

Better control, less busywork

This transcript follows the approved script. Timestamps use approximate automatic alignment; verbatim agreement with the recording has not been verified.

Give the release its own schedule

Emma: Welcome to Business Central Podcast from ITrators. I'm Emma, with Alex. We're AI hosts, and our scripts and voices are AI-generated. Today we're exploring discoveries from the October release, plus practical possibilities for planning deployments, using AI, and making everyday reporting less cumbersome. Now, let's get started. Alex, can scheduling an extension update finally let someone close the laptop before dinner?

[Managing Apps](#)

Alex: Yes. In the online Admin Center, you can queue an update to a company-specific extension for the environment's next update window. Nobody needs to sit there waiting to press Install that evening.

[Managing Apps](#)

Emma: Good. But who's checking it worked before the warehouse starts tomorrow morning?

Alex: That's still somebody's job. I'd agree who checks the installation result and tries the changed warehouse step before the shift. Scheduling removes the late click, not that responsibility.

Emma: As long as we haven't swapped a late evening for a dawn shift.

Alex: Fair point. If checking it means dragging someone out of bed, I'd choose another window. The aim is less waiting, not a different antisocial hour.

Emma: Then let's leave that person asleep. What if an installed customization needs its next version to arrive with the next major Business Central upgrade? How would you line those up?

Alex: That's where I'd choose Next Major Update when uploading the new extension version. Instead of sending it into the next ordinary update window, you're tying its installation to that Business Central upgrade. You can prepare the package beforehand without asking people to start using the changed customization on the version they're running today.

[Managing Apps](#)

Emma: That would make a coordinated change less of a last-minute scramble. But suppose the same package also fixes something sales keeps struggling with today. Would you hold that fix back just to keep everything together?

Alex: Not just for a tidy release plan. I'd ask the developer whether that fix can be delivered separately for the version sales uses now. If it can, we could put the fix into the next update window and keep the upgrade-specific changes with the major release. I'd rather give sales the improvement than make them wait for unrelated work.

[Managing Apps](#)

Emma: I like that distinction. From the sales desk, a small fix can matter more than the big upgrade. And this Admin Center capability became generally available in July, so it's worth asking whoever handles your deployments whether they've started using it.

[Upload, install, and manage PTEs in the admin center](#)

Alex: And once we've queued that fix, we can follow the deployment in Operations in the Admin Center. There's also a useful way to change our minds: while the scheduled installation is still pending, we can cancel it from the app's details. So preparing it this afternoon doesn't mean we're committed to letting it install tonight.

[Managing Apps](#)

Emma: That makes preparing it early much more appealing. If sales raises a concern that afternoon, there's still room to listen instead of saying we've already set everything in motion.

[Managing Apps](#)

Emma: Finance might need more than a different evening. Suppose they're closing the month and want the Business Central upgrade itself to wait until they're finished. How much control do they have over that date?

Alex: For an available Business Central version, the administrator can choose a date within the allowed update period under Next Update. So I'd look for a date after their close, within that period. This is the environment upgrade, not just our extension, and users can't work in that environment while it runs. Finance has a real reason to help choose the day.

[Managing updates in the admin center](#)

Emma: A date after close is something finance can actually plan around. Can they arrange that ahead of time, while the version is still only planned?

Alex: They can agree which dates would suit finance, but a planned version isn't yet an available update to schedule. Once it becomes available, the administrator can choose within its allowed period. I'd keep that preferred date provisional until we can actually set it.

[Managing updates in the admin center](#)

Emma: Provisional, then. And for anyone still uploading their company extensions from inside Business Central, Microsoft has marked that older upload route for retirement, with removal planned for the first release wave of 2027. It hasn't gone away yet. Would you move that routine to the Admin Center now?

[Deploying a tenant customization](#)

Alex: Yes, I'd use the next suitable sandbox deployment to get familiar with it, rather than wait until the old route disappears. For our sales fix, that means seeing the package go through the new route before using it in production. There's a useful reason to move now: we get the scheduling choices we've been discussing.

[Managing Apps](#)

Emma: That's a better reason to start than simply watching a deadline approach.

Try the invoice work, not just the demo

Emma: Finance has a chance to do something similar with its own invoices. Payables Agent has a trial for up to fifty PDF invoices without connecting a mailbox first. That arrived earlier this year, not just in October. I like being able to see what it does with a familiar invoice before bringing the inbox into it.

[Set up Payables Agent](#)

[Try Payables Agent free without setting up email](#)

Alex: Yes, because you already know what a sensible result looks like. Uploading the PDF starts the trial in an eligible production environment, before the agent has been activated; those first fifty invoices don't consume billable credits. I'd choose an invoice tied to a purchase order. Then you're not just judging whether it can read a number. You're seeing whether its proposed match gets finance closer to an invoice they can actually work with.

[Set up Payables Agent](#)

[Review and match incoming invoice drafts to orders](#)

Emma: Suppose the supplier invoices ten units, but we've only received six. It might have found exactly the right order and still leave finance with something to sort out. What would they actually have in front of them at that point?

Alex: They'd have a draft with a proposed order match to review. Finalizing that draft creates an unposted purchase invoice; the agent never posts it. And finding the right order doesn't make those other four units received. Business Central checks quantities and receipts using fixed validation rules, not the agent's confidence. Finance still has to establish whether the delivery is short or the receipt simply hasn't been recorded.

[Review and match incoming invoice drafts to orders](#)

Emma: Then the slow part might still be waiting for someone in the warehouse to answer. I can see the appeal of having the draft ready, but if finance spends the morning chasing those four units anyway, I'd want more than a count of processed PDFs to tell me this was helping.

Alex: Fair, but I'd still count the typing they didn't have to do. Waiting two hours for a receipt doesn't mean they spent two hours working on that invoice. I'd compare the minutes spent entering and correcting it, separately from how long it sat waiting. If the draft saves ten minutes and the warehouse delay stays exactly the same, that's a useful result, not a failed trial.

Emma: I'd still want to see that saving in the work, not just in the PDF count. And I'd put ordinary invoices beside our awkward one: goods received, amount agreed, just the familiar entry work repeated all morning.

Alex: Yes. I'd rather have them reviewing what's changed on that familiar invoice than entering the same details again.

Emma: And once that becomes the easier way, I can imagine finance wanting to keep using it. There's a detail in the trial worth knowing before that happens: after fifty invoices, it automatically continues in billable full mode, under the environment's billing setup. Fifty isn't a stop sign where it asks you to start again.

[Set up Payables Agent](#)

Alex: Then I'd review the results before those fifty are used up, with whoever is responsible for the trial and its cost. Deferring the mailbox doesn't remove the billing prerequisites. If the work is genuinely easier, they can choose to keep it running rather than simply drift past the trial limit.

[Set up Payables Agent](#)

Emma: Suppose they do choose to continue and connect the mailbox. Later, the credits run out, so new invoices stop being picked up. If finance decides to handle them manually while they reconsider the cost, can they treat that pause as the agent being switched off?

[Set up Payables Agent](#)

Alex: No. It's still active and scheduled; it just can't process new mailbox items without credits. When credits become available again, it resumes automatically, including the unread backlog. If finance wants to take over while they decide, turn Active off. Otherwise someone replenishing credits could restart the intake while the team thinks it's still paused.

[Set up Payables Agent](#)

Emma: And for the new mail, there's Known Senders, with Ask, Approve and Reject choices. Our regular supplier might be an obvious sender to approve, even though we still disagree about those four units. What exactly are we giving permission for when we approve that sender?

[Manage known senders for Payables Agent](#)

Alex: Permission to process their email, not approval of what they're charging. Those four units are still a receipt question. Known Senders can remove repeated email-review decisions for that supplier when you're using Manage per sender without a monitored subfolder. If you configure a monitored subfolder, those sender policies and the email-review setting aren't used. That's a different way of deciding what reaches the agent.

[Manage known senders for Payables Agent](#)

Change the footer once

Emma: There's another kind of repeated work in the documents we send out. The company changes its contact details, and someone has to fix the footer on invoices, order confirmations and who knows what else. October brings reusable report themes and headers and footers. Can that turn the same edit in several places into one shared change?

[Update 29.0 for Business Central 2026 release wave 2](#)

[Report and document layouts overview](#)

[Set up reusable themes and header and footer layouts](#)

Alex: Yes, for the layouts using that shared footer. The invoice keeps its lines and totals in a report-specific Word Body layout, while the header and footer come from a reusable part. Change the contact details there, and both the invoice and order confirmation can pick them up without changing their individual bodies.

[Report and document layouts overview](#)

[Set up reusable themes and header and footer layouts](#)

Emma: And it removes that irritating moment when the invoice has the new details but the order confirmation still has the old ones. Suppose a company already uses Word layouts that look exactly how they want. Can they attach a shared footer to those, or does this mean changing the layouts first?

Alex: They need a Word layout with the Body subtype. An ordinary Word layout doesn't become one just because the environment is upgraded. The New Document Report Experience also needs enabling; it's off by default in version twenty-nine. I'd treat the existing design as the reference, then separate its shared header and footer from the invoice-specific content. It's a layout change, not necessarily a visual redesign.

[Update 29.0 for Business Central 2026 release wave 2](#)

[Report and document layouts overview](#)

[Set up reusable themes and header and footer layouts](#)

Emma: Keeping the design would help. Nobody asked for a rebrand just because the phone number changed. Although I'd still want a subsidiary to keep its own contact details, even if both companies use the same fonts and colours. Making maintenance easier shouldn't mean making those documents identical.

Alex: It doesn't have to. The theme and the header and footer are selected independently, so the companies can have the same visual style with different contact details. You can override the choices for a particular report or layout too. I'd keep those differences where they belong, rather than copy an entire invoice design just to change the company at the bottom.

[Set up reusable themes and header and footer layouts](#)

Emma: For our contact-detail change, I'd want to see the short order confirmation beside a long invoice. A footer can look lovely on one page, but the customer still needs to find the total when the lines run over several pages.

Alex: I'd go straight to the last page of that invoice. That's where I'd want the final lines and total to read properly, with the footer leaving them enough room. And I'd compare the generated PDF, not just the layout open in Word.

Emma: There's also a separate October improvement for the person assembling the month-end PDF. Financial report packages can bring selected reports together in one file. Think of the profit and loss, balance sheet and cash-flow report someone currently exports separately and then merges. That feels like a very welcome job to lose.

[View a financial report](#)

Alex: Happily. Though I'd actually run that month-end package manually at first. You can schedule it, but the calendar reaching Monday doesn't necessarily mean finance has finished the month. Set the filters for each report in the package, then generate the combined PDF when they're ready to read it.

[View a financial report](#)

Emma: Yes, I'd happily keep that one click if it means the last adjustment actually makes it into the pack.

Alex: Who are we making this pack for, though? If it's management comparing this month with last month, I'd want to know whether they're looking at the whole company or a particular department. That's a more consequential choice than how we join the PDFs.

Emma: Let's make it the management team, looking at the whole company. They want the overall result, but also a separate view of the service department because its costs have climbed. I'd want both in the pack, without leaving the reader to guess which figures cover which part of the business.

Alex: Then I'd keep the company report unfiltered by department and apply the service dimension filter to its separate report, assuming that's how they track departments. Same comparison periods, clearly labelled scope. But you've made me curious about that cost increase: did service actually spend more, or did someone change which accounts the report includes? I'd want to separate those before discussing the department's performance.

[View a financial report](#)

Emma: I hadn't given you enough to tell. And that makes the other reporting change rather useful here: this release wave automatically logs changes to financial report definitions. If the report has changed underneath the same heading, at least there's something to investigate beyond everyone's memory of last month.

[Auditing and financial reporting](#)

Alex: I'd look at the row definition behind those service costs. The automatic log covers row and column definitions as well as the report itself, so a changed account range gives us a specific lead. Then we can compare the figures on the same basis. That's quite different from knowing that someone ran or exported the report.

[Auditing and financial reporting](#)

A connected answer still needs a clear question

Emma: Not every question needs a monthly pack, either. Microsoft's native MCP connection for Business Central online caught my eye. Suppose someone in finance wants to ask an AI assistant which overdue customer balances to follow up today, instead of exporting another spreadsheet. Is that a sensible use for this connection?

[Model Context Protocol \(MCP\) in Business Central overview](#)

Alex: Yes. MCP gives a compatible AI assistant a way to discover and use the Business Central tools it's allowed to use. So it could retrieve the relevant figures rather than ask you to paste them into a chat. But for your finance colleague, is the goal a list of everything overdue, or a shorter list of customers worth calling first? Those are different questions.

[Model Context Protocol \(MCP\) in Business Central overview](#)

Emma: The shorter list. They've got half an hour before lunch and want to make a few useful calls. Let's keep it to the Danish company, as of today. I'd want the amount still owed and how overdue it is, with a reason each customer made the list.

Alex: I'd ask for overdue outstanding amounts, not just each customer's total balance. They might owe a lot that isn't due yet. For the shortlist, I'd want the underlying invoices and due dates alongside the reason for the ranking. Then finance can ask a useful follow-up, like showing which invoices account for one customer's overdue amount, without rebuilding the spreadsheet.

Emma: That's the useful bit for me: the invoice details ready before the call. Would the assistant see them through that finance colleague's own Business Central access?

Alex: Yes, the native online connection operates with the signed-in user's identity and permissions. But that doesn't expose everything they can see on every Business Central page. The connection also determines which tools and APIs are available. For our call list, I'd first check that those actually expose the outstanding amounts and due dates we need.

[Model Context Protocol \(MCP\) in Business Central overview](#)

[Configure the Business Central MCP server](#)

Emma: Fair. And even if that colleague can change customer records in Business Central, I only want the assistant reading for this task. Preparing a call list doesn't need permission to change anybody's payment terms.

Alex: Then I'd make that a property of the connection, not just an instruction in the chat. The built-in default discovers eligible read-only APIs, which fits this job. An administrator can designate a different default, though, so I'd verify which configuration this assistant actually uses. Your colleague's broader permissions don't mean we need to expose broader tools.

[Model Context Protocol \(MCP\) in Business Central overview](#)

[Configure the Business Central MCP server](#)

Emma: The documentation needs a careful read there. The version twenty-nine overview has a general-availability headline for data querying, but the current setup guide still labels Data Query Tools as preview. When you say we'd use read-only APIs, are you proposing that preview route for our call list?

[Configure the Business Central MCP server](#)

Alex: Not necessarily. API queries expose defined, read-only views of data. Their MCP support has a general-availability record from June. The direct Data Query Tools are a separate route, still explicitly preview in that setup guide. I'd first see whether the existing eligible APIs answer our overdue question. If they do, we haven't gained anything by choosing the newer query route just because it's newer.

[Configure the Business Central MCP server](#)

[Unlock more Business Central data with MCP server enhancements](#)

Emma: Then let's use what does the job. Where does our finance colleague actually ask the question? They already have an AI chat open, but that isn't necessarily connected to Business Central.

[Use Business Central MCP with non-Microsoft clients](#)

Alex: In a compatible AI client with the Business Central MCP connection configured, signed in as that colleague. Depending on the client, someone may need to register the application and arrange consent first. Once that's in place, the question can stay ordinary language. They shouldn't have to learn the API names just to ask for those overdue invoices.

[Model Context Protocol \(MCP\) in Business Central overview](#)

[Use Business Central MCP with non-Microsoft clients](#)

Emma: Good. I'd start with five customers, not fifty. And I'd want to be able to ask, 'Why this customer ahead of that one?' A small old debt and a large invoice that's only just overdue might lead to quite different calls.

Alex: I'd ask for both rankings: largest overdue amount and longest overdue. Then we can see who moves. A confident sentence saying 'highest priority' hides that choice; two short lists make it visible. For those five calls, which matters more to your colleague today?

Emma: Today, the largest overdue amounts. They want to clarify when the bigger payments are coming in. I'd keep the oldest-debt list for a separate session rather than pretend these five calls will tidy up everything.

Alex: Then I'd group the overdue invoices by customer for those calls. And put the amounts on a common currency basis before ranking them, if this company invoices in more than one currency. Five thousand euros and five thousand kroner shouldn't land side by side as if they're equal.

Emma: Now suppose one of those calls reveals that the customer's contact details have changed. We deliberately chose read-only for the preparation. If finance later wants the assistant to help update records too, is that a different configuration rather than just a different request in the chat?

Alex: Yes. A request in the chat doesn't turn our read-only connection into an editor. API queries remain read-only. API pages can expose writes, but editing must be enabled in the configuration, the particular operation allowed, and the signed-in user permitted to perform it. We'd also need to establish that the relevant contact field is available through that API.

[Configure the Business Central MCP server](#)

Emma: I'd leave that update in Business Central for now. Our colleague wanted to get those calls made before lunch, not redesign how customer records are maintained.

Alex: Agreed. I'd rather get the invoice list in front of them and hear which follow-up question they actually ask. That might be quite different from the next feature we'd have built for them.

Fewer indexes is not always faster

Emma: There's a less visible change in the release notes too: more control over database indexes. Index management itself isn't new this October. But the idea of switching an index off sounds counterintuitive when people want Business Central to run faster. What would make you consider it?

[Update 29.0 for Business Central 2026 release wave 2](#)

[Manage database index usage](#)

Alex: A busy import is a good example. An index is a shortcut for finding data, but the database also has to maintain it as that data changes. If an index barely helps the reads in this company, we're paying that maintenance cost during writes for little benefit. Removing the right one could help the import. Removing a useful one could make the next report slower.

[Manage database index usage](#)

Emma: That next report is why the timing detail caught my eye. Microsoft says disabling takes effect immediately, but re-enabling is queued for the next scheduled midnight process. So this isn't a switch you flick back ten minutes later if the report starts crawling.

[Manage database index usage](#)

Alex: Which changes the plan for that import. If finance runs a critical report straight afterwards, I wouldn't disable an index on the assumption that we can restore it before they begin. I'd want to know how that report behaves without it first. The import might finish sooner while finance spends the rest of the afternoon waiting.

[Manage database index usage](#)

Emma: So the administrator needs more than an import stopwatch. They can inspect index usage for the company, but if an index looks quiet, how do they tell whether the report simply hasn't run during the period they're looking at?

[Manage database index usage](#)

Alex: The counter alone can't tell them. I'd ask finance when that report last ran and whether month-end work falls inside the observation period. These are SQL usage counters, and they reset when the database engine starts. So zero might describe a quiet morning, not the life of the system. We need to know which work those numbers actually represent.

[Index usage statistics reference](#)

Emma: Finance might also tell you that report was retired six months ago. That's a rather different answer from 'we only run it at month-end.' The conversation could reveal a real opportunity, not just another reason to leave everything alone.

Alex: Yes, that would get my attention. If the remaining workload doesn't benefit from it either, the import could be maintaining a shortcut to a report nobody needs anymore. That's a candidate I'd want to investigate with the administrator. The business has moved on; the database may still be doing work for the old process.

[Manage database index usage](#)

Check the connection, then the workflow

Emma: Some old connections, though, may still be doing very necessary work. Version twenty-nine removes support for exposing Microsoft's user-interface pages through SOAP. That's not the end of every SOAP service. But suppose a webshop uses one of those pages to send orders into Business Central. Could the order screen still open normally while that connection stops working?

[SOAP web services](#)

Alex: Yes. The browser and the webshop enter through different doors. The page can still work for a person while the old interface another program calls is no longer supported after that environment moves to version twenty-nine. I'd want to know what the webshop actually sends. Is it creating an order once, or does it also change quantities and delivery details later?

[SOAP web services](#)

Emma: Let's say both. The customer orders three items, then changes the delivery address before the warehouse starts picking. The shop confirms the change to the customer. If only the original order reaches Business Central, the warehouse could send the right goods to the wrong place.

Alex: Then a test that only creates the order misses the important second half. I'd send the address change through the webshop's real integration in a test environment and check the same order in Business Central. Does the changed address arrive where the warehouse uses it? A successful response to the request isn't enough if it updated the customer card instead of this order's delivery address.

Emma: Especially if it's a one-off delivery to a different site. Updating the customer's usual address would solve the wrong problem.

Alex: Exactly. That's why I'd map the create and update operations before picking the replacement interface. Microsoft's recommended direction is API pages and queries, but a query won't perform that address update. We need an appropriate writable interface and the right order identified. This isn't a job where changing SOAP to another name in the connection settings proves the migration is finished.

[SOAP web services](#)

[Configure the Business Central MCP server](#)

Emma: Then the webshop team needs to be in that test, not just handed a new connection address afterwards. They're the ones who know what the customer was told when the delivery change was accepted.

Alex: And I'd ask them what 'accepted' means. Has Business Central confirmed the change, or has the shop only queued a message to send later? That tells us where to look if the customer sees a confirmation but the order still has the old address. I'd follow that one order through both systems, including the acknowledgement back to the shop.

Emma: And once that order has arrived correctly, someone still has to work with it. Page Scripting reaches general availability in version twenty-nine, although recording and replay aren't new. How would you use it for the colleague handling that order inside Business Central?

[Update 29.0 for Business Central 2026 release wave 2](#)

Alex: I'd record the supported steps they take on the order in the Business Central web client, then add checks for the values they expect to see. We can replay that after an upgrade instead of relying on someone remembering every step. But what should this colleague accomplish with our order? Just inspect the changed address, or get it ready for the warehouse?

[Use the page scripting tool for acceptance testing](#)

Emma: Get it ready for the warehouse. They need to recognise the webshop order, see the customer's revised delivery address and check that the ordered quantities are still right before releasing it. I'd want the test to catch the old address surviving, even if every button still works.

Alex: Then I'd make that revised delivery address an explicit expected value, not just open the order and move on. Check the quantities too, release the order, and check the resulting status. That gives us a useful failure if the address is wrong, even when navigation succeeds. The webshop test gets the order here; this test checks what the colleague sees and does with it.

[Use the page scripting tool for acceptance testing](#)

Emma: That's much closer to the colleague's actual job than 'the order page opens.' I'd ask them which details they always double-check before letting the warehouse take over. We might not have picked all of those up from the integration description.

Alex: I'd also ask them to show us an order they would refuse to release. Perhaps the delivery instructions contradict the address. That reveals a decision, not just another field to tick off. We can turn the clear, repeatable checks into tests and leave the genuinely ambiguous cases for someone to judge.

Emma: The address says the new site, but the delivery note still says to use the old site's loading bay. Yes, I'd keep that example. Tidying it up before showing anyone would hide exactly why the colleague stops.

Alex: And I'd keep the corrected version as a separate example. In one, the colleague needs to resolve the conflicting instructions; in the other, the order should be ready to release. Otherwise we could accidentally build a test that treats releasing every order as success.

Emma: With the corrected one, though, we've now released our test order. What happens when we replay the same steps tomorrow?

Alex: It starts with an already released order, unless we've deliberately prepared the starting state again. Replay performs the actions; it isn't a video, and stepping backwards doesn't undo the changes. I'd run this in a sandbox with a prepared, unreleased test order each time, containing the address and quantities our checks expect.

[Use the page scripting tool for acceptance testing](#)

Emma: If someone changes that test order from three items to four before the next run, we've changed the question. I wouldn't want the team chasing an upgrade fault when the example itself has moved.

Alex: I'd first compare the failed check with the starting order we intended to use. If it already contained four items, fix the preparation, not the expected answer just to make the test pass. A successful run before the upgrade gives us something useful to compare with, provided we recreate that same starting case afterwards.

Emma: Let's say we've done that, and the address is right when we open the order, but wrong after we release it. That's a rather more useful finding than 'the order test failed.'

Alex: Yes. I'd give the developer that starting order, the replay steps and the address before and after release. They can investigate what ran in between, including any custom logic, rather than starting with the entire webshop connection. We've narrowed down where the result changes, not yet proved which piece of code caused it.

Emma: That would make 'I saw the address change' much easier to act on. The colleague has something they can show, not just try to describe.

Alex: And if we find and fix the cause, I'd keep that address check for the next release. The colleague shouldn't have to rediscover the same problem six months later. Their observation has become a test we can run again.

Emma: Can that colleague make the recording themselves? They know the order routine. I'd rather have them show what happened than ask a developer to guess their way through it.

Alex: Yes, with the recording permission. Whoever replays it needs the playback permission too. I'd have the colleague capture the routine and explain the expected result, then work with the developer on making the test repeatable. Knowing which delivery address is right shouldn't require knowing how to debug the code.

[Use the page scripting tool for acceptance testing](#)

Emma: And if the business changes how those orders should be handled, that colleague needs to be part of changing the test too. The developer shouldn't have to decide what the warehouse ought to receive.

Alex: Right. I'd keep the developer's AL tests for the underlying code alongside this recorded workflow, not replace them with it. And when the handling changes, I'd ask the colleague to walk us through the new outcome before we change the checks. That's a different conversation from fixing a bug.

Emma: There's one more webshop detail worth catching. Microsoft's Shopify guidance says the connector's API calendar isn't the same as the Business Central upgrade window. Suppose you're on the spring connector and your environment's upgrade window runs into February. Can you actually afford to wait that long?

[FAQ for technical details](#)

Alex: I wouldn't plan on February. That spring connector uses Shopify's January twenty-twenty-six API version, and Microsoft's guidance points to upgrading before year-end. The environment's upgrade window doesn't extend Shopify's support for that interface. So I'd bring the webshop owner into the upgrade planning now, rather than treating February as spare time.

[FAQ for technical details](#)

Emma: That changes the plan quite a bit. But is year-end a planning deadline here, or are we saying their orders stop coming through on the first of January?

Alex: A planning deadline, not a prediction that every shop stops on New Year's Day. The sources differ: Shopify currently lists that version as accessible into mid-January, rather than Microsoft's year-end date. Shopify also serves requests for retired versions using a newer supported version. But that doesn't prove the old connector will work correctly against it. I wouldn't make that fallback our upgrade plan.

[FAQ for technical details](#)

[About Shopify API versioning](#)

Emma: The webshop owner needs more than 'it's still responding.' They need the orders to arrive correctly and the stock figures to stay in step. That's what I'd want demonstrated before keeping that February date.

Alex: I'd still challenge February, even if today's sync test passes. First confirm which connector and API version they're actually using, then agree a supported upgrade route with their partner. Test those order and stock movements on the target version. A working connection today doesn't settle what happens after its API version retires.

[FAQ for technical details](#)

[About Shopify API versioning](#)

Emma: Fair. That date needs another look. I'm still particularly pleased about the shared footer: changing a phone number shouldn't mean hunting through every document layout. And I'd bring the colleague who knows the order routine into the upgrade work early, while their knowledge can shape what we check. You'll find the documentation and the date differences we discussed in the episode notes and references. Thanks, Alex, and thanks for listening to Business Central Podcast from ITrators. See you next time.

[Set up reusable themes and header and footer layouts](#)