設し、バッチ設計時に消費量を見積もる必要がある。

注記　事実：提供資料のセキュリティ設計・監視項目・障害対応6ステップ／解釈：API使用量は上限管理が必要な監視項目／提案：障害対応手順を運用引き継ぎ資料に定型化する。 出典：提供構成案（P13）

よくある失敗と対策

失敗の多くは技術的な難易度ではなく、方式選定・正本定義・異常系設計・運用引き継ぎといった「決めるべきことを決めていない」ことに起因する。

#	よくある失敗	対策
1	すべてをリアルタイム連携にする	業務上必要な即時性を確認し、バッチや非同期処理を使い分ける
2	個別要件ごとにAPIを作成する	顧客・商品・受注など、再利用可能な単位でAPIを設計する
3	データの正本が決まっていない	項目単位で更新元と更新権限を定義する
4	正常系だけで設計する	タイムアウト、接続停止、重複、部分失敗を想定する
5	処理量やAPI制限を確認していない	ピーク時の件数、データサイズ、同時実行数を見積もる
6	運用部門への引き継ぎが不足する	監視、通知、再実行、問い合わせ対応を含む運用手順を作成する
7	MuleSoftの開発が属人化する	命名規則、共通部品、レビュー基準、APIカタログを整備する

共通する対策　設計標準（方式選定基準・命名規則・正本定義・異常系要件）を文書化し、レビュー時の判定基準として運用する

注記　事実：提供資料の失敗7項目と対策／解釈：いずれも設計標準の不在に起因する／提案：設計標準をレビュー基準として制度化する。 出典：提供構成案（P14）

導入前チェックリストとまとめ

APIを再利用可能な企業資産として設計し、小規模な連携から共通APIと運用ルールを段階的に整備することで、拡張性と保守性に優れた連携基盤を構築できる。

業務・システム

- ☐ 連携対象となる業務を整理している
- ☐ 対象システムと責任者が明確である
- ☐ 現在の手作業やCSV連携を把握している
- ☐ 連携するデータ項目を定義している
- ☐ データの正本を決めている

方式・性能

- ☐ リアルタイム・バッチ・非同期の方式を決めている
- ☐ 1日・ピーク時の処理件数を把握している
- ☐ Salesforceおよび接続先のAPI制限を確認している
- ☐ 応答時間や処理完了時間の目標を決めている

障害・運用

- ☐ エラー分類と通知先を決めている
- ☐ 自動リトライと手動再実行の方法を決めている
- ☐ 二重登録を防止する設計がある
- ☐ 監視項目と運用担当者を決めている
- ☐ 障害時の連絡・復旧手順を整備している

まとめ

SalesforceとMuleSoftの連携では、単にシステム同士を接続するだけでなく、APIを再利用可能な企業資産として設計することが重要である。小規模な連携から始め、共通APIや運用ルールを段階的に整備することで、拡張性と保守性に優れた連携基盤を構築できる。

注記　事実：提供資料のチェックリスト14項目とまとめ／解釈：3領域（業務／方式／運用）に分類／提案：チェック未充足項目をプロジェクト開始条件として扱う。 出典：提供構成案（P15）

Contact

お問い合わせ

本資料の内容、および連携対象の棚卸し・API設計・導入プロジェクトの進め方に関するご相談は、下記までお問い合わせください。

お問い合わせ先

会社名	Start Challenge Consulting co., ltd.
メールアドレス	info@startchg.com
担当	平出

現状の連携処理の一覧化やデータ項目の整理段階からのご相談も承ります。

Start Challenge
Consulting co., ltd.
挑む力が、世界を照らす

次のステップ　①連携対象業務と現状の手作業の棚卸し　②データの正本と責任者の確定　③連携方式と処理件数の整理　④API設計方針の策定

注記　事実：連絡先情報はご提供内容に基づく／提案：次のステップはP11 Step1～Step3およびP15チェックリストに対応。