THE PLAYBOOK

The Takeover

How I inherited a stack of recurring reports with no handover, and kept a record good enough that the next person will never need one. The repeatable method behind that: run the old file once, list what was already broken, and write every update the moment it lands. Plus the honest chapter on when following the old file exactly is the right call.

Paul Prado Pacardo

Executive Assistant · Project Manager · Operations me.kabuhayan.app

Contents

A recording and an old file the handover that never came

1 **The old file is not the standard** why copying the old file copies its errors

2 **Copy everything on day one** save a copy of everything you touch, before you touch it

3 **Two documents, not one** the method, step by step, with the rules that hold it

4 **Write it the moment it lands** the log is the source of truth, not your memory

5 **Where takeovers go wrong** how takeovers fail, and the catch for each

6 **When not to run it** the cases where this method is the wrong move

Leave the handover you never got closing

The tool: the takeover checklist and the live tool for the day-one habit

OPENING

A recording and an old file

Within a few weeks, several weekly and monthly reports changed hands and landed on me. None of them came with notes. Each one arrived the same way:

> THE HANDOVER, IN FULL
>
> A short screen recording of someone running the report once, and the last finished file. No list of sources, no due day in writing, no word on who reads it.

That is enough to rebuild a recurring report, if you write the steps down the same day. But inherited files carry quiet breakage. Inside these ones: formulas pointing at columns that had been deleted, and summary blocks copied in from another client account. Every person who inherits work faces the same fork, usually without noticing it is there.

THE FORK IN THE ROAD

THE SAFE MOVE

Follow the old file exactly. Copy what the recording shows, send it on the usual day, change nothing. Nobody can fault you for doing it the way it was done. It also reproduces every old error under your name.

THE MOVE I MADE

Run it once, then list what was broken. Follow the steps literally on the first run, then write down every defect that was there before I arrived, separately, and ask before fixing any of it.

Figure: the decision that decides whose errors you end up owning.

This playbook is that method, plus the habit underneath it: a written log, kept as you go, so the work never needs a handover again. It ends with the honest chapter: when the safe move really is the right one, because a method that claims to always apply is a slogan.

CHAPTER 1

The old file is not the standard

An inherited file looks like a specification. It ran last month, it went out, nobody complained. The natural reading is: this is what correct looks like.

A finished file proves it was sent, not that it was right. Reports drift. A column gets deleted upstream and a formula quietly starts pulling from its neighbour. A block gets pasted in from another account to save time and never gets swapped back. Nobody notices, because nobody reads a recurring report as closely as the person building it.

The recording shows the method, not the defects. The trainer runs it the way they always have. If a number is typed by hand, the recording shows the typing but not where the number came from.

> THE BET
>
> If you follow the old file exactly, its errors become yours on the first send. If you run it once and list what was already broken, they stay where they started, and the owner decides what happens to them.

There is a second reason the list matters. A new person who quietly fixes things looks careful right up until a number changes that someone downstream relied on. Then the fix is a surprise, and surprises cost more trust than the error did. Writing it down first turns a silent fix into a question the owner answers once.

Do not be surprised if the already-broken list grows faster than the steps. That is the useful finding, not an embarrassment, and it is only visible because the two lists are kept apart.

CHAPTER 2

Copy everything on day one

Before any report, the first move with inherited work is a record of what you were handed. Not a fix. A copy.

The lesson came from a set of daily exception queues I stepped into, again with no notes on what had already been worked or who had already been contacted. Without that list, people who had already been contacted received a "first contact" message again. Nothing about the message was wrong. It was just the second one.

DAY ONE

Save every queue

A dated copy of every list, queue and file you touch, before you change anything in it.

DAY TWO

Compare, then act

Paste today against yesterday. Sort rows into new, carried, updated and resolved before calling anyone.

EVERY DAY

Keep a contacted list

One running list of who was reached, when and how. It ends the double introduction.

The day-one copy is cheap and cannot be made later. Every question that comes up in week three ("was this already open when you started?") is answered by that first snapshot or not answered at all.

> SEE IT YOURSELF
>
> The compare step is a free tool on the site: **me.kabuhayan.app/tool-queue-diff**. Paste yesterday's list and today's, and it sorts every row into new, carried, updated and resolved, with counts. It runs in the browser; nothing is uploaded.

The tool only works if the day-one copy exists. That is the habit; the tool just makes day two fast.

CHAPTER 3

Two documents, not one

The takeover produces two documents: the steps, and the list of things that were already broken when I got it. The sequence:

Step 01 **Collect every input**

The recording, the last two finished outputs, the data source, the recipients, the due day.

Step 02 **Write literal steps**

Watch the recording slowly. Each action becomes a step with a real file location, and every number typed by hand gets a note on where it came from.

Step 03 **Run it once, exactly as trained**

On the newest data, following your own steps and nothing else.

Step 04 **Explain every difference**

Where your run and the old output disagree, decide: did the data change, or was the old file wrong?

Step 05 **Keep the already-broken list apart**

Bad formulas, copied blocks, stale lookups. Listed, not fixed.

Step 06 **Send three things**

The finished run, the already-broken list, and only the questions that change the output.

Step 07 **Write the SOP and schedule it**

Steps plus a "what goes wrong" box seeded from the list. Set the recurring task for when the data lands. Version it again after the second run.

> THE RULE
>
> Follow the trainer literally on the first run; improvements come after approval. Never invent a value to make a check pass. A file is not final while any lookup shows an error. Ask only questions whose answer changes the output; everything else becomes a labelled assumption.

CHAPTER 4

Write it the moment it lands

Taking over work is half the job. The other half is leaving it so nobody has to take it over the way I did. That record is a running log, and I learned its rules by breaking them.

A long working session lost its morning detail. The counts had lived in my head and in a chat window, and the end-of-day report went out with placeholders. Worse, the "waiting on reply" total had been worked out by subtracting the day's fixes from yesterday's number. Each day's small error carried into the next, and the number drifted.

LOG FIELD	WHAT GOES IN IT
Time	Local time and the account's time, both
System and item	Where it happened, and which row or record
Status	The operator's own word: fixed, sent, approved
Note	Anything the next person would otherwise have to ask

> **Write each update as it lands,** and checkpoint the log to shared storage at midday and end of day.

> **Derive counts from log lines only.** Rebuild waiting lists name by name from the latest queue, never by subtraction.

> **Drafts are never reported as sent.** A draft is a draft until someone confirms it went.

> **Correct with a dated correction line,** never by deleting. The history of what changed is part of the record.

> STATED PLAINLY
>
> If a number cannot be traced to a line you wrote down, do not report it. Write "not recorded" instead. A gap you name is worth more than a number you guessed.

CHAPTER 5

Where takeovers go wrong

Every failure I have seen in inherited work is one of a handful, and each has a cheap catch if you build it in from the first day.

FAILURE	WHAT IT LOOKS LIKE	THE CATCH
Old errors copied forward	Your first send matches last month, mistakes included	Compare your run against the old output line by line
Silent fixes	A number changes and the owner hears it from someone else	The separate already-broken list, sent before any fix
Vague steps	"Update the summary" with no location	Every step names a file and a cell or column
Wrong day	The task fires before the data arrives	Schedule from when the source actually lands
Memory as record	Counts from recall, log lost on a reset	Log written as it happens, checkpointed twice a day
Erased history	Old lines deleted, so nobody can see what changed	Append-only, with dated status marks inline

The handoff I leave follows the same rule as the log: append-only. It opens with a dated "start here" checklist, then the rules that were corrected, what went wrong, what worked, the open state (drafts not confirmed sent, blocked items, who we are waiting on), and the last copy of every queue, row by row. Nothing inherited gets summarised away.

> THE TRAP TO AVOID
>
> Do not clean up the history to make the record look tidy. A handoff with the corrections removed reads well and tells the next person nothing about where the work is fragile.

Worth being precise about what this is evidence for. A few reports, a few queues, one back office. What generalises is the shape: copy first, run literally, list defects apart, log as you go. How long the already-broken list is, and how often it matters, will vary with how the work was kept before you.

CHAPTER 6

When not to run it

A method that claims to always apply is a slogan. There are real cases where the full takeover is too much, or the wrong thing entirely.

> **When the trainer is still in the room.** If the previous owner is a message away, a ten-minute walkthrough beats an hour with a recording. Ask, and write down what they say.

> **When the report is due today.** Run it exactly as the old file does, send it, and start the already-broken list tomorrow. A late report with a defect list is still a late report.

> **When the work is one-off.** An SOP, a recurring task and a versioned record for something that runs once is overhead. Keep the log; skip the rest.

> **When the owner has said "do it exactly as before".** Then do exactly that. Keep the list anyway, privately, and raise it when they ask what you have noticed.

Even in these cases, half the method survives: **save the day-one copy and keep the log.** Neither costs more than a few minutes, and neither can be recreated after the fact.

> THE TEST TO APPLY
>
> Will someone else have to run this after me? If yes, write the steps and the already-broken list. If no, copy what you were handed, log what you did, and move on.

CLOSING

Leave the handover you never got

Every takeover starts with one quiet decision: copy the old file, or run it once and write down what was already wrong. Everything in this playbook is downstream of choosing the second, and of asking before fixing.

The log is the same decision made every day. Write it as it happens, count only what you wrote, mark drafts as drafts, and correct with a dated line instead of an eraser.

Done that way, the person after you inherits a record instead of a recording. That is the whole point: the next takeover should not need this playbook.

Paul Prado Pacardo

Senior Executive Assistant, Operations and Project Manager. Ten years supporting C-level leadership. Available for remote work.

me.kabuhayan.app · linkedin.com/in/paulpacardo · pacardopaul18@gmail.com

THE TOOL

The tool: the takeover checklist

Run this the day recurring work lands on you with no notes. If any box will not tick, chapter 6 applies.

DAY ONE

- [ ] A dated copy saved of every queue, list and file I touched
- [ ] Recording, last two outputs, data source, recipients and due day collected
- [ ] Literal steps written, each with a real file location
- [ ] A contacted list started

FIRST RUN

- [ ] Run done exactly as trained, on the newest data
- [ ] Every difference explained: data change or old error
- [ ] Already-broken list kept separate, nothing fixed yet
- [ ] Only output-changing questions sent; the rest written as assumptions

EVERY DAY AFTER

- [ ] Each update logged as it lands, in the operator's words
- [ ] Counts derived from log lines; gaps written "not recorded"
- [ ] Drafts never reported as sent; corrections dated, never deleted
- [ ] Handoff appended, with the last copy of every queue

THE LIVE ARTIFACT

The day-two compare is free on the tools page: **me.kabuhayan.app/tool-queue-diff**, with the rest of the set at me.kabuhayan.app/tools. Paste two lists and it sorts every row into new, carried, updated and resolved.