Before you make the change

Review deferrals before posting

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 what becomes easier when Business Central makes a distinction visible: invoices before they're posted, materials used versus materials picked, and software that still runs versus software that does the right thing. Some of these are current release changes; others are established features worth finding. We'll start in finance, where a useful view arrived with Business Central 27, in the 2025 second release wave: analyzing unposted deferral schedules across documents. Now, let's get started.

Defer revenues and expenses

Deferral reporting enhancements

Alex: And the useful word there is unposted. A deferral spreads recognition across periods instead of treating the whole invoice amount as revenue or expense immediately. The new view lets finance examine those planned slices together before committing the documents. Imagine a company preparing several annual service invoices. Reviewing each invoice can tell you whether its total looks reasonable. Looking across their schedules can reveal that one starts recognizing revenue a month earlier than the others. The total invoice amount wasn't the problem. The timing was. That's a different question, and it's much easier to ask when you can see the schedules together.

Deferral reporting enhancements

Emma: That sounds more useful than another total to reconcile. But let's keep the slightly awkward invoice. Suppose the service genuinely begins earlier for that customer. I wouldn't want the person reviewing this to make every schedule match just because a neat column feels reassuring. What would make the difference worth investigating rather than correcting?

Alex: The agreement behind it. In our example, compare the schedule with when that particular service is provided, using the company's accounting policy. If the earlier start is justified, leave it. The odd-looking line has helped you ask a question, not answered it. I'd actually prefer a review where the controller can explain two different schedules to one where they've made everything identical. The software can organize the dates and amounts. It cannot infer from matching invoice totals that the underlying obligations are identical. And if nobody has the service dates yet, the right outcome may be to resolve that missing information before posting.

Emma: So this can expose a missing conversation, not just a wrong date. I can see a sales colleague saying, 'The invoice is ready; why are we still discussing it?' While finance is really asking when the service starts. Where does someone find this view without opening every invoice again?

Alex: Look for Sales Deferral Lines, Purchasing Deferral Lines, or G/L Deferral Lines, and use analysis mode on the relevant list. Those are separate views for the source you're reviewing. The underlying deferral setup still matters: accounts, templates, and actual schedules must exist. This isn't a machine that reads every invoice and invents the missing accounting treatment. For our service invoices, I'd begin with the sales view and a modest question: do the planned recognition dates agree with the service dates we've approved? That gives the person opening it a reason to look, rather than an instruction to explore a large grid and somehow discover a problem.

Defer revenues and expenses

Deferral reporting enhancements

Emma: Let's make the amount easy to follow. In this imagined company, we have a twelve-thousand invoice covering twelve months, and the agreed pattern is one thousand a month. One schedule begins in January; another begins in February. Both add up perfectly. If I only inspect the annual total, I've missed the question entirely.

Alex: Yes, and January's result is different even though those annual totals agree. There's another useful variation: suppose one twelve-thousand invoice has been spread across six months instead. Now the monthly pattern changes as well as the end date. You don't need a sophisticated forecast to spot that it deserves attention. But I'd keep our arithmetic separate from a promise about the product's calculation. We've stipulated equal monthly amounts to make the example understandable. Real schedules can use different calculation methods, and the appropriate treatment depends on the transaction. Start with what this invoice is supposed to represent, then see whether its schedule represents that.

Deferral Calculation Method enum

Emma: The six-month one makes me wonder about the template. If several people keep adjusting the same default, perhaps we're asking them to repair setup through individual invoices. But if this is one unusual contract, changing the default could spread the exception to everybody else. Those are quite different jobs.

Alex: They are, and Business Central keeps that distinction: changing a document's deferral schedule doesn't change the template behind it. So our unusual contract can have its own schedule without turning that exception into the next person's starting point. For repeated adjustments, I'd investigate the template separately. Is it genuinely wrong, or are two types of service being squeezed into one default? Those lead to different improvements. I wouldn't begin with a bulk change to historical drafts. First establish which future transactions should use which pattern, then review the drafts that actually need attention. That's a setup decision informed by the review, not an automatic consequence of opening the analysis.

Defer revenues and expenses

Emma: I'd also want the review to survive someone posting an invoice during the morning. Imagine finance sends a screenshot at nine, and another colleague posts two of those documents at ten. By lunchtime, the screenshot and the current draft view tell different stories. Neither necessarily means the numbers went wrong.

Alex: That’s a reason to name what the review covers and when it was taken. ‘These are the draft service invoices we reviewed this morning’ is useful. ‘This is our deferred revenue’ is too broad. For posted activity, the Sales Deferral Summary reports recognized and remaining amounts by customer and document, as of a balance date. The G/L Deferral Summary has the posted account perspective. Those reports answer a different question from our draft review. If you want a management picture combining posted activity and proposed transactions, define the two populations explicitly. Otherwise you can count the same commercial event twice, once from an old draft extract and again after it’s posted.

Sales Deferral Summary report

G/L Deferral Summary report

Emma: Then I’d resist calling this a forecast in the meeting pack. There could be service work we haven’t invoiced, proposals that never become invoices, or a draft nobody intends to post. The interesting discovery is that finance can review what’s already been prepared without treating it as the whole future. That feels smaller than a forecast, but much more dependable.

Alex: And it changes where the effort goes. If the company currently waits until after posting to notice a surprising recognition pattern, moving that question earlier can avoid a correction later. Not every invoice needs another committee meeting. You might start with the small group where timing varies: annual services, contracts starting midyear, or documents using a newly agreed template. Then ask whether the review finds anything that the existing document checks miss. If it finds nothing and takes an afternoon, narrow the scope. A useful control should make a real decision easier. The benefit isn’t having another report; it’s catching a misunderstanding while the document is still a draft.

Manual consumption and picking

Emma: Our next distinction has a more immediate upgrade angle. Manual consumption without requiring a pick was introduced in Business Central 26. In version 29, that feature transition becomes mandatory. The documentation says existing Manual settings are upgraded to Pick plus Manual. That sounds almost backwards if you’ve just heard that manual consumption can now happen without picking. Why preserve the pick in the migrated setting?

Flush components according to operation output

What is new in Business Central update 29.0

What is new in Business Central update 26.0

Alex: To preserve the old behavior rather than quietly remove a warehouse step. Picking and consumption describe different things. Picking concerns supplying material to production. Consumption records that production used it. Under the new distinction, Manual can record consumption without requiring that pick, while Pick plus Manual preserves picking where warehouse handling requires it. Think of a hypothetical component already kept beside a machine. Someone still needs to report how much was used, but a separate pick for each order may not reflect the way that material is supplied. That is a possible fit for Manual. It isn’t a reason to remove picking for components that really are issued order by order from the warehouse.

Flush components according to operation output

Emma: Then seeing Pick plus Manual after an upgrade might actually be reassuring. The name changed because the old name now means something else. I'd be wary of someone tidying the settings back to Manual because that's what the training screenshot used to show. A cosmetic correction could become a process change.

Flush components according to operation output

Alex: Exactly the change I'd avoid. Keep the migrated behavior unless there's a reason to redesign it. And explain the names in the production team's language. 'Do we still require a pick before this component can be consumed?' is more useful than 'Which enum value did the upgrade choose?' There's no prize for adopting the shortest label. If the warehouse deliberately controls the issue of that material, keeping Pick plus Manual can be the correct result. The new capability gives you another way to describe a different physical process. It doesn't make the existing one obsolete merely because it has a longer name.

Flush components according to operation output

Emma: Let's stay beside that machine. Suppose the company keeps a container of material there and replenishes it separately. The amount used varies from job to job. I can understand not creating an order-specific pick for each handful, but somebody still has to measure the consumption. Does Manual mean we've chosen less automation on purpose?

Alex: Potentially, yes. In that example, the information you value is the quantity actually used. If each finished unit theoretically needs a fixed amount, calculating consumption from output could be convenient. But if actual use varies enough to matter, a person's measurement may be more informative. Removing the pick requirement and automating the quantity are separate choices. I'd keep them separate in the discussion, too. Ask the operator what they can reliably record at the point of work. If the answer is a measured amount at the end of the job, design around that reality rather than choosing a setting because it produces fewer clicks in a demonstration.

Emma: Fewer clicks can be a very persuasive demonstration. I'd ask a less glamorous question: when the container is nearly empty, will the records tell the next shift that? If our operator reports everything on Friday, we may have removed a pick and kept a four-day information gap.

Alex: Then I'd change the proposed process before changing the setting. For that company, the handoff between shifts may be the important reporting point, not Friday's tidy batch. We'd need a practical way to capture what was used and who resolves a missing quantity. That's a trade-off: more timely records require an activity someone can actually perform. If reporting every small withdrawal interrupts production, perhaps the job or shift boundary is the better compromise. Those are choices for the business, not timings imposed by this feature. The software distinction is useful because it lets us discuss that consumption record without pretending an order-specific pick is the same piece of information.

Flush components according to operation output

Emma: Now give that same company a second component: an expensive part kept in a controlled store and issued for a particular order. I wouldn't want our sensible decision for the container to turn into a rule for the whole factory. Can the choice be that specific?

Alex: The documented choice can be made for an item, stockkeeping unit, or production-order component. So yes, we should inspect the relevant records, together with the location's production warehouse handling. I wouldn't infer the final behavior from the item card alone. Your expensive part gives us a useful contrast for a sandbox check: one component intentionally supplied through a pick, one intentionally consumed without that order-specific step. The expected difference should be written down before anybody runs the check. Otherwise a successful posting only tells us that something was allowed, not that the process still enforces the distinction the warehouse wanted.

Flush components according to operation output

Emma: I like the contrast, but I'd add the person who has to explain a discrepancy. If the recorded amount is wrong, is the proposed process clear enough for them to work out whether material was used, moved somewhere else, or simply not reported? 'It posted successfully' won't help them much.

Alex: That changes the check from a happy-path demonstration into a useful rehearsal. In our hypothetical process, give the reviewer a known discrepancy and see whether the records and the operator's explanation agree. We're not asking Business Central to diagnose it automatically. We're asking whether the evidence people rely on still exists after removing a step. If the old pick served an additional purpose, perhaps confirming who received an expensive component, that purpose needs to remain supported. For the container, an order-specific pick might never have provided meaningful evidence in the first place. The two cases should be allowed to reach different conclusions.

Emma: There's a documentation trap here as well. The operational article still describes turning the feature on in Feature Management, which makes sense for earlier versions. But the version 29 overview identifies it as mandatory. So an upgrade conversation shouldn't become a hunt for an optional switch that the current version no longer treats as optional.

Flush components according to operation output

What is new in Business Central update 29.0

What is new in Business Central update 26.0

Alex: Right. Check the environment's version, then read the instructions in that context. And separate two pieces of work: confirming the upgrade preserved the intended process, and deciding whether to use the newly distinct Manual behavior anywhere. The first may finish with no business-process change at all. The second needs the production and warehouse owners involved because it changes how they account for material. That's the opportunity here: stop forcing two different physical arrangements into the same description. It's also a perfectly respectable outcome to say, 'Our components really are picked for each order, so we'll keep doing that.'

Canceling a running update

Emma: Staying with upgrades, there's an established administration feature worth knowing before a difficult morning: you can cancel a running environment update from its operation details. This is not just removing an update from the calendar. The important qualification is what happens after you press Cancel. If the business needs access soon, does canceling get people back in faster?

Manage updates in the Business Central admin center

Cancel a running environment update: release plan

Alex: You can't assume that. Cancellation stops the update and restores the environment to its state just before the update began. That recovery can take more than an hour, and the environment remains inaccessible during it. So the button gives the administrator an intervention, not an instant return to service. Imagine it's approaching the start of the working day and the update is still running. There are two uncertain paths: continuing the update, or canceling and waiting for recovery. The fact that one path is called Cancel doesn't make it the shorter one. You need the current operation status and a clear business priority, not just frustration with the elapsed time.

Manage updates in the Business Central admin center

Emma: Then 'We've canceled it' is a poor message to the people waiting. They might hear 'You can log in now.' I'd rather tell them recovery has started and we're checking access before giving an all-clear. But shouldn't the update window already protect the start of the working day?

Alex: Ordinary updates are automatically canceled if they haven't succeeded by the end of that window. Manual cancellation matters when you need to intervene earlier, or when the update has been allowed to run outside the window. There's also a hard limit: cancellation isn't available when the source version is in its Enforced Update Period. That means our morning decision starts with what is actually available for this operation. If cancellation is available, we can weigh it. If it isn't, spending half an hour searching for another way to press the same button doesn't help the people waiting. Focus on status, support where needed, and keeping the business informed.

Manage updates in the Business Central admin center

Emma: Let's say it is available. I'd want the business owner to be involved before the morning turns into an argument about who touched the button. Perhaps today's priority is a time-critical dispatch. Tomorrow, the company might accept a longer outage to complete the update. The same elapsed time could lead to a different decision.

Alex: Yes, and that is worth agreeing before the update begins. Not an automatic 'cancel after forty minutes' rule. More like: who can decide, which business commitment matters, and when will that person be reachable? In your dispatch example, the useful question is whether an intervention has a credible chance of improving the outcome, given the information we actually have. We shouldn't invent a completion estimate for either path. If neither path has a dependable finish time, the business may need to activate its agreed contingency rather than wait for a reassuring number that the administrator cannot honestly provide.

Default methods in AL interfaces

Emma: That's a useful limit on the promise: more control, without a guaranteed faster recovery. Now for developers, version 29 brings another kind of flexibility. AL interfaces can have default method implementations in the 2026 second release wave, with runtime 18. The headline is that you can add functionality without forcing every existing implementor to change immediately. For listeners who don't write AL, what agreement is being changed here?

Interfaces in AL

Choose an AL runtime version

Alex: An interface says which operations different pieces of code agree to provide. Imagine a custom document-export feature with several exporters behind it. The calling code asks each exporter to produce a document, without needing to know its internal details. That's our hypothetical design, not a standard feature we're announcing. Now suppose the owner wants an additional operation: ask the exporter for a friendly description. A default method can provide common behavior, while a particular exporter can supply its own. The existing implementations aren't all forced to write that new method just to keep participating. The interesting design question is whether the shared answer is genuinely useful for every implementation that inherits it.

Interfaces in AL

Emma: A description sounds forgiving. If it says 'Document exporter' instead of a lovely specific name, the feature might still be understandable. But imagine the new question is 'Can this exporter handle confidential documents?' A cheerful default answer would make me nervous. Both additions could be technically optional, while the consequences are very different.

Alex: That's where I'd stop treating the default as a convenience and write down its meaning. In our design, a generic description can be truthful. A default claim about confidentiality cannot be inferred from the fact that an exporter exists. If the implementation has never supplied that information, the caller should not silently interpret missing knowledge as approval. We could design a result that means the capability is unknown or unsupported, with the calling workflow responding appropriately. Or decide this capability needs an explicit implementation before it is offered. Those are design alternatives, not behavior that Business Central supplies automatically. The language feature lets us express a fallback; it doesn't choose a safe business meaning for us.

Emma: I'd choose the honest unknown over an invented yes. But there's a cost on the other side: if every older exporter suddenly looks unsupported, the user may lose a workflow that used to work. How do you avoid turning a cautious fallback into an accidental feature shutdown?

Alex: By not making the old workflow depend on the new answer until you've decided that's necessary. Suppose ordinary exports continue under the existing rules, while a new confidential-export option requires explicit support. Then an older implementation can keep doing its established job without gaining a capability nobody verified. If the business instead decides that every export now needs that stronger condition, that's a deliberate migration with consequences for users. It should not arrive disguised as a harmless extra method. Your objection changes what I'd test: not only whether the new option is blocked correctly, but whether the ordinary export still works as intended for an unchanged implementation.

Emma: That gives the compatibility claim a much clearer boundary. Microsoft also says adding a default method doesn't change the interface's runtime identifier, so dependent extensions don't need recompilation for that addition. I can imagine a system owner hearing that and concluding there's nothing to coordinate. What is that identifier actually protecting?

Interface method lifecycle

Alex: It represents the required methods in the interface's runtime contract. A default method doesn't add another required method to that identity. That removes a specific compatibility obstacle, and it's a real improvement. But it doesn't evaluate our export policy, the wording a user sees, or what a caller does with an unknown result. Think of two questions: can the pieces still fit together, and does the combined behavior still mean what we intend? The feature helps with the first. We still need to examine the second. I'd coordinate around the changed behavior, not automatically schedule every dependent team for a rebuild they may not need.

Interface method lifecycle

Emma: And sometimes that could be pleasantly uneventful. We add the friendly description, older exporters use the generic wording, and only the ones that benefit from a better label need attention. I don't want our confidential-document example to make every small interface addition sound like a security project.

Alex: It shouldn't. A small truthful default is exactly where this can reduce coordination work. For our description example, compare an unchanged exporter with one that supplies a specific description. Check that the caller displays both sensibly and that an ordinary export still behaves as expected. That's a focused check tied to the change. We don't need to invent an elaborate migration just because the language now offers a new tool. The stronger example is there to show where the same shortcut would stop being appropriate. Good defaults make common cases boring in a useful way; they shouldn't conceal decisions that every implementation needs to make for itself.

Emma: What if the common fallback is useful now, but the owner knows every exporter will eventually need its own answer? The documentation introduces RequiredPending for that transition. That name sounds like a promise to developers: you can keep going today, but this will become your responsibility.

RequiredPending attribute

Alex: That's the intended progression. The default body remains, but RequiredPending makes the compiler warn implementors that don't override it. In a later major version, the owner can remove the body and the attribute, making the method required. That later change affects runtime identity and requires dependent extensions to be recompiled. AppSourceCop distinguishes that planned transition from simply removing the body without the warning phase. The technical notes link to the detailed lifecycle. The practical point is that 'optional for now' needs a migration plan people can act on. A warning isn't evidence that the other team has seen it, scheduled the work, or released its update.

Interface method lifecycle

RequiredPending attribute

Emma: Especially if that team hasn't rebuilt its extension recently. The warning lives in a developer workflow; the person planning the business upgrade may never see it. I'd want the reason and the intended release communicated in ordinary language as well. Otherwise 'we gave notice' can mean 'we added text to a place nobody opened.'

Alex: And the reason should say what behavior they need to supply, not merely that a method must exist. In our exporter design, a stub that returns a convenient value could satisfy a superficial implementation task while leaving the actual question unanswered. I'd ask the owning team to demonstrate the intended result for its exporter and the caller to demonstrate how it uses that result. That closes the loop on both sides of the agreement. So default methods can reduce unnecessary synchronized work, and RequiredPending can make necessary work gradual. Neither removes the need to decide what the shared contract means. That decision is where the useful design effort belongs.

Item references in daily work

Emma: We'll finish with something smaller and immediately recognizable: a customer sends you a product number that isn't your product number. Business Central's item references are an established way to connect those terms, including customer or vendor descriptions, units of measure, and variants. This isn't a new version 29 announcement. It's a useful discovery for anyone translating someone else's numbering before they can enter an order.

Use item references

Alex: Imagine the customer orders its code BLUE-TWELVE. Inside the company, the item has a different number, and that customer's code means the blue variant in a carton. A configured reference can carry that relationship onto the document when the reference is selected. It's more than a nickname: where the reference includes a variant or unit of measure, those values are copied too, and its description can replace the line description. So the value is not merely avoiding a lookup. It's helping the order reflect what that customer's code actually means. The reference still needs to have been set up correctly.

Use item references

Emma: The carton is the detail I'd check first. If the customer asks for two, do they mean two pieces or two cartons? An item number that looks correct could still leave us with the wrong quantity of goods. I'd rather establish that meaning once than rely on the most experienced order clerk remembering it forever.

Alex: That's a good place to start small. Take one real trading relationship and confirm the meaning of the reference against an agreed order example. In our invented case, check that the internal item, blue variant, and carton unit all match the request before treating the setup as ready. Then give someone responsibility for maintaining the relationship when the customer changes its pack or numbering. A reference can save repeated interpretation, but only if the stored interpretation remains right. I wouldn't create a second item merely to make its number look like the customer's unless there were some separate business reason for a distinct item.

Emma: There's a mobile angle as well. Microsoft documents barcode scanning for item references in the Business Central mobile app on supported pages, including purchase and sales lines. I can see the appeal when the number is on a label in front of you. Does a successful scan settle our carton question?

Use item references

Alex: It helps enter the reference; it doesn’t establish that the physical contents match the setup. In our carton example, a clear barcode can still point to an outdated interpretation. I’d use scanning to remove typing, then check the resulting line and the actual pack during a small trial on the supported mobile page. If that works, you’ve removed a very ordinary source of friction without making a large project of it. And if the company only needs the reference on desktop orders, that’s useful on its own. The mobile option is an extra entry route, not a requirement for benefiting from customer or vendor references.

Emma: That’s a good place to leave it: a small improvement can be enough. Today we found a chance to question a schedule before posting, a clearer choice about how production records material, and a way for developers to evolve an agreement without making every participant change at once. We also found where the promise stops: canceling an update takes recovery, and reading a reference doesn’t verify the goods. None of those decisions is settled by a new label alone. The useful next step is to choose the distinction that matters in your own work and check it against an actual task. You’ll find the direct sources and technical detail in the episode notes. Thanks for listening to Central Perspectives from ITrators. Until next time.