Supported ServiceNow Operations Matrix

All 45 Flow operations, mapped to their REST endpoints

NexGen Architects | July 2026

Every one of the 45 operations the ServiceNow Flow Connector exposes to Salesforce Flow Builder — mapped to its HTTP method and REST endpoint — across Incidents, Change Requests, Service Requests, Requested Items, Tasks, CMDB, Attachments, Journal Entries, Knowledge, and Users. Includes an explicit statement of what version 1.0.0 does not do.

KEY FACTS

Total operations	45, all callable declaratively from Flow Builder with no Apex.
Full CRUD domains	Incidents, Change Requests, Service Requests, Requested Items, and Catalog Tasks — each with Create, Read, List, Update (PATCH and PUT), and Delete.
CMDB	7 operations against the CMDB Instance API, including create and delete of CI relationships.
Read-only domains	Knowledge articles, Journal entries (comments and work notes), and Users.
Attachments	Metadata only — list, get, and delete. Binary upload and download are not in v1.0.0.
Direction	Salesforce → ServiceNow (outbound) only. The connector does not receive inbound calls from ServiceNow.

How to read this matrix

Operation is the exact name you will see in the Flow Builder action list. **Method** and **Endpoint** are the ServiceNow REST call the operation makes — useful for confirming role requirements and for testing a call with `curl` before you build the flow.

Two conventions apply throughout. **PATCH** updates only the fields you supply, leaving everything else untouched — this is what you want in almost every case. **PUT** replaces the whole record, and any field you omit may be cleared. Choose PATCH unless you have a deliberate reason to replace a record wholesale.

The complete operation set exposed by ServiceNow Flow Connector v1.0.0. All 45 operations.

Flow operation	Method	ServiceNow endpoint
INCIDENTS — TABLE `INCIDENT` · FULL CRUD · 6 OPERATIONS		
`createIncident`	POST	`/api/now/table/incident`
`listIncidents`	GET	`/api/now/table/incident`
`getIncident`	GET	`/api/now/table/incident/{sys_id}`
`patchIncident`	PATCH	`/api/now/table/incident/{sys_id}`

Flow operation	Method	ServiceNow endpoint
`putIncident`	PUT	`/api/now/table/incident/{sys_id}`
`deleteIncident`	DELETE	`/api/now/table/incident/{sys_id}`
CHANGE REQUESTS — TABLE `CHANGE_REQUEST` · FULL CRUD · 6 OPERATIONS		
`createChangeRequest`	POST	`/api/now/table/change_request`
`listChangeRequests`	GET	`/api/now/table/change_request`
`getChangeRequest`	GET	`/api/now/table/change_request/{sys_id}`
`patchChangeRequest`	PATCH	`/api/now/table/change_request/{sys_id}`
`putChangeRequest`	PUT	`/api/now/table/change_request/{sys_id}`
`deleteChangeRequest`	DELETE	`/api/now/table/change_request/{sys_id}`
SERVICE REQUESTS — TABLE `SC_REQUEST` · FULL CRUD · 6 OPERATIONS		
`createServiceRequest`	POST	`/api/now/table/sc_request`
`listServiceRequests`	GET	`/api/now/table/sc_request`
`getServiceRequest`	GET	`/api/now/table/sc_request/{sys_id}`
`patchServiceRequest`	PATCH	`/api/now/table/sc_request/{sys_id}`
`putServiceRequest`	PUT	`/api/now/table/sc_request/{sys_id}`
`deleteServiceRequest`	DELETE	`/api/now/table/sc_request/{sys_id}`
REQUESTED ITEMS — TABLE `SC_REQ_ITEM` · FULL CRUD · 6 OPERATIONS		
`createServiceRequestItem`	POST	`/api/now/table/sc_req_item`
`listServiceRequestItems`	GET	`/api/now/table/sc_req_item`
`getServiceRequestItem`	GET	`/api/now/table/sc_req_item/{sys_id}`
`patchServiceRequestItem`	PATCH	`/api/now/table/sc_req_item/{sys_id}`
`putServiceRequestItem`	PUT	`/api/now/table/sc_req_item/{sys_id}`
`deleteServiceRequestItem`	DELETE	`/api/now/table/sc_req_item/{sys_id}`
CATALOG TASKS — TABLE `SC_TASK` · FULL CRUD · 6 OPERATIONS		
`createTask`	POST	`/api/now/table/sc_task`
`listTasks`	GET	`/api/now/table/sc_task`
`getTask`	GET	`/api/now/table/sc_task/{sys_id}`
`patchTask`	PATCH	`/api/now/table/sc_task/{sys_id}`
`putTask`	PUT	`/api/now/table/sc_task/{sys_id}`

Flow operation	Method	ServiceNow endpoint
`deleteTask`	DELETE	`/api/now/table/sc_task/{sys_id}`
CMDB — CMDB INSTANCE API · 7 OPERATIONS · NO CI DELETE (SEE LIMITATIONS)		
`createCMDBCI`	POST	`/api/now/cmdb/instance/{className}`
`listCMDBCIs`	GET	`/api/now/cmdb/instance/{className}`
`getCMDBCI`	GET	`/api/now/cmdb/instance/{className}/{sys_id}`
`patchCMDBCI`	PATCH	`/api/now/cmdb/instance/{className}/{sys_id}`
`putCMDBCI`	PUT	`/api/now/cmdb/instance/{className}/{sys_id}`
`createCMDBCIRelation`	POST	`/api/now/cmdb/instance/{className}/{sys_id}/relation`
`deleteCMDBCIRelation`	DELETE	`/api/now/cmdb/instance/{className}/{sys_id}/relation/{rel_sys_id}`
ATTACHMENTS — METADATA ONLY · 3 OPERATIONS · NO BINARY UPLOAD/DOWNLOAD		
`listAttachmentsMetadata`	GET	`/api/now/attachment`
`getAttachmentMetadata`	GET	`/api/now/attachment/{sys_id}`
`deleteAttachment`	DELETE	`/api/now/attachment/{sys_id}`
JOURNAL ENTRIES — COMMENTS & WORK NOTES · READ-ONLY · 2 OPERATIONS		
`listJournalEntries`	GET	`/api/now/table/sys_journal_field`
`getJournalEntry`	GET	`/api/now/table/sys_journal_field/{sys_id}`
KNOWLEDGE — KNOWLEDGE MANAGEMENT API · READ-ONLY · 2 OPERATIONS		
`listKnowledgeArticles`	GET	`/api/sn_km_api/knowledge/articles`
`getKnowledgeArticle`	GET	`/api/sn_km_api/knowledge/articles/{id}`
USERS — TABLE `SYS_USER` · READ-ONLY · 1 OPERATION		
`listUsers`	GET	`/api/now/table/sys_user`

How to add a comment or work note

This is the most common point of confusion in the matrix, so it is worth stating plainly. Journal entries are **read-only** in this connector, and that reflects how ServiceNow itself works — you do not create a comment by writing to the `sys_journal_field` table.

Instead, you add a comment or work note by updating the parent record and setting the `comments` field (customer-visible) or the `work_notes` field (internal). ServiceNow creates the journal entry for you. To add a customer-visible comment to an incident, call `patchIncident` with a body of:

```
{
  "comments": "Update from Salesforce: customer confirmed the issue persists."
}
```

To add an internal work note instead, use `work_notes`. Then use `listJournalEntries` to read the resulting thread back into Salesforce. This create-by-update, read-by-journal pattern is how bidirectional comment sync is built — see the Use-Case Cookbook, recipe 2.

Querying and filtering: sysparm parameters

Every `list` operation accepts ServiceNow's standard query parameters. These are the four that matter in practice:

The query parameters you will use in almost every list operation.

Parameter	Purpose	Example
`sysparm_query`	Filter records using ServiceNow's encoded query syntax.	`active=true^priority=1`
`sysparm_limit`	Cap the number of rows returned. Always set this.	`50`
`sysparm_offset`	Skip rows — the basis of pagination.	`50`
`sysparm_fields`	Return only named fields. Reduces payload size substantially.	`sys_id,number,state`

Not supported in version 1.0.0

We would rather you discover these here than three days into a build. The following are not available in v1.0.0:

Known limitations of v1.0.0, and the recommended workaround for each.

Not supported	Detail and workaround
Attachment binary upload or download	The three attachment operations handle metadata only — listing, retrieving details, and deleting. Uploading a file to ServiceNow (`POST /api/now/attachment/file`) and downloading file content are not exposed. **Workaround:** use an Apex callout for the binary transfer, or share a link to the Salesforce file instead of copying the bytes.
Deleting a CMDB configuration item	CMDB supports create, read, list, and update of CIs, and both create and delete of CI relationships — but there is no operation to delete the CI itself. **Workaround:** retire the CI by updating its lifecycle state (for example setting `install_status` to Retired), which is generally better CMDB hygiene than deletion anyway.
Creating or updating Knowledge articles	Knowledge is read-only: you can list and retrieve articles, which is what case deflection needs. Authoring articles from Salesforce is not supported.
Writing directly to journal entries	By design — and consistent with ServiceNow. Add comments and work notes by PATCHing the parent record's `comments` or `work_notes` field, as described above.
Creating or updating ServiceNow users	`listUsers` is read-only, and there is no get-user-by-`sys_id` operation. **Workaround:** retrieve a specific user with `listUsers` plus a `sysparm_query` such as `email=jane@acme.com` .
Arbitrary or custom tables	The connector is scoped to the ten domains in this matrix. There is no generic "query any table" operation, so custom tables and other out-of-box tables such as `problem` are not reachable in v1.0.0.

Not supported	Detail and workaround
Inbound ServiceNow → Salesforce triggers	The connector is outbound only: Salesforce calls ServiceNow. It cannot receive events from ServiceNow. **Workaround:** to react to ServiceNow changes, either poll from a Salesforce Scheduled Flow (see Use-Case Cookbook, recipe 3) or configure a ServiceNow Business Rule to push to a Salesforce inbound endpoint — the latter is outside the connector's scope.
Bulk or batch operations	Each operation is a single REST call against a single record or query. There is no multi-record batch endpoint. Be mindful of the Salesforce per-transaction callout limit (100) when looping.

Frequently asked questions

How many ServiceNow operations does the ServiceNow Flow Connector support?

Version 1.0.0 supports 45 operations. They break down as: 6 each for Incidents, Change Requests, Service Requests, Requested Items, and Catalog Tasks (30 in total, each a full CRUD set); 7 for CMDB; 3 for Attachment metadata; 2 for Journal entries; 2 for Knowledge articles; and 1 for Users. All 45 are callable declaratively from Salesforce Flow Builder without Apex.

Can the ServiceNow Flow Connector create an incident in ServiceNow from Salesforce?

Yes. The `createIncident` operation issues `POST /api/now/table/incident` and returns the new record, including its `sys_id`. Incidents have a complete CRUD set: `createIncident`, `getIncident`, `listIncidents`, `patchIncident`, `putIncident`, and `deleteIncident`. Store the returned `sys_id` on the originating Salesforce record so later flows can update the same incident.

How do I add a comment or work note to a ServiceNow incident from Salesforce Flow?

Call `patchIncident` and set the `comments` field for a customer-visible comment, or `work_notes` for an internal note — do not try to write to the journal table directly. ServiceNow creates the journal entry itself in response to the field update. This mirrors how ServiceNow works natively. To read the resulting comment thread back into Salesforce, use `listJournalEntries`, which is read-only. The same pattern works for change requests, service requests, requested items, and catalog tasks.

Can the ServiceNow Flow Connector upload file attachments to ServiceNow?

No. Version 1.0.0 handles attachment metadata only — `listAttachmentsMetadata`, `getAttachmentMetadata`, and `deleteAttachment`. Uploading binary file content and downloading file bytes are not supported. If you must move files, use an Apex callout to `POST /api/now/attachment/file`, or share a Salesforce file link in the incident description rather than copying the bytes across.

What is the difference between the PATCH and PUT operations in the connector?

PATCH updates only the fields you supply; PUT replaces the entire record and may clear any field you omit. Use PATCH — for example `patchIncident` — for virtually all updates, since it is safe against fields modified by other systems. Reserve PUT for the rare case where you genuinely intend to overwrite a record wholesale. Choosing PUT by accident is a common way to silently wipe fields that ServiceNow-side automation had populated.

Can the connector query custom ServiceNow tables, or tables like Problem?

No. Version 1.0.0 is scoped to ten specific domains and offers no generic "query any table" operation. The supported tables are `incident`, `change_request`, `sc_request`, `sc_req_item`, `sc_task`, `sys_journal_field`, `sys_user`, plus the CMDB Instance API, the Attachment API, and the Knowledge API. Custom tables and other out-of-box tables such as `problem` are not reachable. If you need one, contact us at hello@nexgenarchitects.com — table coverage is driven by customer demand.

Can ServiceNow trigger a Salesforce Flow when an incident changes?

Not through this connector — it is outbound only, meaning Salesforce calls ServiceNow and never the reverse. There are two standard ways to achieve the effect. The first is polling: a Salesforce Scheduled Flow calls `listIncidents` with a `sysparm_query` filtering on `sys_updated_on` to pick up recent changes. The second is push: configure a ServiceNow Business Rule or Flow Designer action to call a Salesforce inbound endpoint, which is built on the ServiceNow side and lies outside this connector's scope.

How do I filter which ServiceNow records a list operation returns?

Use `sysparm_query` with ServiceNow's encoded query syntax, and always pair it with `sysparm_limit`. For example, `active=true^priority=1` returns active priority-1 records. The reliable way to build a complex filter is to construct it in the ServiceNow list view, then right-click the breadcrumb and choose **Copy query** to obtain the exact encoded string. Add `sysparm_fields` to return only the fields you need — an unbounded list call against a large instance can return enough records to breach Salesforce heap limits and fail the flow.

Are there limits on how many ServiceNow records a flow can process?

Yes — Salesforce allows a maximum of 100 callouts per transaction, with a 120-second cumulative timeout, and each connector operation consumes one callout. A flow that loops over a collection calling ServiceNow once per record will therefore fail at the 101st record. The connector has no batch endpoint. For high-volume work, filter the collection before the loop, run the callouts in an asynchronous path, or process the records in scheduled batches.