SF CA 2026

Dreamforce

Financial Services Cloud

Workshop Guide

Your path to Success at Dreamforce 2026

UCM Order CheckBook + BSA Employee Agent — In-Workshop Build Guide

End state: You'll stand up the shipped **Order CheckBook** Unified Catalog Management (UCM) service process on the standard object model, test it from the Action Launcher on a client record, then turn on the **Banking Service Assistant** Employee Agent and run the **Checkbook Ordering** topic conversationally. No managed package, no code.

This guide is for the **pre-configured workshop org** — the prerequisites below are already done for you. Each step states the **expected result** so you can verify as you go. Use the **exact API names** shown; they are load-bearing.

Already done for you (the pre-work — your presenter will walk this)

Your org ships with all of this in place, so we can skip straight to the build:

- **OmniStudio Metadata API Support** = ON; **Discovery Framework** + **Context Definitions** = ON.
- **State and Country/Territory Picklists** enabled.
- The **5 Named Credentials** (the shared `FSC_fsc_integrations_V1_0_0` is the one Order CheckBook uses) — they point at Salesforce demo endpoints that return mock data, so **no core banking system is needed**.
- **Permission sets** assigned to your user; **Org-Wide Defaults (OWD)** set for the service-process objects.
- **Einstein → Agentforce → Banking Service Assistance (Employee Assistance)** all enabled, and the org-level `AgentforceEmployeeAgent` **entitlement** + `Employee Agents for Financial Services` permission set are provisioned (this is what lets an Employee Agent Commit/Activate).
- The **Action Launcher** component is already on the Person Account Lightning page.

> If you're doing this on your **own** org later, none of the above is pre-done — use the take-home kit (`ucm-bsa-vibes-kit`), which audits and sets all of it up first.

Part 1 — Create the Integration Definition

The Integration Definition tells the intake how to call the (mock) checkbook service.

1. Go to **Setup** → **Integration Definitions** → **New** (Integration Definition).
2. Set:
   - **Name:** `FSC_OrderCheckbook`
   - **Type:** `Apex`
   - **Apex Class:** `OrderCheckbookIntegrationProvider` (namespace `fscserviceprocess`)
3. When you select the Apex class, the **Named Credential** attribute auto-populates to **`FSC_fsc_integrations_V1_0_0`**. Leave it.
4. **Save.**

Expected result: an active Integration Definition named `FSC_OrderCheckbook`.

Part 2 — Clone + activate v2 of the intake OmniScript

1. Open the **OmniStudio Designer** (App Launcher → **OmniStudio** → OmniScripts).
2. Find the shipped OmniScript **`FSCOrderCheckbook`**, open it, and **Clone / Create New Version** → this creates **version 2**.
3. **Activate version 2 from inside the Designer** (the Activate button on the version).

> ⚠️ **Activate in the Designer only.** Activation compiles the OmniScript's Lightning component. Do **not** flip the `IsActive` flag through the API/data layer — that produces an "active" record that won't actually run.

Expected result: `FSCOrderCheckbook` **version 2** shows **Active**.

Part 3 — Create the Flow Orchestration

1. Go to **Setup** → **Flows**.

2. Create the fulfillment **Flow Orchestration** for Order CheckBook with the **exact API name** — this is load-bearing and must match everywhere:
    - **API Name:** `ProcessOrderChkbkRqst`
3. **Do NOT activate it yet** — you'll set its start condition in Part 5 (it needs the Product Id first).

Expected result: a Flow Orchestration named `ProcessOrderChkbkRqst` exists, **not yet active**.

Part 4 — Install the Service Process Template (UCM)

1. Open **Unified Catalog** (App Launcher → **Unified Catalog** → **Templates** tab).
2. Find the **Order CheckBook** template → **Install**.
3. When it redirects you to the created **Product**, **copy the `01t…` Product Id from the URL** — you'll paste it in two places in Part 5.
4. Confirm the **Product is Active** (activate the service-process/Product record if it isn't).

> On Summer '25 release the `Product2` record *is* the service process — there's no separate catalog-item record to look for. An **inactive** Product throws "provide a productId / active svcCatalogItemDefApiName" at test time, so make sure it's Active.

Expected result: an **Active** Product named **Order CheckBook**, and you have its **`01t…` Product Id** copied.

Part 5 — Bind the Product Id (the two connections)

Both bindings must point at the **same** Product Id from Part 4.

5a — In the map Integration Procedure

1. In the **OmniStudio Designer**, open Integration Procedure **`ServiceProcess_MapServiceNameToProductId`** → **Clone / New Version**.
2. In the **`ServiceNameToProductIdMap`** step, find the element named **`OrderCheckbook`** and set its value to your **Product Id**.

3. **Activate** the Integration Procedure (in the Designer).

Expected result: the `OrderCheckbook` key maps to your Product Id, IP version Active.

5b — In the Flow Orchestration Start

1. Back in **Setup** → **Flows**, open **`ProcessOrderChkbkRqst`** → the **Start** element.
2. Set the entry condition: **All Conditions Are Met** → **`Product Id`** **Equals** **`<your Product Id>`**.
3. **Activate** the orchestration.

> ⚠️ Prefer the **condition** form (`Product Id Equals <id>`) over the shipped **formula** form — the formula is fragile and can fail to activate.

Expected result: `ProcessOrderChkbkRqst` is **Active** with the Product-Id start condition.

Part 6 — Add the intake to the Action Launcher

1. Go to **Setup** → **Action Launcher** → **New**.
2. Set:
   - **Guidance to Show:** `OmniScripts`
   - **Selected Object:** `Account`
3. Add the **`FSCOrderCheckbook`** intake OmniScript to the deployment. **Save.**

> The Action Launcher **component** is already on the Person Account page in your workshop org — this step wires *your* OmniScript into it.

Expected result: the Order CheckBook intake appears in the Action Launcher on a Person Account record.

Part 7 — Test the UCM process

1. Open a **Person Account** with financial accounts — use **Rachel Morris**.
2. In the **Action Launcher** on her record, launch the **Order CheckBook** intake.

> ⚠️ Launch it **through the Action Launcher on the record**, not from a raw URL — the OmniScript needs the record context the Action Launcher provides.

3. Step through the OmniScript (confirm the client's data loads — that proves the OmniScript → Integration Definition → Named Credential chain works) and **submit**.
4. Confirm a **Case** was created and the **Flow Orchestration** kicked off for `ProcessOrderChkbkRqst` (check the Orchestration Work Items related list).

Expected result — this IS success: the orchestration is `Status = InProgress` and parked at `CurrentStage = NotifyCaseOwnerOfFailure`. That is **not** a failure — it means the process ran and called the mock endpoint, exactly as expected before real core-banking is wired. ✅

Part 8 — Turn on the BSA Employee Agent + Checkbook Ordering topic

1. Go to **Setup** → **Salesforce Go** → **Banking Service Employee Assistance** and confirm the feature is **ON**.
2. **Build an Agent** from this Go step (this routes through the sanctioned Employee-Agent template — do it here, not from a cold "New Agent" in Studio).
3. In the agent, go to **Topics** and confirm/enable the **Checkbook Ordering** topic (it's an Employee-Agent-only topic; it's backed by the Order CheckBook process you just built).
4. **Commit Version** → **Activate.**

> ⚠️ **If Commit fails** with `Failed to find source action 'fscp_agent_pc__…'` on the **CreditLimitManagement**, **AccountClosure**, or **CardDeliveryManagement** subagents: **remove those 3 subagents** in Agentforce Builder, then **Commit** → **Activate**. That's a known platform gap on those 3 newest subagents and it does **not** affect Checkbook Ordering.

Expected result: the Banking Service Employee Assistance agent is **Active** with the Checkbook Ordering topic.

Part 9 — Test the agent

1. Open the agent's **Conversation Preview** (in Agentforce Builder).
2. Ask, conversationally: *"I need to order a checkbook for Rachel Morris."*
3. Let the agent walk the intake.

Expected result: the agent fires the Checkbook Ordering action and **creates a Case** — the same outcome as the Action Launcher test in Part 7, now driven by conversation.

You are done when: the Action Launcher intake creates a Case + orchestration (Part 7), and the agent creates a Case conversationally (Part 9).

Troubleshooting

Symptom	Cause	Fix
Country dropdown in the intake is empty	State/Country picklists not enabled	Pre-baked in the workshop org — flag your presenter if you see this
Intake errors calling the service / empty picker	Named Credential missing	`FSC_fsc_integrations_V1_0_0` is pre-baked — flag your presenter
"provide a productId / active svcCatalogItemDefApiName" at submit	Product not Active, or Product Id not bound in **both** places	Confirm the Product is Active (Part 4) and the Id is in **both** the map IP element **and** the orchestration Start condition (Part 5)
OmniScript "active" but won't run	Activated via API instead of the Designer	Re-activate **version 2 in the OmniStudio Designer** (Part 2)

Symptom	Cause	Fix
Orchestration doesn't fire on submit	Wrong/misspelled orchestration API name, or not activated	Confirm the name is exactly `ProcessOrderChkbkRqst` and it's Active (Parts 3, 5b)
Agent Commit fails on Credit Limit / Account Closure / Card Delivery	Platform prebuilt-action gap on those 3 subagents	Remove those 3 subagents → Commit → Activate (Part 8)
Agent Commit fails: "no access to agent templates for AgentforceEmployeeAgent"	Org missing the entitlement	Pre-baked in the workshop org — flag your presenter (not fixable by a permission set)

Notes

- Shipped Salesforce product on the standard object model — no managed package, no custom code.
- **Green toasts are not proof — the runtime is.** Every "Expected result" above is a real record/state to confirm.
- Want to automate all of this and do it for *every* process on your own org? That's the take-home CTA kit — ask your presenter for the **Agentforce Vibes** walkthrough.