CERTIFICATION GUIDE

ServiceNow

# CIS-SAM

*The Implementation-First Guide*

**Software Asset Management Professional**

**Configuration · Compliance · Reconciliation**

All five exam domains · 60-question practice exam

**R. PATHAK**

# Copyright

ServiceNow CIS-SAM: The Implementation-First Guide

Software Asset Management Professional — Configuration, Compliance, and the CIS-SAM Exam



Published by Rahul Pathak. First edition, 2026.



For permission requests or corrections, contact the author via linkedin.com/in/rahul-pathak-b7a65b244

# Disclaimer

This is an independent, unofficial study guide. It is not affiliated with, endorsed by, or sponsored by ServiceNow, Inc. ServiceNow, the ServiceNow logo, Now Platform, and related marks are trademarks of ServiceNow, Inc.

Exam domain names, weightings, and sample questions referenced in this book are drawn from ServiceNow's publicly published exam specification for the Certified Implementation Specialist – Software Asset Management (SAM) Professional exam. ServiceNow updates this specification with each platform release; readers should confirm the current version through their Now Learning account before final exam preparation.

Screens, field names, and workflows described reflect the SAM Professional application as commonly deployed at the time of writing and may differ slightly by release version and customer configuration.

# Introduction: Why This Book Exists

Search for a CIS-SAM study guide on Amazon and you will find exactly one author with anything published, and what they have published is a questions bank — a list of exam-style items with answer keys, no explanation of why the answer is correct or how the underlying ServiceNow application actually works. Beyond that, the available material is a scatter of brain-dump sites that reprint the same recycled questions under different names.

That gap is not an accident. CIS-SAM is a narrow, specialist certification — a genuine Implementation Specialist exam, not a general knowledge test. It does not ask whether you understand software asset management as a discipline. It asks whether you can configure the SAM Professional application: how a reconciliation actually runs, what a normalization status means, how an entitlement import fails and why, how a license metric gets calculated on the workbench. That is product configuration knowledge, and it is much harder to write convincingly than general theory, which is exactly why nobody has done it well.

This book is written by someone who has implemented SAM Professional and HAM Professional directly, not someone summarizing product documentation from the outside. Every chapter is organized against ServiceNow's own published exam blueprint, domain by domain, so nothing here is filler chosen because it was easy to write.

# Exam Overview

The Certified Implementation Specialist – Software Asset Management (SAM) Professional exam consists of 60 questions, a mix of multiple-choice (single answer) and multiple-select (select all that apply, no partial credit) items. There is no separate practical/hands-on component — everything is tested through scenario-based questions.

## Exam domains and weighting

| Domain | Weight |
|---|---|
| 1. Software Asset Core Overview & Fundamentals | 14% |
| 2. Data Integrity — Attributes and Sources for the Data | 28% |
| 3. Practical Management of Software Compliance | 30% |
| 4. Operational Integration of Software Processes | 13% |
| 5. Extending SAM | 15% |

Domains 2 and 3 together account for 58% of the exam — more than half. This book weights its chapter depth the same way: the data-integrity and compliance-management sections are the longest in the book, deliberately, because that is where the exam actually lives.

Registration and eligibility now run through ServiceNow University rather than the older Webassessor/voucher system most existing community discussion still describes — ServiceNow moved its mainline certification exams to Pearson VUE in 2026 and replaced the voucher model with an eligibility model: completing the required training course unlocks eligibility to register, rather than issuing a redeemable voucher. Once registered, the exam must be completed within 90 days or the attempt is forfeited. Because this process changed recently, confirm the current registration steps directly on ServiceNow University before starting — the Exam Day & Registration Guide in the back matter covers this in full.

# Contents

*A working manuscript from an independent, unofficial guide — cross-check exam-registration specifics against ServiceNow University before relying on them.*

# Chapter 1

## What Software Asset Management Actually Solves

Two organizations each run roughly forty software vendors across a few thousand endpoints and several hundred SaaS subscriptions. Both get an audit notice from a major publisher on the same Monday morning.

At the first organization, the notice triggers three weeks of scrambling — pulling install logs out of half a dozen disconnected spreadsheets, chasing down proof-of-purchase emails from a procurement inbox nobody has fully indexed, and negotiating from a position where nobody in the room actually knows whether they are over-licensed or under-licensed until the vendor's own auditors tell them. At the second organization, the licensing position was already current before the letter arrived. The audit response goes out inside a day, because reconciliation between what is installed and what is owned is something that runs continuously, not something that gets assembled in a panic.

The difference between those two organizations has nothing to do with which software they use. It is whether Software Asset Management is a live, operating discipline, or a folder of spreadsheets nobody has opened since the previous audit.

## What SAM actually is

Software Asset Management is the practice of tracking three things and keeping them reconciled: what software an organization has the legal right to use, what software is actually installed and running across the estate, and the evidence trail — purchase orders, contracts, entitlement records — that proves the first two match, or flags it when they don't.

That definition sounds simple. In practice it hides three separate operational problems that rarely get solved by the same tool or the same team: discovering what is genuinely out there (which gets harder every year as SaaS, cloud instances, and shadow subscriptions multiply outside of IT's direct control), knowing precisely what you are entitled to run under often-ambiguous licensing terms, and reconciling the two on an ongoing basis rather than as a once-a-year fire drill.

## The three problems SAM is actually solving

1. Audit and compliance exposure. Enterprise software vendors audit their largest customers as a matter of routine, not exception, and the financial exposure from an unfavorable audit — true-up costs plus negotiated settlements — routinely runs into the high six or seven figures for organizations with any real scale. The exact size of that exposure varies enormously by vendor mix and organization size, and published benchmarks disagree with each other, so the figure that matters is the one an organization calculates against its own Tier 1 vendor contracts rather than any industry average.

2. Wasted spend. The most common failure mode is not one dramatic overspend but many small, duplicated ones: three departments independently subscribing to overlapping project-management or

collaboration tools, dormant licenses nobody reclaims when an employee leaves, and enterprise agreements sized for peak usage that never comes back down after a headcount reduction. None of this shows up as a single alarming number — it shows up as a steady drag that nobody notices until someone actually reconciles installations against entitlements line by line.

3. Operational and security visibility. Software that IT does not know about is software IT cannot patch, secure, or account for. A subscription that finance discovers only because a credit-card statement finally gets audited is also, from a security standpoint, an unmanaged endpoint nobody has been watching. This is the point where SAM stops being a licensing and finance topic and starts being a security one as well — a connection Chapter 12 returns to, where software running past its end-of-support date becomes a live security exposure rather than a licensing footnote.

## Unmanaged vs. SAM-managed: what actually changes

| Unmanaged software environment | SAM-managed environment |
|---|---|
| Audit notice triggers a multi-week scramble | Licensing position is current; response goes out in days |
| Duplicate SaaS tools bought independently by different teams | Central visibility catches overlap before renewal |
| Reconciliation happens once a year, under pressure | Reconciliation runs continuously, in the background |
| Nobody can say with confidence whether the org is over- or under-licensed | License position is a known number, not a guess |
| Unmanaged installs are also unmanaged security exposure | Discovery feeds both licensing and security visibility |

## Where this fits in ServiceNow, and why the exam is not a theory test

Everything above is true of Software Asset Management as a discipline, independent of any specific platform. The CIS-SAM exam does not test that discipline in the abstract. It tests whether you can make it operational inside ServiceNow specifically — how SAM Professional ingests discovery and import data, how it normalizes messy installation records into a clean catalog of products and models, how the reconciliation engine actually calculates a license position, and how that position flows into contracts, procurement, and remediation workflows elsewhere in the platform.

That is a deliberate design choice on ServiceNow's part, and it is why the existing study material on the market — question banks with answers but no explanation, or generic ITAM theory with no ServiceNow specifics — leaves candidates underprepared. Knowing what a license position is does not help you on exam day. Knowing what happens on the License Workbench when a reconciliation runs, and why a specific remediation option is or is not available afterward, does.

The next chapter goes inside the application itself — the roles, the core tables, and the process architecture that every later chapter in this book builds on.

### Key Takeaways

- SAM solves three distinct problems: audit/compliance exposure, wasted software spend, and operational/security visibility.
- Reconciliation that only happens once a year, under audit pressure, is not the same discipline as reconciliation that runs continuously.
- The CIS-SAM exam tests ServiceNow-specific implementation knowledge, not general ITAM theory — that distinction shapes every chapter in this book.
- Domains 2 and 3 (data integrity and compliance management) make up 58% of the exam and get the deepest treatment here accordingly.

# Chapter 2

## Inside the SAM Application — Roles, Data Model, Process Architecture

Chapter 1 made the case for what Software Asset Management solves. This chapter goes inside the application itself — who can do what, which tables actually hold the data, and how a request for software turns, step by step, into a documented license position. Every later chapter in this book assumes the architecture laid out here.

### The three SAM roles

SAM Professional access is built as a strict hierarchy: each higher role inherits everything the role below it can do, plus one additional capability.

| Role | What it adds |
|---|---|
| sam_user | Baseline access — view software models, installations, entitlements, and reconciliation results. Cannot run reconciliation or change configuration. |
| sam_admin | Everything sam_user has, plus the ability to run on-demand reconciliation and manage reclamation rules. This is the working role for day-to-day SAM administration. |
| sam_developer | Everything sam_admin has, plus permission to write or modify custom metric scripts and other server-side logic. sam_admin can read a custom metric script; only sam_developer can change one. |

A handful of supporting roles sit around these three and matter for the exam because they mark where SAM stops and another application starts: catalog_admin (Service Catalog configuration, relevant when software requests route through a catalog item), procurement_user (purchase orders and transfer orders), and ham_admin (the equivalent administrative role for Hardware Asset Management, which only appears once the HAM application is installed). Part 4 of this book returns to exactly where these boundaries sit.

### The two tables everything else is reconciled against

Two tables sit at the center of SAM Professional, and almost everything else in the application exists to populate, normalize, or reconcile them.

Software Installation [cmdb_sam_sw_install] holds what is actually running — Discovery populates this table automatically using discovery patterns (SAM Professional ships with support for common

ones such as SQL Server, Exchange, and Oracle Database out of the box, and more can be added), and manual or third-party imports can add to it as well.

Software Entitlement [alm_license] holds what the organization has the legal right to run — a point-in-time record of a purchase, tied to a Software Model and, where a Publisher Part Number was supplied, to the Discovery Map that part number resolves to. Chapter 6 covers Discovery Maps in full. This is the table that answers "what did we actually buy, and how many."

Reconciliation is the process that compares the two. It doesn't just count installs against entitlements — it also brings in the metric type (per-device, per-user, per-core, and so on) attached to each Software Model, which is why Chapter 8 spends as much time on license metrics as it does. Get the metric wrong and the reconciliation number is wrong even when the raw counts are right.

## From raw install to license position: the end-to-end flow

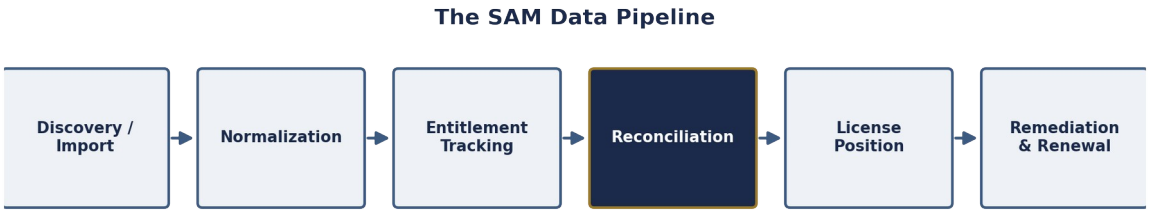


*Figure 2.1 — The SAM data pipeline. Reconciliation sits at the centre, but depends on every stage before it.*

| Step | What happens |
|---|---|
| 1. Discovery / Import | Raw installation and usage data lands in cmdb_sam_sw_install, via discovery patterns or manual/API import. |
| 2. Normalization | Discovery Models match messy raw data (varying product names, versions) to a clean Software Model. Chapter 5 covers this in depth. |
| 3. Entitlement tracking | Purchased licenses are recorded in alm_license, linked to the matching Software Model — and, via the Publisher Part Number, to its Discovery Map. |
| 4. Reconciliation | sam_admin runs it (on-demand or scheduled). Installs are compared against entitlements using the model's license metric. |
| 5. License Position | The output — a documented, point-in-time compliance status, viewable on the License Workbench. Chapter 9 covers reading and troubleshooting this. |

| Step | What happens |
|---|---|
| 6. Remediation & renewal | Over- or under-licensed positions feed into reclamation rules, procurement, and contract renewal. Part 3 and Part 4 pick this up. |

Notice that nothing in this pipeline is optional or skippable — a weak Discovery Model produces a weak license position even if every entitlement record is perfect, and a sloppy entitlement import produces a wrong compliance number even if Discovery is flawless. That's the practical reason Domains 2 and 3 carry 58% of the exam between them: the exam is built around this exact pipeline, and each domain maps to a stage of it.

### Key Takeaways

- Three roles, strictly hierarchical: sam_user (view only), sam_admin (adds reconciliation + reclamation rules), sam_developer (adds custom metric scripting).
- cmdb_sam_sw_install (what's running) and alm_license (what's owned) are the two tables everything else in SAM Professional serves.
- The pipeline — Discovery → Normalization → Entitlements → Reconciliation → License Position → Remediation — is the spine of the whole application, and of the exam's domain structure.
- A weak link anywhere in the pipeline produces a wrong license position, even if every other stage is done correctly.

# Chapter 3

## Asset Data Quality — Reconciliation Is Only as Good as Your CMDB

Every step in Chapter 2's pipeline assumes normalization already happened correctly. This chapter is about what happens when it hasn't — and why a discovery scan reporting near-complete coverage can still leave an organization with no real idea of its compliance position.

### Three tables that sound alike and aren't

A Software Model is a specific, curated version or configuration of software that the organization has purchased or made available — this is the canonical catalog entry that entitlements attach to. A Software Discovery Model [cmdb_sam_sw_discovery_model] is a completely different table: a record auto-created whenever Discovery finds something new in the environment, representing what was found, not what was purchased. Multiple discovery models can eventually map to a single software model once normalization resolves the differences between them (a product reported as "Adobe Acrobat DC" and one reported as "Acrobat Reader DC 23.1" might both settle on the same underlying model).

Installed software itself lands in a third table, Software Installation [cmdb_sam_sw_install], with a primary key built from Publisher, Display Name, and Version. Discovery uses that key to match each installation to a discovery model automatically. Confusing these three tables — model, discovery model, and installation — is one of the more common ways candidates lose easy points on this domain.

### The six normalization statuses

| Status | What it means |
|---|---|
| New | Not yet evaluated by the normalization engine. |
| Normalized | Fully matched on publisher, product, and version. Key fields are locked. |
| Manually Normalized | A normalization suggestion was accepted, or an editable field was corrected by hand. |
| Partially Normalized | Matched on publisher and product only — version remains editable. |
| Publisher Normalized | Matched on publisher alone — product and version remain editable. |
| Match Not Found | None of the three key fields matched a rule in the Content Library. Everything is editable. |

Version matching itself has a fallback rule worth remembering: the system first looks for an exact version match, and if none exists, it rounds down to the nearest major version — a discovered install at version 8.0.4 will fall back to matching against 8.0 if 8.0.4 has no rule of its own. Installations with "Security Update" in the name are deliberately excluded from getting their own discovery model, so patch noise doesn't clutter the catalog.

## Why "mostly normalized" is a dangerous phrase

A discovery scan that reports 95% coverage tells you almost nothing about compliance on its own. Reconciliation only compares records it can confidently interpret — a discovery model still sitting in Match Not Found, Publisher Normalized, or Partially Normalized status isn't wrong, exactly, but it also isn't being reconciled against anything yet. It doesn't register as compliant. It doesn't register as a violation either. It simply sits outside the calculation, invisible, until someone resolves it.

This happens even with mainstream, well-known software. Content Service — the ServiceNow team that maintains and publishes the normalization rules everything else in this chapter depends on — sometimes lags behind a new release. A widely deployed product can sit in a partially normalized state for weeks after launch, not because anything is misconfigured, but because the version-level rule for that specific release hasn't been published to the Content Library yet.

The practical lesson carries over directly from general ITAM practice: completeness is not a percentage game. What matters is not how much of the estate Discovery has found, but how much of what it found can actually be trusted in a reconciliation run. A smaller, fully normalized dataset produces a more defensible license position than a larger one with a long tail sitting in Match Not Found.

## Keeping normalization healthy

When a discovered value differs from what the Content Library suggests, ServiceNow surfaces a Normalization Suggestion with the discovered and suggested values side by side — publisher, product, version, and edition. Accepting a suggestion updates the discovery model and moves its status to Manually Normalized; rejecting one leaves the model exactly as it was, still flagged for review.

Normalization can also be reverted deliberately, resetting a model back to Match Not Found — useful when an automatic match turns out to be wrong and needs to be corrected from scratch rather than patched. Chapter 6 picks up the other half of this relationship: how Content Service actually builds and ships the rules that normalization runs against, and what to do when the rules an organization needs simply haven't been published yet.

### Key Takeaways

- Software Model, Software Discovery Model, and Software Installation are three distinct tables — mixing them up is a common source of easy mistakes.

- Six normalization statuses exist: New, Normalized, Manually Normalized, Partially Normalized, Publisher Normalized, and Match Not Found.
- Version matching falls back to the nearest major version when no exact match exists.
- Records that aren't fully normalized are invisible to reconciliation — not compliant, not a violation, just absent from the calculation.
- Discovery coverage percentage and usable, reconciled data are two different numbers. Only the second one tells you anything about compliance.