THE PLAYBOOK

The Diff

How I took over several daily queues with no handover and stopped introducing myself twice to the same people. The repeatable method behind that: save a copy, compare before touching a row, and chase missing information one message per person. Plus the honest chapter on when a diff is more ceremony than control.

Paul Prado Pacardo

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

Contents

Nobody kept the last copy the handover that never happened

1 **Today's list cannot tell you what changed** why the top-to-bottom pass loses the story

2 **Four buckets before any work** the method, in eight steps

3 **One person, one message** chasing missing information without the noise

4 **The queue started telling a story** what changed, and what it is evidence for

5 **Where a diff lies to you** the failure modes, and how to catch them

6 **When not to diff** the cases where this is overkill

Start from what changed closing

The daily diff checklist and the live artifacts from this story

OPENING

Nobody kept the last copy

I stepped into a small back office and was handed several daily queues at once: approvals, exceptions, entries waiting on missing information. No notes came with them. No list of what had already been worked, and no list of who had already been contacted.

> WHAT IT COST, EARLY
>
> People who had already been chased received a "first contact" introduction again. Polite, well written, and wrong: it told them nobody on our side knew the history.

Every morning the queue looked like a fresh to-do list. It was not. Some rows were new, some were left over, some had quietly changed, and some had vanished. The screen showed all of them the same way.

THE FORK IN THE ROAD

THE SAFE MOVE

Work the list top to bottom. Open the queue, start at row one, fix what you can, chase the rest. It feels productive from the first minute and nobody can say you were idle. It also treats every row as if it arrived this morning.

THE MOVE I MADE

Save a copy and diff before touching a row. Keep yesterday's list, compare today's against it, and sort every row into new, carried, updated or resolved. Ten slower minutes at the start, and then every action is aimed.

Figure: the decision that separates working a queue from re-reading it.

This playbook is the story of taking the right-hand branch, the outreach routine built on top of it, and the rules that kept both honest. It ends with the chapter on when a diff is overkill, because a method that claims to always apply is a slogan.

CHAPTER 1

Today's list cannot tell you what changed

A queue screen is a photograph. It shows what is open right now. It does not show what was open yesterday, what was fixed overnight, or what came back after someone thought it was done.

Working top to bottom answers the wrong question. It asks "what is on the list?" The useful question is "what is different from the last time I looked?" Only the second one tells you where the work is.

Without a last copy, every row is a guess. Is this person new, or have they been asked twice already? Did that error clear because I fixed it, or because the system recalculated? Is this the same entry with a corrected date, or a second one? You cannot answer any of these from today's screen alone.

> THE BET
>
> Ten minutes spent comparing two lists saves an hour of re-asking, re-checking and re-explaining. The first copy you save is the one everything later depends on.

The repeat introductions were the visible failure. The quieter one was reporting. When the supervisor asked "how many of these are new since yesterday?", the honest answer at the start was "I do not know." That is not an answer anyone can plan around.

So the first thing I built was not a fix. It was a record of what I had been handed: a saved copy of every queue, every day, taken before I touched anything. On day one that copy is just a list. On day two it becomes a baseline, and the queue starts telling a story.

The rule that came out of it is short: on day one, save a copy of every queue you touch; on day two, compare. Nothing else in this playbook works without that first copy.

CHAPTER 2

Four buckets before any work

The diff takes about ten minutes on a queue of a few dozen rows. The order matters more than the speed.

Step 01 **Save today, untouched**

Copy the list exactly as the system shows it, and note which view or filter it came from.

Step 02 **Open the last copy, and check its date**

Comparing against the wrong day is the most common way to get a confident, wrong answer.

Step 03 **Pick a match key**

Person plus item plus date, using the system ID. Never a display name that can vary.

Step 04 **Sort every row**

NEW (only today), **CARRIED** (same key, same data), **UPDATED** (same key, changed data), **RESOLVED** (only yesterday).

Step 05 **Count and say it out loud**

"Eighteen rows: six new, nine carried, one updated, two resolved." Before working anything.

Step 06 **Attribute every resolved row**

Fixed by us, or cleared on its own. Write which.

Step 07 **Explain old carried rows**

Anything carried more than two days gets one line on why it is still open.

Step 08 **Save tonight's copy as tomorrow's baseline**

With statuses, so tomorrow starts from tonight, not from memory.

> THE RULE
>
> **Never call a row new without checking the last copy.** And "I can't find it anymore" means RESOLVED, never FIXED. A row that disappeared is a fact; a fix is a claim you have to be able to show.

CHAPTER 3

One person, one message

Once the diff says who still owes something, the next job is asking them. When many people each owe one missing piece, the outreach batch decides whether the day ends with answers or with a pile of confused replies.

FILTER

Clear the list first

Remove test records, anything outside the current period, and anything dated today.

GROUP

By person, not by row

Every open date for one person goes into one message, in the order the source list shows them.

CHECK

Against the contacted list

First-contact template for new names, repeat template for everyone else. Update the list the same day.

The rules that kept it clean came from mistakes, one each:

> **One person, one message.** Three rows for one person is one email with three dates, not three emails that each look like the only problem.

> **Nothing dated today.** An item from this morning may still be in progress. Messaging someone about work they have not finished reads as surveillance.

> **Confirmations carry no details.** "The adjustment has been entered and the error has cleared. No further action needed." No dates, nothing to fill in, nothing to get wrong. Details belong in the request, not the reply.

> **Registered names only.** A nickname in the log sent a colleague searching for a person who did not exist in the system, and the item was wrongly marked done. Use the name as the system stores it, nickname in brackets.

SEE IT YOURSELF

Both halves are live and free, each loading a worked example: **me.kabuhayan.app/tool-queue-diff** sorts two pasted lists into the four buckets, and **me.kabuhayan.app/tool-exception-email-builder** drops today's rows, groups by person and picks the template.

CHAPTER 4

The queue started telling a story

Within a week the mornings looked different. The day opened with four counts instead of one long list, and the outreach went out as one message per person with the right template.

NO MORE REPEAT INTRODUCTIONS

The contacted list was checked before every batch. Nobody received a first-contact message twice after it existed.

A NAMED LIST BEFORE THE DEADLINE

Roughly two thirds of a full batch replied and were fixed the same day. The rest went to the supervisor as names, rebuilt from the latest queue.

One decision came out of the volume. Every outreach email copied the supervisor, and a single batch could add dozens of messages to her inbox. I did not change it on my own. I offered two options: stop copying, or stop copying and send one daily summary. **The person whose inbox it is decides.** The risk to watch either way is a reply that reaches only her.

> STATED PLAINLY
>
> A queue you cannot compare is a queue you cannot report on. The diff is what turned "a lot came in" into "six new, two resolved, one came back."

Worth being precise about what this is evidence for. One back office, a handful of queues, a weekly deadline that made every open row matter. The figures are banded and they belong to that setting. What generalises is the habit: keep the last copy, compare before acting, and separate what you fixed from what simply disappeared. That travels to tickets, approvals, invoices and inboxes. The exact reply rate does not.

CHAPTER 5

Where a diff lies to you

A diff feels rigorous, which is exactly why its failures are dangerous. Each of these happened, or nearly did:

FAILURE	HOW IT SHOWS UP	THE CATCH
Wrong baseline	Half the queue looks new on a quiet day	Check the date on the last copy before comparing
Name as the key	One person appears as new and resolved at once	Match on the system ID, never the display name
Filtered view	A sudden wave of "resolved" rows	Write down which view each copy came from
Mixed date formats	Carried rows show as updated	Normalise dates before matching
A date the queue never flagged	An email asks about an item that was fine	Say so under the draft before it goes
Overnight items	Started yesterday, ended today, skipped twice	Judge "today" by the end, and check the edge by hand

The outreach side has its own: introducing yourself twice (check the contacted list), a bounced address (go to the account's alternate contact), and anyone silent after two touches (onto the non-responder list, not a third email).

> THE TRAP TO AVOID
>
> Counting by subtraction. "Yesterday twelve were waiting, five replied, so seven are waiting" drifts within days. Rebuild the waiting list name by name from the latest queue, every time. If a number cannot be traced to a row, report "not recorded" instead.

Check any "probably fixed" item against the latest screen before logging it. Probably is not a status.

CHAPTER 6

When not to diff

A method that claims to always apply is a slogan. There are real cases where working the list top to bottom is the right call.

> **When the queue empties every day.** If everything is cleared or closed by evening, there is nothing to carry and nothing to compare. The diff is ceremony.

> **When the system already tracks history.** A ticket tool with reliable created, updated and closed stamps is doing the diff for you. Use its views instead of maintaining a parallel copy.

> **When the list is a handful of rows.** Five items you can hold in your head do not need four buckets. Start diffing when you stop remembering.

> **When the deadline is the next hour.** In a genuine fire, fix the rows that block the deadline first and save the copy afterwards. Then diff tomorrow.

The outreach rules have limits too. **One message per person** fails when the items belong to different owners on their side; split by who can actually answer. **Nothing dated today** bends when the deadline is today and the item truly blocks it; then ask, and say why.

Even in these cases, half the method survives: **separate "I fixed it" from "it went away."** That distinction costs nothing and protects every report you write.

> THE TEST TO APPLY
>
> Will tomorrow's list contain anything from today's? If yes, save a copy and diff. If every row is gone by evening, work it top to bottom and keep only the log.

CLOSING

Start from what changed

Before any work, one quiet decision: read the queue, or compare it. Everything in this playbook is downstream of choosing to compare, then asking each person once, plainly, about everything they owe.

None of it is clever. A saved copy, a match key, four buckets, a contacted list and a two-line confirmation. The value is in doing it every day, before touching a row, so that every number you report can be traced back to one.

The routine did not stay a private habit. Both halves are now free tools on this site, loaded with worked examples, which is its own lesson: a method written down for one handover keeps working for the next one.

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 daily diff checklist

Run this every morning on any queue that carries rows from one day to the next. If the first box will not tick, chapter 6 applies.

BEFORE TOUCHING A ROW

- [ ] Today's list saved exactly as shown, with the view noted
- [ ] Last copy opened, and its date checked
- [ ] Match key is the system ID plus item date, not a name
- [ ] Four buckets counted and stated: new, carried, updated, resolved

BEFORE SENDING OUTREACH

- [ ] Tests, out-of-period rows and anything dated today removed
- [ ] Grouped by person: one message, every date in it, source order kept
- [ ] Each name checked against the contacted list; template chosen per person
- [ ] Registered names used throughout, nickname in brackets

BEFORE THE DAY ENDS

- [ ] Every resolved row marked fixed by us or cleared on its own
- [ ] Contacted list updated; confirmations sent with no details
- [ ] Non-responders rebuilt name by name from the latest queue
- [ ] Tonight's copy saved with statuses as tomorrow's baseline

THE LIVE ARTIFACT

Both tools from this story are free and load a worked example: **me.kabuhayan.app/tool-queue-diff** and **me.kabuhayan.app/tool-exception-email-builder**. More are on the tools page at me.kabuhayan.app/tools.