

Fewer clicks, more control

One supplier invoice, several orders

Emma: Welcome to Central Perspectives, a Business Central podcast from ITrators. I'm Emma, here with Alex. We're AI hosts, and our scripts and voices are AI-generated. Today we're looking at work that becomes easier when the system follows the real business process: handling a supplier's combined invoice, reviewing journal batches, and preparing recurring bills. We'll also separate a reporting integration change from the reports people actually use, and look at what hiding a field on screen does, and doesn't, protect. Some of these are useful features from earlier this year, rather than October announcements. Now, let's get started.

Emma: Let's start with a very ordinary piece of paperwork. Imagine the company has raised three purchase orders with one supplier, and the supplier sends a single invoice covering all three. The accounts person now has one external document and several internal orders to connect. I would rather not ask the supplier to rearrange their invoice just to suit our screen. Where would you start in standard Business Central, Alex?

Alex: I'd start on the purchase invoice, rather than trying to force one invoice onto one order. There are two useful routes. If the goods have already been received, Get Receipt Lines lets you bring received but uninvoiced lines together, for the same vendor and currency. Get Order Lines lets you associate invoice lines with eligible order or receipt lines, including more than one source line. So your accounts person can preserve the connection to the purchases instead of typing a fresh, disconnected description just because the supplier combined the paperwork.

[Combine receipts or purchase order lines on a single invoice](#)

Emma: Get Order Lines dates from April, rather than being an October launch. And 'more than one source line' matters here. Suppose the supplier combines one item we ordered on Monday and again on Thursday. Can the reviewer inspect those two relationships, rather than just trusting the total?

[Combine receipts or purchase order lines on a single invoice](#)

[Match purchase invoices to multiple order and receipt lines](#)

Alex: Yes. Matched Order Lines is where you can inspect and adjust the relationships, from an individual invoice line or from the invoice header. Your Monday-and-Thursday example is a good reason to look there. A plausible total isn't enough to identify which purchases the invoice settles. I'd want the reviewer to follow the line back to the intended orders and quantities. The useful improvement is that the invoice can reflect the supplier's presentation without throwing away the company's purchasing detail.

[Combine receipts or purchase order lines on a single invoice](#)

Emma: That feels like a sensible place to save effort: fewer reconstructed lines, but still something meaningful to check. Now let me complicate the example. The boxes have arrived, but nobody has posted the receipt yet. Accounts has the invoice, the buyer says the delivery is fine, and the warehouse is busy. I can imagine someone thinking, 'We're nearly there, let's just finish the invoice.' Is that the moment when matching and receiving start to get mixed up?

Alex: It is, and there is a real option here, not just a warning. Get Order Lines can match an order that has no receipt. Separately, Receipt on Invoice can create the receipt for the matched quantity before invoicing. But those are two different things: connecting the invoice to the order, and recording that quantity as received. For your example, I'd ask whether accounts is supposed to take that receiving step on the warehouse's behalf. The feature can support that decision; it shouldn't make the decision accidentally.

[Combine receipts or purchase order lines on a single invoice](#)

Emma: I'd be comfortable with that if the same person receives the supplies and handles the invoice. Less comfortable if accounts has only an email saying 'looks fine.' Does the receiving option work with every warehouse setup?

Alex: No. Receipt on Invoice isn't available with directed put-away and pick, or item tracking, or where an existing posted receipt is involved. Those boundaries matter before anyone designs a new routine around the checkbox. Your same-person receiving case may be a good candidate, but a tracked item or a different location setup can change the answer. And a company can still benefit from matching invoice lines even where automatic receiving isn't available. We shouldn't dismiss the useful matching part because the receiving shortcut doesn't fit.

[Combine receipts or purchase order lines on a single invoice](#)

Emma: That's worth keeping separate. There is also a vendor policy called Receipt on Invoice, with Manual and Automatic choices. The word Automatic makes this sound like a switch someone might turn on to clear a backlog. But Microsoft's guidance says it applies to new eligible orders, not existing ones. If the accounts team changes it today, yesterday's unfinished orders haven't silently acquired a new receiving policy. What would you make explicit before using that default?

[Combine receipts or purchase order lines on a single invoice](#)

Alex: I'd agree who confirms receipt, and then check a newly created order rather than using the backlog as the test. Manual leaves the option off; Automatic enables it on eligible new headers and lines. That still isn't evidence the goods arrived. I like your distinction between a default and clearing a backlog, because otherwise the first person trying it may judge the feature from an old order that was never changed. A small, deliberately chosen example will tell you more than repeatedly toggling the vendor setting.

[Combine receipts or purchase order lines on a single invoice](#)

Emma: And before somebody tries their most complicated purchase first: Get Order Lines has exclusions. The guidance lists prepayments and item charges, and orders linked to projects, subcontracting, blanket orders or intercompany. That's a fairly specific boundary, not 'every purchase document now combines.' For a team already using Get Receipt Lines successfully, would you actually tell them to change their process? Or is this one of those cases where knowing another route exists is enough?

[Combine receipts or purchase order lines on a single invoice](#)

Alex: Knowing is enough if their current route fits. If they reliably receive first and then combine those receipts, I wouldn't create a change project just to use a newer action. I'd investigate Get Order Lines where the supplier's combined line is awkward to relate to multiple orders. Either way, check the invoice before posting. A posted combined-receipt invoice needs a credit memo to correct it, and fully received and invoiced source orders aren't automatically deleted. Combining the invoice doesn't remove the need for sensible follow-through on the purchase documents.

[Combine receipts or purchase order lines on a single invoice](#)

Approve the batch before execution

Emma: Let's move from checking an invoice to checking a proposed change to stock. Another April capability extends approval to item journal batches and planning or requisition worksheets. What caught my eye is that the pending decision protects the batch from changes. Imagine a stock count where somebody sends the adjustment batch for approval, then discovers a counting mistake. I can see the attraction of freezing what the reviewer sees. I can also see the person with the correction getting frustrated.

[Approve requisition worksheets and item journals](#)

Alex: That frustration is real, but the freeze is doing a useful job. The reviewer should be deciding on the batch that was submitted, not a moving target. While journal approval is pending, editing, deleting and posting are blocked until approval completes or is canceled. For worksheets, pending approval blocks inserting, modifying or deleting lines. So if that count correction changes what needs approval, I'd bring it back into the preparation process, correct it, and submit the intended batch. Not find a side door around the review.

[Approve requisition worksheets and item journals](#)

Emma: Then the uncomfortable question is when the team submits it. If we're still counting the last shelves, sending the batch early might create more interruption than control. I'd want the person preparing it to know when it is genuinely ready. But I don't want 'ready' to mean they've hidden every unusual result. A large adjustment may be exactly what the reviewer needs to see. What should that reviewer be deciding, rather than just clicking a green button?

Alex: For that count, I'd give the reviewer enough context to distinguish a credible correction from a quantity entered against the wrong item or location. That's our review design, not something the approval status proves for us. They approve the batch, not a separate ballot for every line. A large difference with a clear explanation may be perfectly reasonable. A tiny difference against the wrong item still isn't right. Your point about not tidying away unusual results is important: review needs the explanation, not a more comforting number.

[Approve requisition worksheets and item journals](#)

Emma: That also changes how I'd group the work. If one batch contains five people's adjustments from completely different areas, one approver may not be well placed to judge all of it. But splitting everything into tiny batches could mean the reviewer spends the afternoon opening requests. There has to be a middle ground between one enormous approval and an approval notification for every loose screw.

[Approve requisition worksheets and item journals](#)

Alex: I'd group work around a decision somebody can reasonably own. That might be a coherent count area or a particular planning responsibility, depending on the company. The configuration can use journal template and batch conditions, so the batch structure is relevant to the approval route. But more batches aren't automatically better control. If the same person needs the combined picture to spot a problem, splitting it may make their judgment harder. Start with who can explain and review the work, then choose the structure.

[Approve requisition worksheets and item journals](#)

[Use approval workflows](#)

Emma: So this is something a stock or planning manager should help shape, not just a workflow setting handed to them afterwards. The release lists Item, Physical Inventory, Output and Consumption journals, as well as the planning and requisition worksheets. I wouldn't read that as a promise about every journal anywhere in Business Central. With a planning worksheet, what is the person actually reviewing? It isn't a stock count this time.

[Approve requisition worksheets and item journals](#)

Alex: They are reviewing a proposed plan before it turns into the next operational step. In a hypothetical purchasing team, the questions might be whether the suggested quantities and dates still fit known demand, or whether an unusual proposal needs explanation. The freeze matters because otherwise the plan could change while the approver is considering it. I wouldn't ask the reviewer to repeat every calculation manually. I'd agree which exceptions need judgment and make those visible alongside the batch, so approval adds a decision rather than a second round of data entry.

[Approve requisition worksheets and item journals](#)

Emma: Now the approver is away and the batch is locked. We can call that governance while the company calls it Tuesday's problem. Is there a normal way to move the decision to someone available?

Alex: The approval actions include delegation, as well as approve, reject and comments. So availability should be part of setting this up, not an excuse to remove the control later. The user and notification setup matter, and Approval Entries gives people a place to see the request's status. I'd rehearse the absent Approver case before relying on the workflow for time-sensitive work. If there are multiple required approvers in a step, one response doesn't stand in for all of them. The route must fit the decision you actually need.

[Approve requisition worksheets and item journals](#)

[Use approval workflows](#)

Emma: And when approval arrives, does the batch post automatically? I'd want the manager to know whether that click releases the work or actually carries it out.

[Use approval workflows](#)

Alex: Not necessarily. Automatic posting can be a configured workflow step; it isn't the inherent meaning of every approval. I'd show the team the actual route, including what remains manual. That gives us a useful acceptance check: submit a representative batch, confirm the expected restrictions while it is pending, and follow the decision through to the agreed next action. Include a correction and a delegation. If those everyday cases aren't workable, adding another notification won't fix the process. Configure the useful control, not the largest workflow the template permits.

[Approve requisition worksheets and item journals](#)

[Use approval workflows](#)

Choose how far billing should run

Emma: That distinction leads rather nicely into recurring billing. Microsoft's April release added more automation around customer contract invoicing in Subscription Billing. The appealing promise is less monthly preparation, but people can hear 'automation' and imagine invoices going out without anyone looking. The documentation actually describes different levels, from preparing proposals to creating documents for review and, for predictable contracts, posting. Which level would you choose for a company that's tired of rebuilding the same month's billing work?

[Billing automation overview](#)

[Reduce time spent on recurring invoicing in Subscription Billing](#)

Alex: I'd first remove the repetitive preparation without necessarily removing the review. If the difficult part is gathering what's due, automated proposals may already help. If the proposals are reliable but somebody spends time creating each document, the next useful step may be draft invoices ready for checking. Fully automatic posting is a further choice, not the entrance fee. And this is customer contract billing: the documented automation doesn't automate vendor contract invoicing. That's worth saying early, before somebody plans both sides of the business around it.

[Billing automation overview](#)

[Recurring billing](#)

Emma: I'd choose the ready-to-review invoices in our hypothetical company. Let's say it has a set of straightforward monthly service contracts, but a few customers have changes that need attention. I don't want the uncertain few to dictate a manual routine for everybody. At the same time, I don't want the person reviewing to lose sight of which contracts were included. How does the process decide what belongs in this run?

Alex: The recurring billing proposal selects eligible contract lines using their Next Billing Date. Automation templates define the contract types and how the documents are grouped, such as by contract, partner or recipient. So I'd give your predictable service contracts a deliberately bounded template and review the proposal it produces. The grouping deserves a business decision as well: does the customer expect one combined document, or separate documents they can route to different people? Fewer invoices for us isn't automatically easier processing for the customer.

[Billing automation overview](#)

[Recurring billing](#)

Emma: You've just taken away my beautifully small invoice count. Fairly, I think. If the customer's two departments need different explanations, combining them might simply send the sorting work across the table. I'd rather judge the grouping by whether the recipient understands the bill. Can the person reviewing get back to the calculation behind a billing line, especially when the total covers more than one contract?

Alex: Yes. Billing Lines exposes the calculation detail, and after posting there are Archived Billing Lines. A posted entry covering multiple contracts is identified as multiple contracts rather than pretending there's one contract behind it. I'd use that detail in the review, particularly for the cases your customer might query. It also changes how we judge the combined-document option: can somebody explain the charge without reconstructing it from memory? Keeping the calculation traceable can make a combined invoice useful rather than merely shorter.

[Posting documents](#)

Emma: Then let's use one of the awkward cases. The proposal is ready in the morning, but someone changes a contract before the invoice is created. Perhaps a service quantity was corrected after a customer call. I wouldn't want yesterday's understanding of the contract to quietly become today's invoice. Does the proposal notice that its source has changed, or is the reviewer expected to remember every conversation?

Alex: Changed source lines can be marked Update Required, and document creation is blocked until the proposal is updated. That is a useful interruption: the system isn't asking the reviewer to infer a fresh calculation from stale lines. For your quantity correction, I'd update the proposal and inspect the changed calculation before creating the document. The important point is that the proposal is a working stage, not an immutable invoice. Preparing it early can save time, but it doesn't freeze the business facts forever.

[Recurring billing](#)

Emma: Move that customer call one step later: the draft invoice exists, but isn't posted. Can I still correct the contract and refresh, or has creating the document changed the route?

Alex: Creating the unposted billing document changes the situation. It blocks the relevant source-line changes. For a billing-relevant correction, the documented route is to delete the unposted document, correct the source and recreate the document. So your instinct to correct the underlying contract is sound, but the order matters. I would make that distinction part of the team's routine: proposal needing an update, versus a document that must be removed and recreated. Otherwise people can spend time looking for a refresh that isn't the appropriate correction at that stage.

[Recurring billing](#)

Emma: And I notice another easy misunderstanding in the documentation. Creating the proposal advances Next Billing Date. So if someone sees the date move forward, that doesn't by itself tell them the customer has a posted invoice. I'd want our morning billing check to look for the result we intended, not use that date as a reassuring shortcut. What protects the same contract period from being proposed repeatedly while we're sorting this out?

[Recurring billing](#)

Alex: A contract line can't be proposed twice for the same period. Deleting a proposal or changing its period adjusts that next date appropriately. Create Documents works on billing lines that don't already have documents. Those controls are useful, but I agree with your proposed check: distinguish the proposal, the created document and the posted result. If a colleague asks whether this month's billing is finished, tell them which stage is complete. A scheduling marker is a poor substitute for the invoice they actually need.

[Recurring billing](#)

Emma: Let's make 'this month's billing' less tidy for a moment. Suppose one contract missed the previous run, and the team wants to catch up now. I can imagine the reviewer expecting one line because they're doing one billing run. But that run might need to cover more than one period. Which date tells Business Central what to include, and which tells it how far the bill should reach?

Alex: Those are distinct settings. Billing Date is used to select eligible contract lines by Next Billing Date. Billing to Date defines the end of the period to be billed, and it can shorten or extend that period. If the billing rhythm fits more than once between the next billing date and that end date, the proposal can contain multiple lines. So in your catch-up case, one run doesn't necessarily mean one period or one proposal line.

[Recurring billing](#)

Emma: Then two lines for that contract might be exactly what we wanted, not a duplicate to remove. I'd want to see the periods before somebody cleans up the proposal. In our catch-up example, would you judge it against the customer's agreement rather than just comparing it with the number of lines on last month's invoice?

Alex: I would. I'd establish the period we intend to charge and inspect the billing detail against that. Last month's line count isn't a reliable definition of this month's obligation, especially when we're recovering a missed period. It also gives the reviewer something concrete to explain if the amount differs from the familiar monthly charge. We aren't promising a particular rounding or proration result here; the useful step is to inspect the calculation for the actual contract and intended period.

[Recurring billing](#)

[Posting documents](#)

Emma: There is a smaller discovery here for a shared billing team. The My Suggestions Only view can hide a proposal another colleague has already created. So a contract that appears to be missing from my view may not be missing from the process. I'd be tempted to run the proposal again. Would that actually help?

[Recurring billing](#)

Alex: Not if the same contract line and period already have a proposal. That uniqueness rule still applies, even when the view hides the colleague's suggestion. I'd check the filter and the existing proposal before changing dates or trying to recreate anything. Then agree who is taking it forward. Otherwise two people can each believe the other has missed the contract, when the immediate problem is visibility and ownership rather than billing eligibility.

[Recurring billing](#)

Emma: I'd put that in the handover between colleagues: what's already proposed, for which period, and who finishes it. Much more useful than 'I couldn't find it, so I tried again.'

Alex: And it keeps a helpful personal filter from becoming a misleading picture of the whole team's work. I'd keep the filter if it makes the daily queue manageable, but make the wider check part of investigating a missing contract. We don't have to remove a convenience to understand its boundary.

[Recurring billing](#)

Emma: Let's say the company becomes comfortable with its predictable contracts and wants posting automated too. I can accept that as a second decision. What would make you nervous isn't the automation itself, I suspect, but an overly broad posting job. There may be ordinary sales invoices waiting for entirely different reasons in the same company. How do we keep this billing routine from helping itself to those as well?

Alex: With the specific filters on the posting job. The documented batch posting setup uses both Recurring Billing and Auto Contract Billing, so it targets the intended documents rather than unrelated invoices. Preparation and posting are separate jobs: the automation codeunit handles proposals and documents, and a separate batch-posting report can post them. I'd have the administrator show exactly what that posting job can select. The useful question isn't 'does it run every night?' It's 'which documents can it turn into posted invoices when it runs?'

[Billing automation overview](#)

Emma: Then an unrelated invoice left waiting would be a useful check, not just a nuisance in the test company. We should be able to see it remain untouched. I'd also put one deliberately problematic contract alongside the good ones. If everything is perfect in the demonstration, we haven't learned what the accounts person wakes up to when something fails. Does one bad line stop all billing, or can the rest continue?

Alex: Processing can skip problematic entries, and failures go into the Contract Billing Error Log. But a failing line means the whole affected document is discarded. That makes the grouping decision we discussed more interesting: several contracts combined into one document also share that document's failure. The rest of the run may progress, while that combined invoice doesn't get created successfully.

[Billing automation overview](#)

Emma: So the customer-friendly grouping might still be right, but now the person handling exceptions needs to recover the whole missing document. Who owns that morning check?

Alex: I'd assign it to somebody who can understand the billing result, with an administrator available for job failures. The review should compare the expected billing with the documents actually created or posted, then use the error log to investigate what's missing. If you only check whether a job ran, you can miss the contract that was skipped. And the recovery should include confirming the corrected document appears, not merely clearing an error message. That's a smaller daily job than preparing every invoice manually, but it is still a real responsibility.

[Billing automation overview](#)

[Posting documents](#)

Emma: That gives the accounts person a better job than watching a progress bar. One practical setup detail before we leave this: users need permission to edit the automation templates and Auto Contract Billing in User Setup. So we shouldn't tell someone to find a button and assume they can switch the routine on. Would you still start with draft documents for our example company?

[Billing automation overview](#)

Alex: I would. We've found useful things to learn about grouping, contract corrections and exceptions before posting becomes automatic. Draft documents remove effort while keeping those decisions visible. Once the predictable group really is predictable, the company can consider the posting step with the right filters and an exception owner. It may also decide the draft stage is enough. That's not an incomplete automation project; it can be the right stopping point for the work.

[Billing automation overview](#)

[Recurring billing](#)

Find the reporting dependency

Emma: Now for a change that is genuinely about the coming release calendar. Microsoft has deprecated the beta Finance reports API in the current wave, with removal in the first wave of 2027, version thirty. The headline needs that full name. It does not mean financial reports are disappearing from Business Central. Alex, who actually needs to pay attention?

[Deprecated features in the application](#)

Alex: The people responsible for integrations that consume that particular API family. A company may have an external report or service using it without the finance team knowing the endpoint's name. But we shouldn't assume every company with a dashboard is affected. The first useful step is to identify the actual connection behind the output, and whether it uses this beta Finance reports interface. Microsoft's notice also appears on the beta subscriptions resource. That is a concrete clue for a developer, not a reason to rebuild every report with the word finance in its title.

[Deprecated features in the application](#)

[subscriptions resource type \(Beta\)](#)

Emma: I'd start with the report somebody needs for a real meeting, then find its connection owner. Otherwise a broad message saying 'finance API removal' could send everybody investigating different things. Suppose we do find an affected connection. Is there a recommended destination, or are we choosing a completely new reporting architecture?

Alex: Microsoft recommends the Analytics API for reporting and analysis. That gives us a direction, but not a promise that an old request can simply have its address changed. The Analytics API exposes read-only datasets. We'd need to identify the entities the consumer requires, compare the fields and understand how the existing report uses them. Starting from your real meeting report helps: it gives the migration a concrete result to preserve, instead of treating a successful connection as the whole job.

[Analytics API overview](#)

Emma: What would you compare first? A grand total can match even when the departments underneath it are wrong.

Alex: I'd compare a bounded set of familiar results at the level people use to make decisions. For your department example, that means the same company, period and department breakdown, with an agreed treatment of adjustments. Then investigate differences rather than masking them with a matching grand total. I would also choose a period where the team understands an unusual transaction, because it can expose a mapping assumption ordinary totals hide. That's a proposed migration check, not evidence we've run this migration or that the two interfaces return equivalent structures.

Emma: And then I'd want it to refresh without the developer sitting beside it. A successful demonstration is useful, but our hypothetical report has to be ready for the meeting every week. What changes when we move from one successful request to a routine the company relies on?

Alex: Authentication, permissions and operational ownership need to work in the intended routine, not only in somebody's interactive session. The Analytics API uses OAuth, with an Entra token and permission to the company data. I'd verify the actual consuming identity and schedule, then make failure visible to somebody who can act. Finance also needs to know when the report last refreshed successfully. None of that requires replacing the report just for appearance. It means preserving its usefulness once the underlying connection changes.

[Analytics API overview](#)

Emma: One boundary I wouldn't leave buried in a technical note: the Analytics API is read-only. It doesn't create, update or delete transactions. So if someone discovers that an integration does more than supply a report, the recommendation doesn't automatically cover that other work. We should inspect what the consumer actually does before calling it a reporting migration.

[Analytics API overview](#)

Alex: Exactly that boundary. The Analytics namespace is microsoft slash analytics slash v one point zero; we'll put the precise reference in the notes for the person investigating. Standard Power BI apps use these APIs, but that doesn't establish a drop-in replacement for every older consumer. For now, the useful action is a targeted inventory and a migration owner ahead of version thirty. If the company has no consumer of the retiring Finance reports API, this specific notice doesn't create work for it. We can let an unaffected system remain unaffected.

[Deprecated features in the application](#)

[Analytics API overview](#)

Hide a value on screen

Emma: Our last discovery is much more visible, or deliberately less visible. Business Central has UI support for masking a field, with the feature enabled by default from version twenty-nine. Imagine sharing your screen to explain a customer card, while a value you don't need to show sits beside the useful information. I like the idea of keeping it concealed until needed. But I suspect 'enabled by default' isn't the same as every sensitive field suddenly hiding itself.

[Introduce UI support for masking sensitive data feature key](#)

Alex: You're right. A developer still has to use the MaskType property on the relevant field. The feature being enabled makes the UI capability available; it doesn't decide which information your company considers sensitive. On a supported field, the hide-and-show control lets the user reveal the value when needed. Your shared-screen example is a good use for that: reduce casual exposure while the person is working, without making the value impossible for that user to access.

[MaskType property](#)

[Introduce UI support for masking sensitive data feature key](#)

Emma: So it's a courtesy to the people accidentally seeing my screen, not a new permission model for the person using it?

Alex: That's the useful distinction. Microsoft is explicit that the unmasked data still reaches the browser. This protects presentation, not access to the underlying value. It isn't encryption, an export restriction or a compliance guarantee. If the requirement is that a person must not have access, masking doesn't meet it. But if they legitimately need access and simply don't want the value visible during a demonstration or at a shared desk, reducing that accidental exposure can still be worthwhile.

[MaskType property](#)

Emma: I wouldn't dismiss that just because it isn't a security wall. There are plenty of moments when the goal is simply not displaying something unnecessarily. But I would ask the person requesting it to show the actual page. A concealed field on a card won't help if their usual presentation happens somewhere else. Does the same property work in list rows?

Alex: Not in a repeater, which is the important limitation for that list-row idea. It also isn't supported in a configuration dialog. The property supports several field types, including text and numbers, but the developer needs to check the actual control and existing field setup. I'd start with your real presentation page, choose the intended field, and verify its concealed and revealed states there. Then check the other places people use that information. That is a small, focused improvement, not a promise that every appearance of the value has been hidden.

[MaskType property](#)

[Introduce UI support for masking sensitive data feature key](#)

Emma: That's a useful size for the idea. Show the page, explain who might see it, and decide whether hiding that field solves the actual problem. We don't need to turn a screen-sharing improvement into a promise about everything the company does with its data. And we don't need to reject the improvement because it was never meant to do all of that.

Emma: That's it for today. We found room to simplify invoice matching and recurring billing without pretending that every next step should happen automatically. We also looked at keeping a reviewable batch stable, identifying the particular reporting connections that need attention, and making shared screens a little more discreet. The useful question throughout was what the person doing the work actually needs to happen next. You'll find the episode notes and direct Microsoft sources on our website, including the setup details and limitations we touched on. Thanks for listening to Central Perspectives. See you next time.