Contract-to-Cash Lifecycle Architecture (Salesforce <-> Smartsheet)

Generic system documentation | February 08, 2026

This document describes a reusable, client-safe reference architecture for a Contract-to-Cash (C2C) / document lifecycle tracker that integrates a CRM (Salesforce) with Smartsheet as the system of record, enforces stage-gated governance, and produces operational and executive visibility.

Primary purpose	Track revenue-critical milestones from Opportunity stage through SOW/PO, invoicing, and payment.
System of record	Smartsheet master tracker (one row per engagement).
Upstream trigger	Salesforce Opportunity stage change (event-driven intake).
Core controls	Stage gating, helper-date scheduling, role-based routing, evidence capture.
Visibility outputs	Operational action queues + executive dashboards (pipeline, aging, exceptions).

1. Architecture overview

- Event-driven intake from Salesforce into Smartsheet through a defined integration handshake.
- Smartsheet master tracker operates as a lifecycle state machine: each stage has prerequisites, completion fields, and evidence capture.
- Helper-date columns compute the next reminder date using business-day logic to drive a consistent follow-up cadence.
- Automations route update requests/reminders to the correct role (Sales, Delivery, Finance) and optionally escalate when SLAs are missed.
- Reports and dashboards provide both 'what do I do today?' operational visibility and 'where are the bottlenecks?' executive visibility.
- Standardization and scaling are supported via template governance and optional Control Center provisioning.

2. Logical layers

Source + Trigger	Detect lifecycle events and provide upstream metadata.	Salesforce Opportunity stage change; field mapping.
Integration / Handshake	Create/update a canonical record in Smartsheet with validated mapping.	API/middleware workflow; controlled form intake; idempotent upsert.
System of Record	Maintain the golden row per engagement and lifecycle history.	Smartsheet master tracker; reference/lookup sheets.

Workflow Engine	Enforce stage gates, stop/skip logic, and lifecycle transitions.	Stage prerequisites; completion checkbox/date; validation rules.
Scheduling Layer	Drive follow-up cadence using business days.	Helper-date columns; WORKDAY/NETWORKDAYS logic; holiday calendar.
Automation Layer	Notify and collect updates from owners based on role.	Alerts; update requests; conditional routing; optional escalation.
Access Control	Ensure stakeholders see only what they should and can edit only what they own.	Dynamic View / WorkApps patterns; role-based permissions.
Visibility Layer	Provide operational queues and executive reporting.	Reports; dashboards; pipeline/aging views; exception widgets.
Extensibility	Enable analytics and cross-platform reporting.	Data Shuttle; ODBC/API extracts; BI semantic model; Power BI.
Provisioning	Scale the solution consistently across portfolios.	Control Center provisioning; standardized templates and naming.

3. Data model (Smartsheet system of record)

The master tracker is designed as a one-row-per-engagement table. Each lifecycle stage is represented by a consistent column set to keep reporting and automation predictable.

- Canonical keys: Opportunity ID (primary key), engagement/client identifiers, owner roles (Sales, Delivery, Finance).
- Stage columns (repeated pattern): Stage Status (checkbox or status), Stage Date (completion date), Evidence (link/attachment), Notes.
- Reference tables: owner lookup and role routing tables; optional holiday calendar for business-day calculations.
- Audit readiness: store evidence links/attachments and date-stamped milestone completion for traceability.

4. Stage gating and lifecycle controls

- Prerequisites: a stage becomes active only when prior-stage conditions are met (prior stage completed + prior stage date present).
- Stop condition: reminders/update requests cease once the stage completion date is populated (or status indicates completion).
- Skip logic: if a stage is completed at intake (date present), the workflow bypasses reminders and advances to the next gate.
- Validation: enforce required fields at key gates (for example, do not advance to Finance steps without Delivery approval fields).

5. Scheduling and reminder cadence

Helper-date columns compute the next follow-up date using business-day rules. This enables consistent reminders without manual calendar work.

- Baseline date: created date or prior-stage completion date depending on the stage.
- Helper date: Next Stage Reminder (Auto) calculated with business-day logic (WORKDAY/NETWORKDAYS style).
- Cadence: weekly (or business-weekly) reminders until the stage is completed.
- Holiday support: optional holiday table integrated into business-day calculations.

6. Automation routing and escalation

- Role-based routing: update requests/reminders go to Sales, Delivery, or Finance based on the active stage and role mapping.
- Message content: request the owner to update the stage status/date, add notes, and attach/link evidence where applicable.
- Escalation (optional): after defined SLA thresholds or repeated reminders, notify leadership/ops roles.
- Orchestration (optional): use Bridge/middleware for advanced branching, multi-step logic, or integration calls beyond native automation.

7. Access control and governance

- Role-based visibility: limit stakeholders to their own engagement rows and relevant fields.
- Controlled editing: owners update only the fields for the stages they own; sensitive fields are restricted.
- Dynamic View / WorkApps patterns: provide a safer UI for contributors while protecting the underlying tracker structure.
- Operating model: weekly review using exception reports; SOPs define who updates what and when.

8. Reporting and dashboards

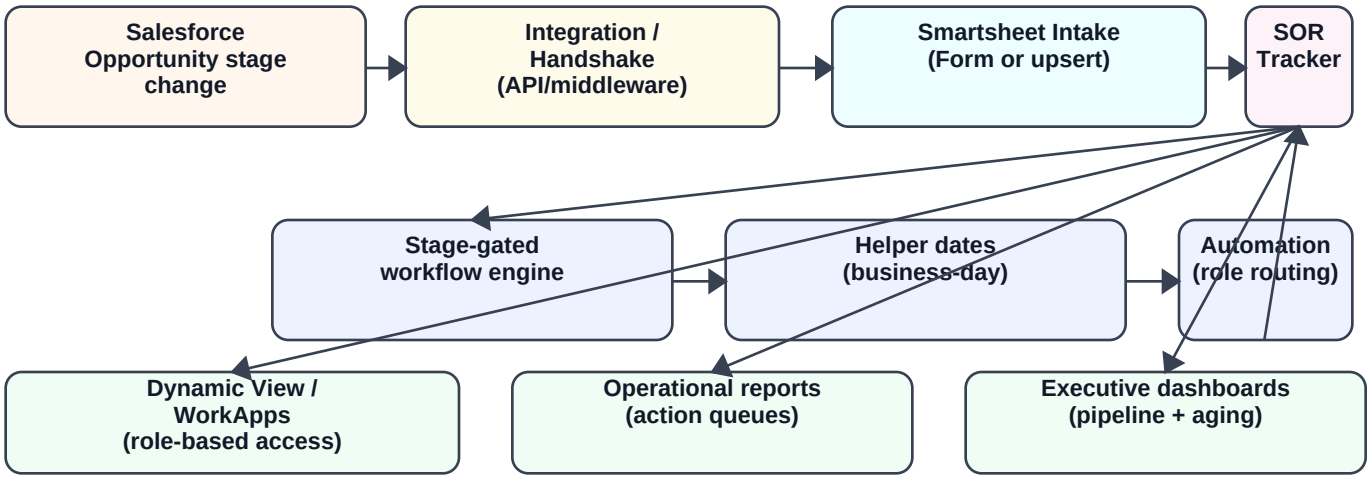
- Operational action queues: overdue items, stuck stages, next reminder due, and owner task lists.
- Executive dashboards: pipeline by stage, aging distribution, bottleneck heatspots, and next actions by team.
- Performance metrics: cycle time per stage, SLA compliance, and throughput trends (optional).

9. Extensibility and BI

- Exports/integration: publish structured data via Data Shuttle schedules or ODBC/API extracts.
- BI layer: build a semantic model for cross-platform analytics and executive reporting (for example, Power BI).
- Correlation: join lifecycle data with finance/project systems to analyze revenue leakage, cycle time, and operational bottlenecks.

10. Workflow diagram

High-level flow (box-and-arrow):



Appendix A: Implementation checklist

- Define lifecycle stages and required fields per stage (status, date, evidence).
- Define role ownership per stage and build routing rules (Sales, Delivery, Finance).
- Define helper-date logic and reminder cadence; include holiday calendar if required.
- Implement intake mapping (Salesforce -> Smartsheet) with idempotent create/update behavior.
- Build exception reports and dashboards; test stop/skip logic end-to-end.
- Document SOPs and handoff rules; enable contributors via Dynamic View/WorkApps if needed.