Use-Case Cookbook

Ten Salesforce–ServiceNow Flows You Can Build Today

NexGen Architects | July 2026

Ten Salesforce–ServiceNow flows you can build today, in Flow Builder, with no code and no middleware. Every recipe uses only operations that ship in ServiceNow Flow Connector v1.0.0.

BEFORE YOU BUILD: THREE RULES THAT APPLY TO ALL TEN

Persist the sys_id	Every create operation returns the new ServiceNow record's `sys_id`. Write it to a custom field on the Salesforce record immediately. Without it you have no durable handle for any later update, and every subsequent flow is reduced to fuzzy-matching on description text.
Go async for callouts	A record-triggered flow cannot make a callout in the same transaction as its DML. Put ServiceNow actions on the asynchronous path, or the flow fails with a "callout not allowed" error.
Respect the 100-callout limit	Salesforce permits 100 callouts per transaction. Any flow that loops and calls ServiceNow per record will fail on the 101st. Filter hard before the loop.

Recipe 1 — Escalate a Salesforce Case to a ServiceNow Incident

The classic: support hands a technical problem to IT.

TRIGGER Record-Triggered Flow on Case, when `Escalate_to_IT__c` becomes true (async path).

OPERATIONS `createIncident`

1. Call `createIncident`, mapping the `incident` body: `short_description` from the Case Subject, `description` from the Case Description, `urgency` and `impact` derived from Case Priority, and `caller_id` from a ServiceNow user `sys_id`.
2. Store the returned `sys_id` and `number` on the Case in `ServiceNow_Incident_Id__c` and `ServiceNow_Incident_Number__c`.

Recipe 2 — Two-way comment sync between Case and Incident

Keep the customer-facing conversation and the IT conversation in step.

TRIGGER Two flows: a Record-Triggered Flow on Case Comment / Feed Item (Salesforce → ServiceNow), and a Scheduled Flow every 15 minutes (ServiceNow → Salesforce).

OPERATIONS `patchIncident`, `listJournalEntries`

1. **Outbound:** when an agent posts a comment, call `patchIncident` on the stored `sys_id` with a body of `{"comments": "..."}`. Setting `comments` is how you create a customer-visible journal entry; use `work_notes` for an internal note. You never write to the journal table directly.

2. **Inbound:** on a schedule, call `listJournalEntries` filtered with `sysparm_query` on `element_id` (the incident `sys_id`) and `sys_created_on` later than the last sync, then post new entries to the Case feed.

Recipe 3 — Mirror ServiceNow incident status back onto the Case

Close the Case automatically when IT resolves the incident.

TRIGGER Scheduled Flow, every 15–30 minutes.

OPERATIONS `listIncidents`

1. Call `listIncidents` with `sysparm_query` filtering on recent changes — for example `sys_updated_on>javascript:gs.hoursAgo(1)` — plus `sysparm_fields=sys_id,number,state,resolved_at` and an explicit `sysparm_limit`.
2. Query the matching Cases by `ServiceNow_Incident_Id__c`, map the ServiceNow state to your Case Status, and update.

Recipe 4 — Raise a Change Request when an Opportunity closes

Won deals that require provisioning shouldn't wait on a handover email.

TRIGGER Record-Triggered Flow on Opportunity, when Stage becomes Closed Won (async path).

OPERATIONS `createChangeRequest`, `getChangeRequest`

1. Call `createChangeRequest` with `short_description` naming the account and product, a `description` carrying the provisioning detail, `type` (Normal or Standard), and `start_date` / `end_date` for the planned window.
2. Store the returned `sys_id` and `number` on the Opportunity so account teams can see the change reference.

Recipe 5 — Deflect cases with ServiceNow Knowledge articles

Surface the answer before a human ever picks up the case.

TRIGGER Screen Flow on the Case page (agent-facing), or an Experience Cloud self-service screen.

OPERATIONS `listKnowledgeArticles`, `getKnowledgeArticle`

1. Call `listKnowledgeArticles`, passing the Case Subject or the user's typed question as the search query.
2. Display the top results. When one is selected, call `getKnowledgeArticle` to retrieve the full body and render it in the flow screen.
3. Offer "This solved it" — and if clicked, close the Case without ever opening an incident.

Recipe 6 — Sync a Salesforce Asset to a CMDB Configuration Item

Make the CMDB reflect what was actually sold and installed.

TRIGGER Record-Triggered Flow on Asset, on create or update (async path).

OPERATIONS `listCMDBCIs`, `createCMDBCI`, `patchCMDBCI`

1. Call `listCMDBCIs` for the relevant `className` (for example `cmdb_ci_computer`), with a `sysparm_query` matching on serial number to check whether the CI already exists.
2. If no match, call `createCMDBCI`. If matched, call `patchCMDBCI` to update it. Store the CI `sys_id` back on the Asset.

Recipe 7 — Provision employee onboarding via Service Requests

New hire in Salesforce triggers laptop, accounts, and access in ServiceNow.

TRIGGER Record-Triggered Flow on your onboarding object (or Contact / custom Employee record) when status becomes Hired.

OPERATIONS `createServiceRequest`, `createServiceRequestItem`, `listServiceRequestItems`

1. Call `createServiceRequest` to open the parent request (`sc_request`) for the new starter. Keep its `sys_id`.
2. For each item needed — laptop, phone, system access — call `createServiceRequestItem` (`sc_req_item`), linking each to the parent request's `sys_id`.
3. Track progress by calling `listServiceRequestItems` filtered on the parent request, and surface the fulfilment status on the Salesforce record.

Recipe 8 — Dispatch and track fulfilment tasks

Give field engineers their work, and see when it's done.

TRIGGER Record-Triggered Flow on Work Order or Case; plus a Scheduled Flow to poll status.

OPERATIONS `createTask`, `patchTask`, `listTasks`

1. Call `createTask` (`sc_task`) with `short_description`, `assignment_group`, and `due_date`.
2. Poll with `listTasks` filtered by `sysparm_query` on `state` to detect completion, and update the Salesforce record.
3. Use `patchTask` to push changes the other way — reassignment, a revised due date, or a note added by the Salesforce agent.

Recipe 9 — Audit attachments on escalated incidents

Know what evidence is attached to an incident, without leaving Salesforce.

TRIGGER Screen Flow on the Case, or a Scheduled Flow for compliance reporting.

OPERATIONS `listAttachmentsMetadata`, `getAttachmentMetadata`, `deleteAttachment`

1. Call `listAttachmentsMetadata` with a `sysparm_query` on `table_sys_id` equal to the linked incident's `sys_id`.
2. Display file names, sizes, content types, and upload dates on the Case, so an agent can see what IT holds.
3. Where a retention policy applies, call `deleteAttachment` to remove files past their retention date.

Recipe 10 — Build a CMDB impact map from Salesforce data

Record which configuration items depend on which — so IT can see blast radius.

TRIGGER Record-Triggered Flow when an Asset relationship is created in Salesforce, or a Scheduled Flow reconciling the two systems.

OPERATIONS `createCMDBCIRelation`, `deleteCMDBCIRelation`, `getCMDBCI`

1. Resolve both CIs to their `sys_id` values with `getCMDBCI` or `listCMDBCIs`.
2. Call `createCMDBCIRelation` against the parent CI's class and `sys_id`, supplying the target CI and the relationship type (for example Depends on::Used by).
3. When the relationship no longer holds, call `deleteCMDBCIRelation` with the relationship's own `sys_id`.

Frequently asked questions

How do I create a ServiceNow incident from a Salesforce Case without code?

Build a Record-Triggered Flow on Case that calls the connector's `createIncident` action on its asynchronous path. Map the Case Subject to `short_description`, the Description to `description`, and derive `urgency` and `impact` from Case Priority. Then store the returned `sys_id` on the Case in a custom field so later flows can update the same incident. Two things break this most often: forgetting the async path (a record-triggered flow cannot call out in the same transaction as its DML), and omitting a duplicate guard, so the flow opens a second incident when the trigger condition is re-met.

How do I sync comments between a Salesforce Case and a ServiceNow incident?

Push comments by calling `patchIncident` with the `comments` field set, and pull them back with `listJournalEntries` on a Scheduled Flow. Setting `comments` creates a customer-visible journal entry; `work_notes` creates an internal one. You never write to the journal table directly. The critical detail is the echo loop: without a guard, a comment you push into ServiceNow is read back on the next poll and re-posted to Salesforce indefinitely. Prevent it by prefixing synced comments with a marker such as `[SF]` and skipping those, or by filtering out journal entries authored by the integration user.

Can a Salesforce Flow react when a ServiceNow incident is updated?

Only by polling — the connector is outbound only, so ServiceNow cannot trigger a Salesforce flow through it. The standard pattern is a Salesforce Scheduled Flow running every 15–30 minutes that calls `listIncidents` with a `sysparm_query` filtering on `sys_updated_on` within the recent window, then updates the matching Cases. Keep the query window slightly wider than the schedule interval so no update falls between runs, and always set `sysparm_limit`. If you need true real-time push, configure a ServiceNow Business Rule to call a Salesforce inbound endpoint — that is built on the ServiceNow side, outside this connector.

How do I use ServiceNow Knowledge articles for case deflection in Salesforce?

Build a Screen Flow that calls `listKnowledgeArticles` with the customer's question as the search query, then `getKnowledgeArticle` to display the full article body. Offer a "This solved it" action that closes the Case without opening an incident. Note that Knowledge is read-only in v1.0.0 — you can search and read articles but not author them from Salesforce. Also verify visibility: the integration user only sees knowledge bases its ServiceNow roles permit, so an article visible to an agent in the ServiceNow UI may be invisible to the connector.

Why does my flow fail with a "callout not allowed" error?

A record-triggered flow cannot make a callout in the same transaction as its database operation — move the ServiceNow action to the flow's asynchronous path. This is a Salesforce platform rule, not a connector limitation: callouts are not permitted after uncommitted DML. In Flow Builder, add the ServiceNow action under **Run Asynchronously** rather than in the immediate path. Every record-triggered recipe in this cookbook depends on getting this right.

How many ServiceNow records can one Salesforce flow process at once?

Fewer than you would expect: Salesforce allows 100 callouts per transaction, and each connector operation uses one. A flow looping over a collection and calling ServiceNow per record fails on the 101st. There is no batch endpoint in the connector. Watch for this in composite recipes — an employee-onboarding flow that creates one service request plus ten requested items consumes eleven callouts per new starter, so a bulk import of fifty starters in a single transaction will breach the limit. Filter the collection before the loop, use asynchronous paths, or process in scheduled batches.

Should I use PATCH or PUT to update a ServiceNow record from Flow?

Use PATCH. It updates only the fields you supply, leaving everything else untouched. PUT replaces the entire record, and any field you omit may be cleared — including fields populated by ServiceNow-side automation, assignment rules, or other integrations. Because the damage is silent, choosing PUT by accident is one of the more painful mistakes available here. Reserve PUT for the rare case where you genuinely intend to overwrite a record wholesale.

Can I attach a Salesforce file to a ServiceNow incident?

Not in version 1.0.0 — the connector supports attachment metadata only, not binary upload or download. You can list attachments on an incident, read their metadata (file name, size, content type, upload date), and delete them, but you cannot transfer file bytes in either direction. The two practical workarounds are to use an Apex callout to `POST /api/now/attachment/file` for the binary transfer, or to include a link to the Salesforce file in the incident description rather than copying the file itself.