



A FIELD GUIDE FOR INDEPENDENT DISTRIBUTORS

The Margin Leak Field Guide

Five places money hides in a distributor's ERP — how to find each one, how to put a dollar figure on it, and what fixing it actually looks like. Written for owners and GMs of independent houses in plumbing/PVF, HVAC/R, electrical, industrial MRO, and building materials.

Each check starts with records your ERP may already collect. Data availability and preparation depend on your system. Review any estimate in your own business context.

The first step is a useful baseline.

Pricing, purchasing, stock, and quoting decisions leave traces in your ERP. Reviewing those records can show which workflow deserves attention.

This guide contains five practical checks. Each explains what to look for, why the pattern can arise, how to estimate its scale, and which records to review first.

An estimate identifies an opportunity for investigation. It is not a promise of savings: data quality, business rules, customer commitments, and the cost of changing a process all matter.

How to use this guide

Read the five checks, choose one relevant to your operation, and collect the records for its first review. Assign a reviewer and agree what counts as a useful result.

What each check contains

- What it looks like
- Why it happens
- How to size it
- What good looks like
- A 30-minute check

Stale pricing & underpriced SKUs

The price file ages faster than anyone updates it.

What it looks like

Certain SKUs sell briskly and never seem to make money. A handful of accounts get "their" pricing that predates the current owner. Cost increases from vendors land monthly, but sell prices move annually — if that. Margin by SKU, when anyone finally looks, is a sawtooth: fine, fine, fine, inexplicably 4%.

Why it happens

Distributor pricing lives in a matrix — customer group × product group × override — and matrices decay. A cost update arrives and reprices the default column, but the hundreds of per-account overrides layered on top don't move. Nobody's job is to re-walk the exceptions. After a few years of vendor increases, a meaningful slice of the file is priced against costs that no longer exist.

Illustrative formula

review opportunity \$ / yr =
 Σ below -floor lines:
 (floor GM% – actual GM%) × line revenue

floor GM% = cohort median GM%
 (product group + customer size, last 12 months)

Illustrative estimate; validate assumptions with your own data.

How to size it

Don't argue about what price "should" be. Use your own house as the benchmark: for each product group, compute the median realized margin across comparable customers. Lines running materially below that median — say, more than five points — are your candidates. Multiply the gap by the revenue on those lines and you have a defensible annual figure. Treat the result as a review estimate, not recoverable revenue. Validate the comparison and any exceptions with your pricing team.

What good looks like

Not a repricing project — a habit. A weekly view that ranks the ten most underpriced account/SKU combinations by dollars left behind, so the pricing conversation is specific ("Acme, copper fittings, 11 points under cohort") instead of general ("we should raise prices"). Fix the top ten, next week there's a new top ten, and the file stops aging.

The 30-minute check

1. Export last month's invoice lines: SKU, customer, revenue, cost, margin%.
2. Sort ascending by margin%. Look at the bottom 100 lines.
3. Count how many are below your worst acceptable margin — and note how many of those are your best customers. That pattern is the leak.

Discount creep & counter overrides

Yesterday's favor becomes today's price.

What it looks like

Realized margins drift down a fraction of a point each year with no single decision anyone can point to. Counter staff override price to "match" a competitor quote that nobody verified. A discount given to win one job in 2023 is still being applied to every order in 2026, because it was keyed as a standing override instead of a one-time adjustment.

Why it happens

Overrides are individually rational. The counter wants to keep the contractor moving; the outside rep wants the order; the branch manager approves because the customer is "important." The system faithfully records every one — and then nothing reviews them. Overrides don't expire, don't escalate, and don't aggregate anywhere a manager naturally looks. So creep compounds quietly, order by order.

Illustrative formula

$$\begin{aligned} \text{override exposure / yr} &= \\ &\text{override line revenue} \times \\ &(\text{standard GM\%} - \text{overridden GM\%}) \\ \\ \text{drift} &= \text{avg realized discount this year} \\ &- \text{avg realized discount two years ago} \end{aligned}$$

Illustrative estimate; validate assumptions with your own data.

How to size it

Two numbers. First, the override rate: what share of invoice lines shipped at a manually keyed price rather than the matrix price, and at what margin cost. Second, the drift: for your top 50 accounts, compare average realized discount today against two years ago. Review the rate and the reasons for each override with the branch team before deciding which changes are warranted.

What good looks like

Overrides still happen — flexibility is part of being an independent. But they expire by default, they're visible weekly to the branch manager grouped by rep and account, and the handful that cost real money get a conversation. The goal is not zero overrides; it's zero permanent accidental overrides.

The 30-minute check

1. Pull one week of invoice lines with a price-override flag (confirm which flag your ERP provides).
2. Compute override rate: overridden lines ÷ total lines.
3. Take your single biggest account and compare its average realized discount this year vs. two years ago. If it drifted more than a point with no contract change, you've found creep.

Dead & slow-moving stock

Inventory that stopped being an asset and started being rent.

What it looks like

Aisles you don't walk. Bins whose labels have faded. A warehouse that's "full" while fill rates on the items customers actually order are slipping. Working capital tied up in product bought for a job that finished, a line you no longer push, or a vendor minimum that made sense at the time.

Why it happens

Dead stock is created upstream — by over-buying, job fallthrough, line changes, and safety-stock formulas nobody revisits — but it persists because carrying it is silent. There's no invoice for the cost of keeping it. Carrying cost includes capital, space, insurance, shrink, and obsolescence. Estimate those costs using your own finance assumptions.

Illustrative formula

dead stock \$ = inventory value with
0 sales in trailing 12 months

annual bleed = dead stock \$ × your carrying-cost
rate

recovery = liquidation value +
redeployed capital – write-down

Illustrative estimate; validate assumptions with your own data.

How to size it

Segment inventory by months since last sale: 0–6, 6–12, 12–24, 24+. Value each bucket. The 12-month-plus buckets are your dead stock number; apply your own carrying-cost rate to estimate the annual cost. Then do the harder, more useful thing: for the top 50 dead items by value, answer how each got there. The pattern — one buyer, one vendor, one branch, job leftovers — tells you which upstream valve to close so the pile stops refilling.

What good looks like

A standing dead-stock review with teeth: every item over 12 months gets a disposition — transfer to the branch that sells it, return under vendor terms, kit it, discount it to contractors who'll take it, or scrap it and take the write-down honestly. And a purchasing guardrail (see Leak 04) that warns before anyone buys more of something the company already can't sell.

The 30-minute check

1. Run an inventory valuation report grouped by months-since-last-sale.
2. Sum the value of everything over 12 months. Apply your own carrying-cost assumption.
3. That's the annual cost of deciding nothing. Compare it to your branch's profit target and decide if it's worth an afternoon a month.

Purchasing misfires

Buying tomorrow's dead stock at yesterday's terms.

What it looks like

POs cut for items the company already holds a year's supply of. Vendor bracket and rebate thresholds missed by a few cartons — or blown past for product that won't move. Expedited freight on stockouts of A-items while C-items pile up. Buyers working from min/max settings that were tuned once, years ago, under a different demand pattern.

Why it happens

Purchasing at most independents is a high-volume, low-information job. The buyer sees the suggested-PO screen, not the slow-moving report; the rebate tracker lives in a different module (or a different spreadsheet); seasonality lives in someone's head. The ERP has every input needed to buy well — demand history, on-hand, open orders, vendor terms — but rarely puts them on one screen at the moment of decision.

Illustrative formula

```
misfire $ / yr =
  PO $ for items > 12 months supply
  + missed rebate/bracket $
  + expedited freight premium $
  + stockout GP lost on A-items
```

Illustrative estimate; validate assumptions with your own data.

How to size it

Audit last quarter's POs against on-hand at time of order: how many dollars were committed to items that already had more than a year of supply? Separately, list vendor programs with volume thresholds and check how many were missed by less than 10% — those are usually recoverable with nothing more than timing. Add the freight ledger's expedite premiums. The total is your purchasing leak, and it's usually the most fixable number in this guide because it's forward-looking: every future PO is a chance to stop it.

What good looks like

The buyer's screen answers three questions before the PO is cut: Do we already own too much of this anywhere in the company? Does this order help or hurt a rebate threshold? Is this the right time in the demand curve? Same buyer, same vendors — better information at the moment it matters.

The 30-minute check

1. Pull last month's POs. Join against months-of-supply at order date.
2. Flag every line where supply exceeded 12 months. Sum the dollars.
3. Ask your best buyer what information they wish they had on screen when they cut a PO. Write the answer down — that's your spec.

Slow quotes

The order often goes to whoever answers first.

What it looks like

A contractor emails a materials list at 7am. The quote goes out after lunch — or tomorrow, if pricing needs a manager's look or an item needs sourcing. By then the competitor across town, quoting from the same vendor price sheets, has the PO. Inside sales spends its day re-keying line items instead of selling.

Why it happens

Quoting touches everything — pricing, availability, sourcing, margin approval — so it inherits every slow spot in the operation. Each quote is a small research project: look up the part, check three branches, guess at the price the matrix intends, get sign-off if it's big.

Illustrative formula

$$\text{slow-quote \$ / yr} = \text{quotes/year} \times \text{slow share} \times (\text{win\% fast} - \text{win\% slow}) \times \text{avg GP/quote}$$

Illustrative estimate; validate assumptions with your own data.

How to size it

Your quote header file has created-at and won/lost. Bucket quotes by turnaround (<1h, same-day, next-day, later) and compare win rates across buckets. The gap between your fast bucket and slow bucket, times the gross profit on an average quote, times the number of quotes stuck in the slow buckets, is the leak.

What good looks like

A useful draft brings matched items, branch availability, pricing references, and exceptions together for a human reviewer. Agree the input formats, review steps, and acceptance criteria before delivery. Measure turnaround against your own baseline.

The 30-minute check

1. Pull 90 days of quotes with timestamps and outcomes.
2. Compute win rate for same-hour vs. next-day quotes.
3. Ask inside sales to time one representative quote end to end, and note where the minutes actually went. That's your bottleneck map.

The ten-question leak scorecard.

Score each question: 2 = yes, systematically; 1 = sometimes; 0 = no. Twenty points possible. Use the score to prioritize review, not to estimate savings.

Score on paper: 2 = yes, systematically; 1 = sometimes; 0 = no. This is not a fillable PDF form.

Question	Score
Can you list your ten most underpriced SKU/account combinations, ranked by dollars?	___
Do per-account price overrides expire automatically unless renewed?	___
Does someone review counter overrides, grouped by rep and account, each week?	___
When vendor costs change, are account overrides re-checked as well as default prices?	___
Do you know the value of stock with no sales in the past 12 months?	___
Does each item beyond your aging threshold have a disposition decision on record?	___
Can buyers see company-wide months of supply before committing a purchase order?	___
Are vendor rebate and bracket thresholds tracked during the period?	___
Do you know your median quote turnaround time and its trend?	___
Do you compare win rates for fast and slow quotes?	___

Total score: ___ / 20

- 16-20 Keep the checks in your regular review cycle.
- 10-15 Choose the least consistent workflow and run its check first.
- 5-9 Start with pricing and stock visibility to establish a baseline.
- 0-4 Pick one workflow, assign an owner, and collect the data for its first check.

A solution built around your business.

dstl consulting helps distributors address purchasing, pricing, inventory, and quoting challenges with solutions shaped around their processes, people, and existing systems.

Start with one workflow worth improving. Together, we define the data, decisions, and requirements a useful solution needs to address.

01 Understand the problem

Discuss the decisions that need attention, the people involved, and the information available.

02 Agree the requirements

Define the solution, ERP integration, deployment needs, responsibilities, and acceptance criteria. Scope, fees, and timing are agreed for your engagement.

03 Deliver against the agreed scope

Build and validate the solution against the requirements and outcomes agreed with your team.

Your environment. Your data. Clear terms.

Designed for on-premises deployment. You own your data and control deployment and access. Application software is licensed from dstl consulting; software and source-code ownership are not transferred.

Discuss your business needs: info@dstlconsulting.com