Two packages. One on Salesforce, one on NetSuite. Here is exactly what lands in each org.

Continuous Control ships as a **Salesforce unlocked package** and a **NetSuite SDF application**. Nothing runs between them: Salesforce calls NetSuite directly over a signed, outbound-only connection. This page is the component inventory both administrators will see after install, with the prerequisites and the manual steps stated up front.

Salesforce package

UNLOCKED, ORG-DEPENDENT
KEYLESS · PRODUCTION-INSTALLABLE

APP AND TABS

- **Continuous Control** Lightning app with **5 tabs**: NetSuite Connection, NS Field Mappings, NS Sync Runs, Sync Run records, Performance Obligations.

LIGHTNING WEB COMPONENTS

- **NetSuite Sync card** `ccNetSuiteSyncCall` — one card for Order, Invoice, Invoice Batch Run, Credit Memo and Payment pages. Resolves its own mode from the record page, shows live progress, and stores the exact JSON payload sent.
- **Revenue Summary card** `ccContractRevenue` — Booked, Billed, Recognized and Deferred per obligation on the Contract, plus Refresh from NetSuite.
- **Connection Setup** `ccNsConnectionSetup` — the guided 9-step setup and health screen.
- **Field Mapping editor** `ccNsMappingEditor` — the live field map per record type, with NetSuite field discovery and type checking.
- **Sync Runs dashboard** `ccNsSyncRuns` — success rate, volume by sync type, and every run with its payload and result.

FLOW ACTIONS (11)

- Book Order · Sync Invoice · Sync Invoice Batch Run · Sync Credit Memo · Sync Payment · Sync Payment Batch Run · Sync Refund · Sync Account · Sync Product · Refresh POBs from NetSuite · Pull Cash Application from NetSuite. Each carries its own branded icon, so any of it can be automated without writing Apex.

DATA MODEL

- **4 custom objects**: Sync Run, Field Mapping, Field Mapping Detail, Performance Obligation.
- **3 hierarchy custom settings** (credentials, sync settings, pull state) and **2 custom metadata types** with 13 seeded record-link rows.
- **Fields on 14 standard objects** — the NetSuite Record Id, Sync Status, Sync Message and Sync Timestamp set on Account, Order, Invoice, Payment, Credit Memo, Refund, Contract, Product and the rest of the billing graph, plus Delivered and Fulfilled flags on Order Product for fulfillment-driven recognition.

AUTOMATION AND ACCESS

- **44 Apex classes** plus 18 test classes: payload builders, the queueable sync engine with chunking, writeback, and the retention purge.
- **1 permission set**, **1 custom notification type** for sync failures, an Unapply Payment quick action and flow.

NetSuite application

SDF PROJECT · ONE-COMMAND INSTALLER
OAUTH 2.0 MACHINE-TO-MACHINE

NAVIGATION AND THE CLOSE SCREEN

- A **Continuous Control center tab** in the NetSuite menu bar with three categories: **Close the Books** (Revenue Close Cockpit, Run Portfolio Close, Recognize Revenue, Recognition Backlog, Manage Accounting Periods, Close Run Log), **Reports**, and **NetSuite ARM**.
- **Revenue Close Cockpit** `ct_cockpit_sl` — a full-page controller screen: portfolio Booked, Billed, Recognized and Deferred, per-contract waterfalls, this period's close tasks, engine buttons and recent runs. A matching **dashboard portlet** puts the same figures on the NetSuite home page.

THE REVENUE ENGINE

- **1 RESTlet** — the single inbound endpoint. Everything Salesforce sends arrives here.
- **7 Suitelets** — Update POBs, Update Revenue Plans, Recognize and Reverse, the Cockpit, Native Sales Order, Run Close Now, saved-search builder.
- **2 user event scripts, 3 client scripts, 1 portlet, 1 scheduled map/reduce** for the hands-free close.
- **Engine Run Log record** — every run stamped with trigger, stages, contracts processed, obligations updated, schedule rows written, journal entries posted and the period recognized through.

RECORD MODEL (13 CUSTOM RECORDS)

- ARM Contract, ARM Order, ARM Order Line, **Performance Obligation**, Obligation Detail, Billing Schedule Group, Billing Schedule, Billing Period, Revenue Plan, Revenue Schedule, Revenue Event, Engine Run Log, Settings.

WHAT IT ADDS TO YOUR NATIVE RECORDS

- A **revenue rule field on the Item**, which the obligation and its revenue plan inherit.
- **Contract, period, amount and obligation fields** on transactions and transaction lines, so every posted entry traces back to the obligation that caused it.
- **Salesforce deep-link fields** on invoice, credit memo, payment, order and invoice line, so anyone in NetSuite can click straight back to the Salesforce record.
- A **GL Impact subtab** on every posting transaction, and billing schedule groups sublisted on the Customer.

REPORTING AND CONFIGURATION

- **25 saved searches**: portfolio and recognition waterfalls, deferred revenue roll-forward, close status by contract, recognition backlog, billed-not-recognized, obligations missing a rule, pending billings, open invoices, journal entry register. Native searches, so you can clone them, extend them and put them on any dashboard.
- **Settings record and script parameters** for subsidiary, currency, deferred and revenue accounts, tax item and schedule, and default revenue rule. Nothing account-specific is hardcoded, and there is no code to edit.

PREREQUISITES

Salesforce: Revenue Cloud Advanced and Billing. One self-contained package, nothing to install first.

NetSuite: administrator access with Custom Records, Server SuiteScript, Token-Based Authentication, REST Web Services and Revenue Recognition enabled.

SECURITY AND DATA FLOW

One integration record, one role, one token, **one endpoint**. Requests are signed and **outbound-only from Salesforce**; NetSuite never calls in.

No middleware and no third party holds your data. Credentials live in a protected org setting and are never displayed again after saving.

WHAT YOU CONFIGURE YOURSELF

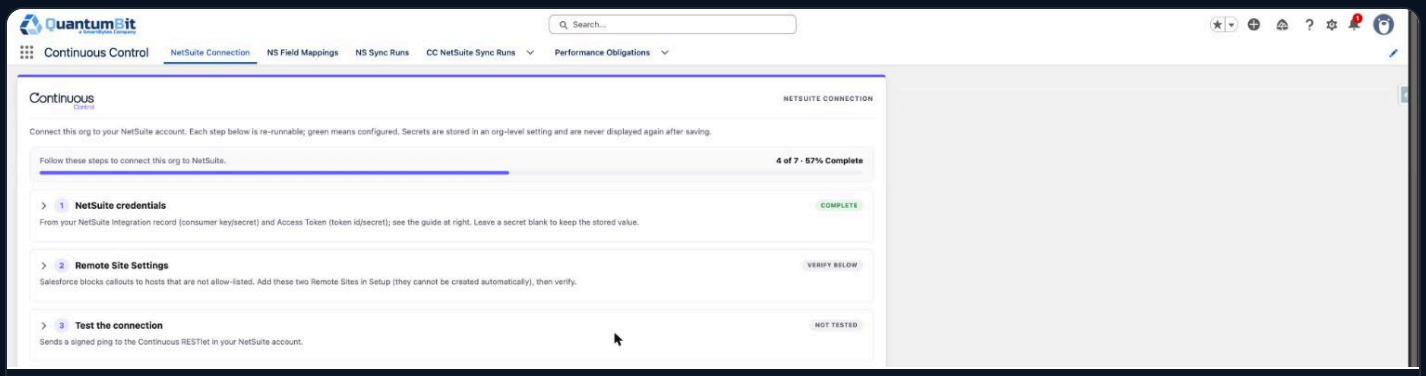
NetSuite, about 20 minutes: attach four saved-search sublists, mark the obligation form preferred, set one accounting preference, item and tax prep, lay out the dashboard.

Salesforce, about 20 minutes: assign the permission set, run the Connection wizard, place the cards on your page layouts.

IN YOUR SALESFORCE ORG

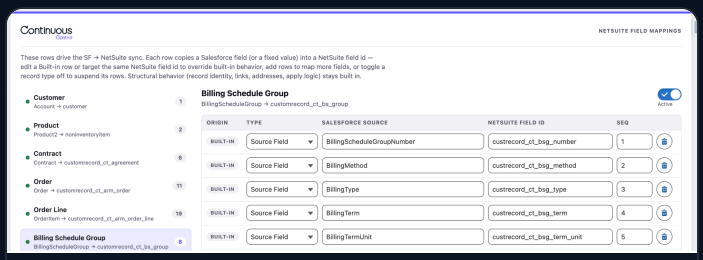
Setup, mapping, sync and observability, all inside Lightning.

Every screen below is part of the package. Nothing here is a custom build, and none of it needs a NetSuite login.



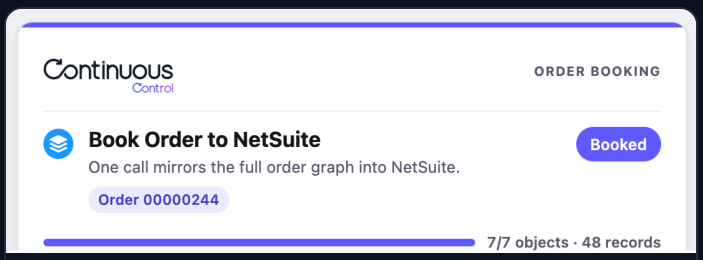
NetSuite Connection — the guided setup

Nine re-runnable steps with live status: credentials, remote sites, a signed connection test, component verification, environment discovery, mapping seed, reference data, sync settings, and a health screen that audits the live connection. `ccNsConnectionSetup`



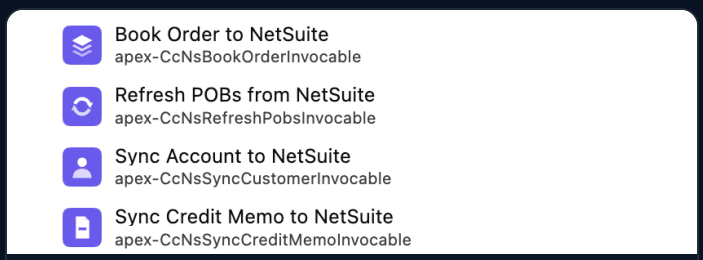
NS Field Mappings

The live field map per record type. Built-in rows show packaged behavior; add your own with NetSuite field discovery and type checking on save. `ccNsMappingEditor`



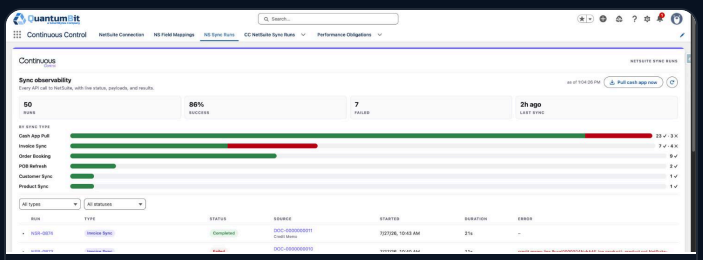
NetSuite Sync card

One card, five record pages. Builds the order graph as a single payload, sends one authenticated call, and links to the JSON that went over the wire. `ccNetSuiteSyncCall`



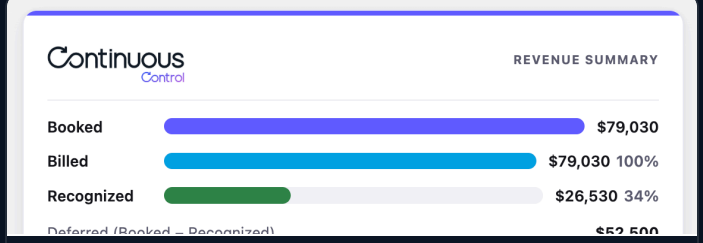
Flow actions for every sync

Booking, billing, credit memos, payments, refunds and obligation refresh are all invocable, so they automate from record-triggered or scheduled Flows without Apex. Four of the eleven shown, as they appear in the Flow action picker.



Sync observability

Every API call to NetSuite: success rate, volume by type, duration, source record and the exact error text when one fails. Failures also raise a notification. `ccNsSyncRuns`



Revenue Summary on the Contract

Booked, Billed, Recognized and Deferred per obligation, mirrored back from NetSuite so revenue is visible to people who never log in there. `ccContractRevenue`

Not pictured

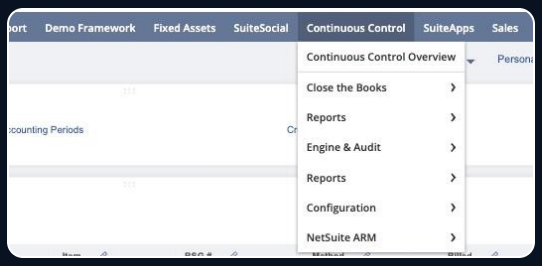
Sync Run records store the exact payload sent and the result returned, so any sync can be audited or replayed. A retention setting purges them on your schedule.

Performance Obligation records get their own tab and a related list on the Contract.

One permission set grants the whole app, and sync failures raise a custom notification.

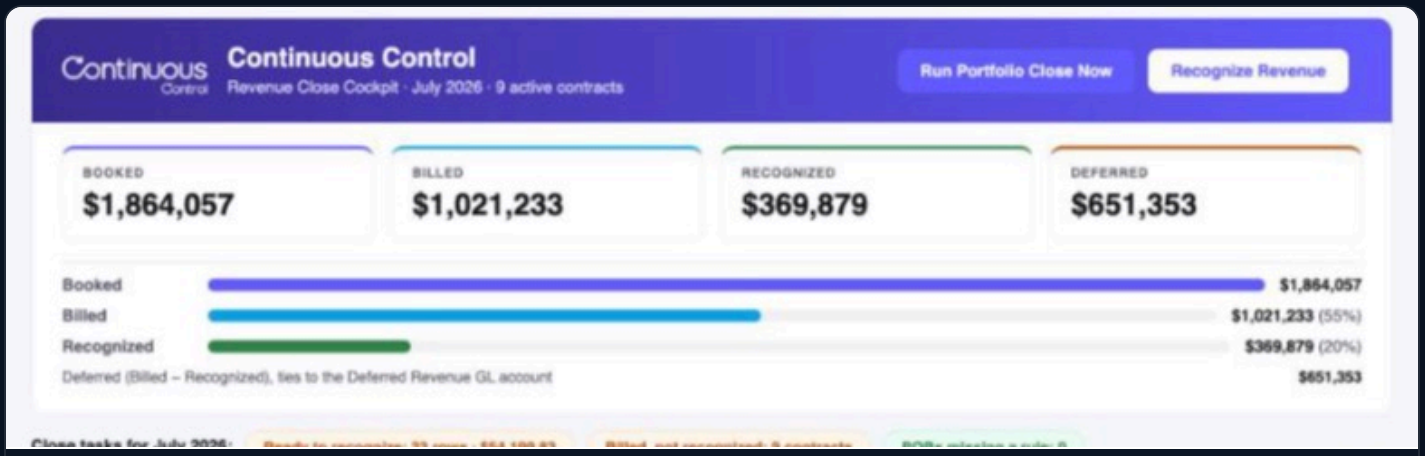
IN YOUR NETSUITE ACCOUNT · THE RECORD MODEL

A real NetSuite application, not a pile of custom records.



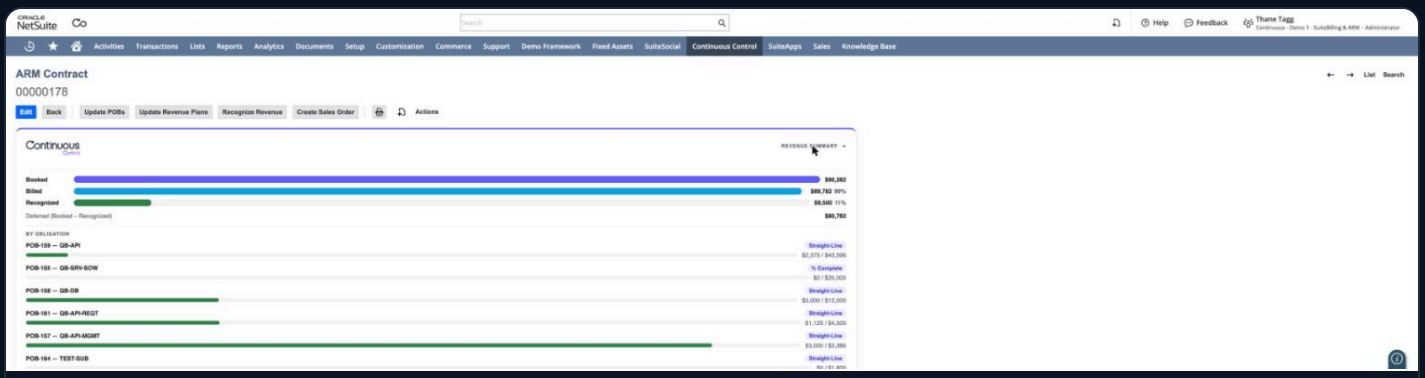
Its own place in the menu bar

The install adds a **Continuous Control center tab** with Close the Books, Reports and NetSuite ARM categories, so the close routine, the reports and the record lists are where a controller expects them. Everything below is reached from this menu or from the record itself.



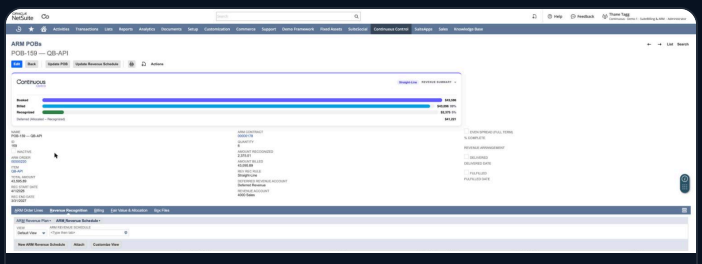
Revenue Close Cockpit

The controller's screen: portfolio Booked, Billed, Recognized and Deferred, this period's close tasks, a waterfall per contract, one-click Run Portfolio Close and Recognize Revenue, and the recent engine runs. Deferred equals Billed minus Recognized and ties to the general ledger. `ct_cockpit_sl`



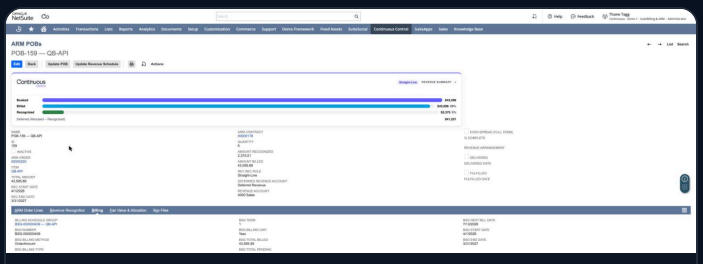
ARM Contract — the mirrored order graph

The Salesforce contract, its orders and every order line, mirrored with quantity, list price, discount, net price, term dates and billing frequency, each line already attached to its performance obligation. The revenue summary and the engine buttons sit on the record itself, and a link opens the source contract in Salesforce.



Obligation → plan → schedule

Each obligation carries a revenue plan and a month-by-month schedule: planned, recognized, status, and a link to the journal entry that posted it.



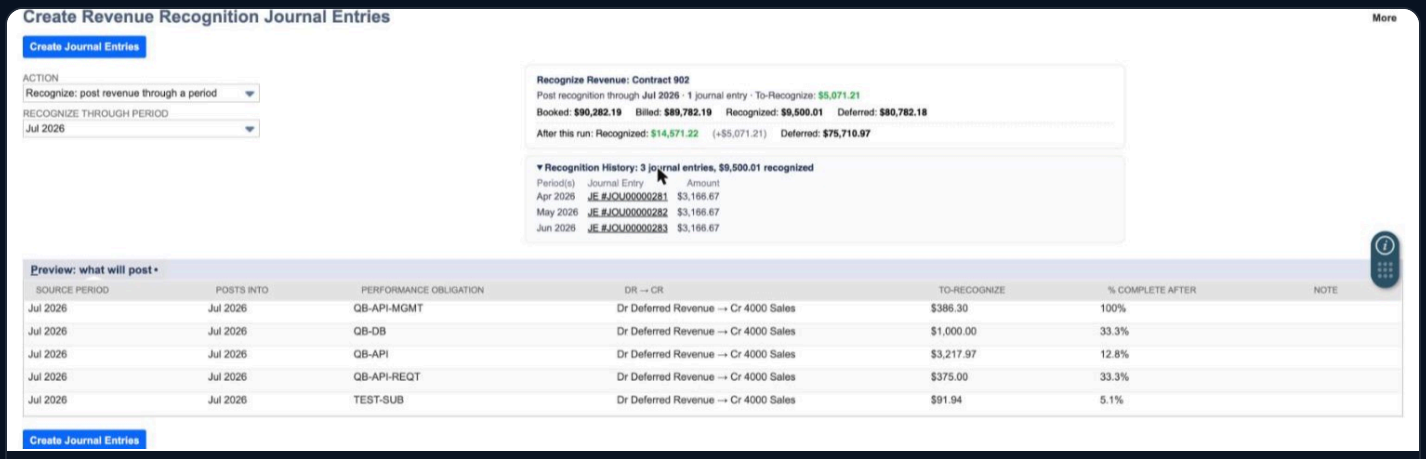
Billing, mirrored beside it

The billing schedule group header with its schedules and periods, original and amendment, so billed and pending sit next to the revenue on the same obligation.

IN YOUR NETSUITE ACCOUNT · THE LEDGER

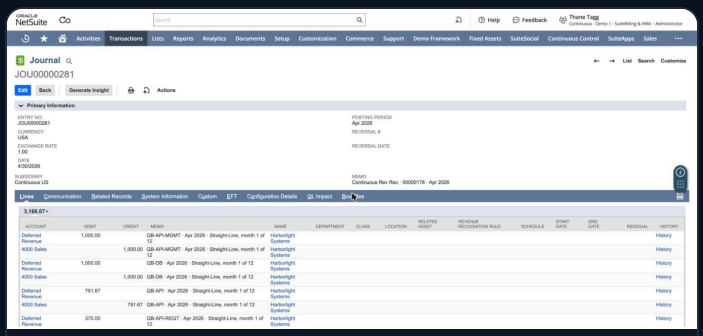
Revenue is previewed before it posts, and it posts as ordinary NetSuite entries.

Nothing here is a shadow ledger. The engine writes native journal entries and the invoices it receives are native invoices, so the trial balance, the financials and every standard NetSuite report already include them.



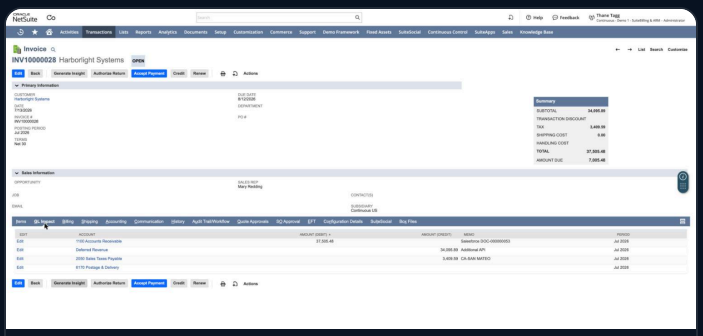
Recognition is preview, then post

Balances before and after the run, the full recognition history with a link to every journal entry, and a line-by-line preview naming the obligation, the accounts, the amount and the period each row will land in. Opening the screen never posts anything. The same form reverses a period, deleting entries in open periods and reversing them into the current one when the period is closed.



Journal entries a person can read

Debit Deferred Revenue, credit Revenue, in the right period, customer on every line, and memos naming the item, the month and the method.



Native transactions, native GL

Synced invoices are ordinary invoices. GL Impact shows the register on the record: receivable debited, deferred revenue and tax credited.

Also in the box

25 saved searches, including the deferred revenue roll-forward, the recognition waterfall by period, and the exception lists a close depends on. Native searches, so you can clone and extend them.

A scheduled close that runs the engine unattended, with a run log recording what each run touched.

A settings record and script parameters for accounts, subsidiary, currency, tax and the default revenue rule.

One engine at a time

Continuous Control recognizes revenue from its own obligation model, so you do not need NetSuite Advanced Revenue Management switched on to use it.

If you already run native ARM, or you graduate to it later, **each obligation maps one-to-one to a revenue element**, and the application can generate the native sales order that ARM works from. Pick one recognizer per contract.

WHAT INSTALLATION LOOKS LIKE

Your NetSuite administrator runs a **one-command installer** and creates a role and token. Your Salesforce administrator installs **one package** and works down the Connection screen on page 2, which tests the connection and verifies the NetSuite side before anything syncs. A setup guide covers both halves, from prerequisites through the first recognized revenue, and **every step is safe to re-run**.