

# Less guesswork in everyday work

## Find the interface you already have

**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 a few places where a clearer view can save the business some unnecessary work: finding an interface it already has, dividing a cost sensibly, and noticing a change that deserves someone's attention. There are new capabilities here, and useful established ones too. We'll keep that distinction clear. Now, let's get started.

**Emma:** Let's start with a question that can arrive before an integration project has really begun: can our other system get this information from Business Central? Version twenty-nine brings an API Overview page showing the published interfaces in the current environment, including those from installed extensions. Alex, how does that change the first conversation? I'd like the business to get further than, 'Someone will have to build something.'

[Discover APIs with the API Overview page](#)

**Alex:** It gives that conversation somewhere concrete to start. An API is an interface another program can use, rather than a person opening a page and copying information. Suppose the company wants a second system to read a particular business status. Before designing a new interface, we can look at what this environment already publishes. Not just what standard Business Central might offer in general, but what is installed here. That can turn a vague development request into a much smaller question: does an existing interface expose the information we actually need?

[Discover APIs with the API Overview page](#)

[REST API web services](#)

**Emma:** I like that smaller question. Although a list of interfaces could still be a rather elaborate way of telling the colleague to go and ask a developer. The page lets you filter by publisher, group and version. What would you want that colleague to bring to the conversation so the search is about the job, rather than whichever name looks promising?

[Discover APIs with the API Overview page](#)

**Alex:** One real example of the information they need, and what they intend to do with it. In our hypothetical company, 'the status' might mean a document has been created, or it might mean the business work is complete. Those aren't automatically the same thing. I'd ask them to show which record they mean and what decision the receiving system will make. Then the developer can inspect the relevant interfaces and their fields against that example. The publisher and group help narrow the search, but the business meaning is what tells us whether we've found an answer.

[Discover APIs with the API Overview page](#)

[REST API web services](#)

**Emma:** Then let me make that example a little less comfortable. At first the other system only needs to display the status. Later, someone says, 'While we're connecting it, could it update the record as well?' That sounds like one extra sentence in the request. Is it one extra sentence in the work?

**Alex:** That sentence can change which interface is suitable. The overview distinguishes API pages, queries and codeunits. A query is read-only. API pages support reading and writing, while codeunits can expose procedures as unbound actions. So if we found a query that beautifully answers the original display question, it doesn't become a writing interface because the request grew. And even finding an API page doesn't prove that the particular change we want is supported. We still have to examine the operation. I'd keep the original read requirement separate from that newly added write requirement.

[Discover APIs with the API Overview page](#)

**Emma:** That helps explain a detail in Microsoft's guide that could otherwise look like a fault: a codeunit's API URL is deliberately blank in this overview. So if someone sends a screenshot saying, 'This row is missing its link,' we shouldn't immediately turn that into a broken-integration investigation.

[Discover APIs with the API Overview page](#)

**Alex:** Exactly. That row identifies a different kind of capability; the blank isn't proof that it failed. The useful next question is what action the codeunit exposes and whether it matches the business operation. That belongs with the developer, not with someone trying random links. More broadly, this page is an inventory, not an integration wizard. It doesn't configure the API or grant access. We may have found a promising interface and still need the authentication and permissions work before another system can use it.

[Discover APIs with the API Overview page](#)

**Emma:** I don't want that remaining setup to swallow the good news. Discovering that the company already has a suitable interface is still a useful result. Especially if it arrived with an extension rather than the standard application. But that changes who I'd call when the meaning of a field is unclear, doesn't it?

[Discover APIs with the API Overview page](#)

**Alex:** Yes, and that's a good use of the publisher information. It helps identify whose interface we're evaluating. I would want the explanation of its fields and supported operations from the appropriate documentation or owner, rather than assume an extension's field means the same thing as a similarly named standard one. But your point stands: we're evaluating something that already exists. If it meets the need, the result may be a connection and some careful validation, not a new piece of custom code to maintain. That is a perfectly worthwhile outcome.

[Discover APIs with the API Overview page](#)

[REST API web services](#)

**Emma:** And a small naming trap while we're looking at that list: version twenty-nine of Business Central doesn't mean an interface should be labelled version twenty-nine. The overview has API versions, and the routes have version information too. I can imagine someone dismissing an existing interface as old just because it says version two.

[Discover APIs with the API Overview page](#)

[API endpoint structure](#)

**Alex:** Those numbers describe different things. The product version and the API route aren't one shared release counter. I wouldn't judge suitability by comparing the numbers. I'd check the documented interface version, the environment we mean to connect to, and usually the company within it. That last part matters to our original status example: finding the right kind of information is only half the question if the receiving system reads it from the wrong company. The detailed route structure belongs in the notes, but the business can absolutely help identify which company's information it expects.

[API endpoint structure](#)

**Emma:** So I'd bring two examples, not just the easy one. One record where the business work is complete and one where the document exists but the work isn't complete. If the other system treats those as the same thing, a technically successful connection could still give someone the wrong answer.

**Alex:** I'd use that pair in the evaluation. Ask the receiving system to show the difference the colleague cares about, and compare it with the underlying business records. That is more useful than declaring success because a request returned data. If the existing interface supports the distinction, we have a good candidate. If it doesn't, we've identified a specific gap rather than assuming we need to replace everything. The overview hasn't made the integration for us. It has made it easier to start with evidence instead of a blank development brief.

[Discover APIs with the API Overview page](#)

[REST API web services](#)

## Divide the cost without losing the reason

**Emma:** Here's a decision where I'm not sure more automation is automatically better. A company has a shared cost to divide between departments. Business Central's allocation accounts can distribute amounts from documents and general-journal lines using fixed or variable allocations. This is an established feature, not a new October release. My first instinct would be a clear, agreed split that everyone understands. Why would I choose a changing one instead?

[Allocate revenue and costs to multiple general ledger accounts](#)

**Alex:** You might not choose a changing one. If the agreement is that two departments take sixty and forty percent, and that remains a sensible basis, a fixed allocation expresses it plainly. The configured split stays that way until someone changes the setup. Variable allocation earns its place when the business wants the share to follow a measure that genuinely changes. For example, it might agree that a shared staff-related cost should follow departmental headcount. Then the useful question isn't whether variable sounds more sophisticated. It's whether headcount is the basis the business actually intends to use.

[Allocate revenue and costs to multiple general ledger accounts](#)

**Emma:** Let's stay with those departments. Suppose one grows quite sharply, while the other stays the same size. The sixty-forty agreement may still be perfectly clear, but now one manager is looking at the bill and asking why the split hasn't moved. I can see the attraction of a headcount basis there. I'm less enthusiastic about creating a second spreadsheet that somebody has to remember to copy into the system.

**Alex:** That maintenance concern is the part I wouldn't automate away in the explanation. Business Central can use statistical accounts to hold number-based measures such as headcount, and combine that information with dimensions for variable allocation. Statistical entries aren't financial debits and credits; they don't go into the trial balance. But someone still has to supply the appropriate numbers. If we move from an agreed percentage to headcount, we're also choosing a process for keeping headcount meaningful. The calculation can work correctly on an out-of-date measure. That would make the arithmetic consistent, not the allocation useful.

[Allocate revenue and costs to multiple general ledger accounts](#)

[Analyze Data with Statistical Accounts](#)

**Emma:** Then I'd change my proposal. I wouldn't start by switching the cost to variable allocation. I'd first ask the two managers which headcount they mean. The number at month-end? The period the bill covers? A growing department might quite reasonably care about that distinction, rather than just whether the total is correct.

**Alex:** That would change the setup I'd evaluate. Variable allocation has a calculation period, and the basis can be narrowed with filters. So I'd want that period to match the agreement, then inspect the registered statistics for it. We shouldn't quietly substitute 'the latest number we happen to have' for the measure the managers agreed. And there's a legitimate outcome where they decide a fixed split reviewed periodically is good enough. If the variable basis creates more maintenance than useful accuracy, the simpler arrangement hasn't failed just because a more elaborate feature exists.

[Allocate revenue and costs to multiple general ledger accounts](#)

[Analyze Data with Statistical Accounts](#)

**Emma:** There is a nice practical way to make that discussion less abstract: Test Allocation lets you enter a trial amount. If I'm the person trying to explain this to the managers, I'd rather put a recognizable bill in front of them than a screenful of setup. What would you look for in that trial?

[Allocate revenue and costs to multiple general ledger accounts](#)

**Alex:** I'd choose an amount simple enough to reason about. For a hypothetical ten-thousand cost, a sixty-forty split means six thousand and four thousand. That gives everyone an expectation before we run anything. With the headcount basis, we'd work out the expected shares from the agreed period's numbers instead. Then inspect where the amounts go, including the accounts and dimensions, not only whether they add back to ten thousand. A complete total can still hide a wrong destination. The trial is useful because it lets us discuss the result before it becomes a posted result.

[Allocate revenue and costs to multiple general ledger accounts](#)

**Emma:** That wrong destination is the one I can see causing a very long meeting. Here's another wrinkle: most of the bill is shared, but one part was only for one department. I wouldn't want a convenient allocation to make that exception disappear. Does using an allocation account commit us to accepting its first answer?

**Alex:** No. There's a Redistribute Account Allocations action for adjustments before posting, and journal Posting Preview lets you inspect the proposed posting. For your exception, I'd first separate the question of this bill from the standing allocation rule. A one-off department-specific item doesn't necessarily mean the normal sixty-forty agreement should change for every future bill. We'd work through the document and the allocation so that this exception lands where intended, then review the posting. That is different from editing the reusable setup simply to make today's total come out the way we want.

[Allocate revenue and costs to multiple general ledger accounts](#)

**Emma:** There's one boundary I want to catch before anyone tries that in their monthly routine. Microsoft explicitly says Allocation Accounts aren't supported in recurring journals, even though the account-type option appears there. That is an awkward invitation. If the company already divides a recurring cost in a recurring journal, what should it use?

[Allocate recurring costs and income](#)

**Alex:** Use that journal's own Allocations action. It has allocation keys based on quantities, percentages or amounts. So the business may already be in the right place for its recurring process; it doesn't need to move merely because Allocation Accounts are another feature. There's also a specific restriction for allocation accounts in journal batches with an approval workflow for creating journal lines. That's a reason to check the actual batch and workflow before proposing a change, not a reason to announce that allocations and approvals can never coexist anywhere in Business Central.

[Allocate recurring costs and income](#)

[Work with general journals](#)

**Emma:** That makes me less interested in replacing a working monthly process, and more interested in the bills people still split by hand. There's a reuse option called Inherit from parent, where the allocation can use the G/L account selected on the line. I can see why someone managing several kinds of shared costs would like that. Does it make the result harder to follow?

[Allocate revenue and costs to multiple general ledger accounts](#)

**Alex:** It can, and Microsoft explicitly warns that it's less transparent. With a fixed destination in the allocation, you can read the setup and see that account. With Inherit from parent, you also need the account selected on the document or journal line to understand where the cost will land. That may be useful when the same departmental split applies to different expense accounts. But if someone reviews only the allocation setup, they won't have the whole picture. I'd make the actual line and posting preview part of the review, rather than assume reuse has removed the need to understand the destination.

[Allocate revenue and costs to multiple general ledger accounts](#)

**Emma:** I started out wanting the simplest agreed percentage, and I still like it when the agreement is stable. But I can now see a better reason for variable allocation than 'less manual work': the cost may follow a measure that the business actually believes is fairer. The work doesn't vanish. Some of it moves into agreeing and maintaining that measure. I'd rather make that visible before promising the finance team an effortless month-end.

## How far does calculated capacity reach?

**Emma:** Now, a smaller discovery for people planning production. Business Central twenty-nine includes a clearer view of how far calculated capacity calendars extend. Work and machine centers have a Calendar Entries Available Until field. Imagine the planner looking at work beyond year-end: before debating whether the factory can take it on, there's a more basic question. Have we actually calculated the calendar that far? Alex, what does that date tell them?

[Set up shop calendars](#)

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

**Alex:** It shows the latest date for which calendar entries have been calculated. That's the horizon of the calculated information, not a promise that every day up to it has spare capacity. The calendar calculation works from the working pattern and the center's capacity settings. So if our planner is looking into January and the calculated entries stop in December, that is a different issue from January being fully occupied or having fewer working days. Seeing the horizon helps them ask the right question before interpreting the plan.

[Set up shop calendars](#)

**Emma:** The warning has a precise boundary too: a date before the Work Date gets warning styling. That sounds useful for something already out of date. But our planner is looking ahead. December can still be in the future and still be too short a horizon for January's work.

[Set up shop calendars](#)

**Alex:** That's why I'd compare the actual date with the period we intend to plan, rather than wait for the warning. And a blank value isn't marked expired, so a quiet-looking field isn't evidence that the calendar covers the period either. For our January question, the useful check is whether the relevant centers have calculated entries through the dates we need. If they don't, the calendar calculation takes a starting and ending date. The new field points to the gap; it doesn't extend the calendar on its own.

[Set up shop calendars](#)

**Emma:** Suppose they extend it, and January still looks thinner than expected. I wouldn't want the team to keep calculating further and further ahead just because that was the first thing we checked. At that point we need to look at what those January days contain, don't we?

**Alex:** Yes. The assigned shop calendar supplies working days and shifts, and the center's capacity and efficiency also matter. Unspecified weekdays and defined holidays have zero available capacity. Recorded absences can reduce a day's capacity too. So I'd inspect the calculated days in the calendar or matrix. Is a day intentionally nonworking, or is the working pattern wrong for this center? Those lead to different decisions. Extending the same assumptions into another month won't tell us which assumption is wrong. The day-by-day view gives the planner something concrete to compare with the intended working pattern.

[Set up shop calendars](#)

**Emma:** That's a useful distinction to make visible: 'we haven't calculated that far' is not the same as 'we don't work that day.' I like this as a small planning aid rather than a claim that scheduling has solved itself. The horizon tells the planner where to look, and the calendar view helps explain what is there.

[Set up shop calendars](#)

## Undo the posting, not the physical delivery

**Alex:** There's another distinction that saves a lot of confusion: correcting what the system records versus changing what happened outside it. If a delivery was posted incorrectly, the business needs a way to correct that record. But an undo button doesn't make the goods travel backward. That matters especially when the vendor sends directly to the customer.

[Make drop shipments](#)

**Emma:** And there is a useful improvement for exactly that process. Since the April release, Business Central supports reversing a posted drop shipment when both the sales shipment and its linked purchase receipt are still uninvoiced. A drop shipment is the vendor sending directly to the customer, with the purchase and sales documents linked. So this is not an internal transfer between the company's locations. Alex, why is the condition about both sides so important?

[Make drop shipments](#)

[Reverse drop shipments when sales and purchase documents aren't invoiced](#)

**Alex:** Because this isn't just correcting an isolated sales posting. The Undo Shipment action starts from the posted sales shipment, and Business Central also reverses the linked purchase receipt. It creates correction entries and applies the sales and purchase entries to their reversals. So the documented eligibility depends on both sides remaining uninvoiced. Looking only at whether the customer has been invoiced isn't enough. Before using this route, the person investigating the mistake needs to check the linked purchase receipt as well, not assume its status from what they can see on the sales side.

[Make drop shipments](#)

**Emma:** Let's separate two mistakes, then. In one case, the company recorded a delivery that hasn't happened. In another, the goods really reached the customer, but someone now wants to change the order. I can see why an available undo action might look tempting in both cases. But those are very different conversations with the vendor and customer.

**Alex:** They are, and the second case is why I'd establish the physical facts before choosing a correction. The feature gives us a way to reverse eligible postings. It doesn't decide whether the customer has the goods, whether they should keep them, or what the parties have agreed. In the first case, a mistaken system record may be what we need to correct. In the second, the business must resolve the real delivery and commercial situation too. Technical eligibility tells us a route is available; it doesn't tell us that route accurately represents what should happen.

[Make drop shipments](#)

**Emma:** And if one of those linked documents has already been invoiced, we should stop describing this as the same quick undo. I'd want that caught early, before someone has planned the correction around a button they can't legitimately use. What would you ask the person reporting the problem to bring you?

[Make drop shipments](#)

**Alex:** The posted sales shipment, the linked purchase receipt, their invoice status, and a plain description of what was recorded incorrectly. Alongside that, what actually happened to the goods. Those few facts keep the investigation from bouncing between sales, purchasing and finance with everyone seeing only their own document. If both sides are uninvoiced and Undo Shipment is the appropriate correction, we can use that supported route. Afterward, I'd inspect the correction entries and both linked sides. If either is invoiced, I'd assess the appropriate correction process instead of stretching this feature beyond its documented condition.

[Make drop shipments](#)

**Emma:** I'm glad you said inspect the correction entries. 'Undo' can sound as though the original posting simply disappears. For the colleague trying to explain the history later, a correction and a disappearing record are very different things.

[Make drop shipments](#)

**Alex:** They are. The documented process creates corrections and applies the entries to their reversals; it isn't a description of erasing the original history. That gives the investigation a before-and-correction relationship to examine. The improvement is that an eligible mistaken drop-shipment posting has a supported reversal route across the linked sales and purchase sides. It can remove a real obstacle to putting the records right, while still leaving us responsible for understanding why they were wrong.

[Make drop shipments](#)

## Notice the change that needs a response

**Emma:** Our last feature is about noticing a change while someone can still do something useful with the information. Business Central already has field monitoring: selected fields can be logged, and selected changes can trigger an email. The record can include the old and new values, who changed the field and when. This isn't a new October feature. It is one I think is easy to overlook if the conversation jumps straight from 'we should know about this' to 'we need a new monitoring system.'

#### [Audit changes](#)

**Alex:** And the useful word there is selected. You can decide which important fields deserve attention, and whether a change should only be logged or also notify someone. I'd start with a change that has a clear consequence for the business, not a plan to monitor everything because the option exists. A change to an important payment detail is a good hypothetical example. But before choosing email for it, I'd ask what you expect the person receiving that email to do. Is it a prompt to verify the change, or are you hoping the message itself prevents a problem?

#### [Audit changes](#)

**Emma:** A prompt to verify it. I wouldn't want every legitimate correction treated like an incident. If the payment detail was changed for a good reason, the useful outcome is that the right person can establish that quickly. But I also don't want 'an email was sent' to become the entire control. A message arriving in somebody's inbox is a very small part of getting an answer.

**Alex:** That changes how I'd frame the setup. We're creating a signal for a named review, not an automatic verdict. The old and new values, user and time can help the reviewer find the change and ask a specific question. They don't establish whether the change was authorized or whether someone acted wrongly. Field monitoring also isn't an approval step that blocks the edit. So the process needs a person who knows how to verify it, and an agreed next step if they can't. The feature makes the change visible; the business still supplies the judgment.

#### [Audit changes](#)

**Emma:** That makes the log-only choice interesting too. There are changes I might want to investigate later without interrupting someone every time. If we put all of them into the same stream of emails as the payment detail, we could make the important signal harder to notice. How would you decide which changes deserve the interruption?

#### [Audit changes](#)

**Alex:** I'd compare the consequence of waiting with the action someone can realistically take. If a change should be checked before the next important business step, a notification may be useful. If the main need is to reconstruct what changed during a later investigation, logging may be enough. Then I'd ask how often the field changes legitimately. A field edited many times during ordinary work could generate a lot of messages without producing many useful decisions. That doesn't make the data unimportant. It means the way we draw attention to it should fit the work, rather than treating every edit as equally urgent.

#### [Audit changes](#)

**Emma:** There's a setup detail worth making explicit before that first useful email: the sending account must be configured, and this feature uses Microsoft three-sixty-five or SMTP. Microsoft says the Current User account type doesn't send these notifications. So a familiar account option isn't necessarily a working choice for this particular task.

[Audit changes](#)

**Alex:** And it's exactly why I'd test the path rather than stop at the setup screen. Monitor Field Change Setup includes the fields, the user to notify and the sending account. After adding the fields and starting monitoring, sign out and back in so the settings take effect. Then, in an appropriate test setting, make a controlled change and check the result with the intended recipient. Did the log record the change? Did the notification arrive? Can that person identify what needs reviewing? Those are separate observations. A saved configuration alone doesn't answer them.

[Audit changes](#)

**Emma:** I'd add one uncomfortable question to that trial: what happens when the named person is away? Not because we need to turn a small feature into a large project, but because an important message can arrive perfectly and still wait until Monday. That would change which business step I'm comfortable relying on it for.

**Alex:** I'd include that in the operating agreement, not assume this notification feature handles it. Who covers the review, how quickly does it need attention, and what happens if the change can't be verified? Your Monday example exposes a gap that a successful email test wouldn't catch. It may mean the company needs a different process around that business step, rather than more fields in the monitoring setup. The standard feature can still be useful. We just need to be honest about the job it does: drawing attention to a change, not guaranteeing that somebody responds in time.

[Audit changes](#)

**Emma:** Let's follow the person who does receive it, then. The Monitored Fields Log Entries page gives them the old and new values, the user and the time, with links back toward the originating page. That sounds much more useful than a message saying only, 'Something changed.' But I'd still want them to ask why it changed, not infer the answer from a username.

[Audit changes](#)

**Alex:** Yes. The record helps narrow the question: this value changed to that value at this time, under this user. The reviewer can compare it with the intended change and the business context. It isn't a complete explanation of the decision that led to the edit. In our payment-detail example, a legitimate correction and an unexpected change can both produce a log entry. The value of the log is that you have something specific to investigate, rather than trying to reconstruct the previous value from memory or asking everyone whether they noticed anything unusual.

[Audit changes](#)

**Emma:** There's another detail that could save a confusing setup conversation. A table used for field monitoring isn't available in change-log setup, to avoid tracking it redundantly. So if an administrator expects to add the same table in both places, that absence isn't necessarily a fault. And changes to the monitoring setup itself create entries too, which can be shown or hidden in the view.

[Audit changes](#)

**Alex:** That gives us two different things to inspect: the business field changing, and the monitoring configuration changing. If the team is investigating why an expected signal wasn't available, the setup history may be relevant. I wouldn't assume that hiding those entries in a view means the configuration never changed. And I wouldn't try to solve every uncertainty by adding duplicate tracking. Start with what the selected monitoring actually covers, then check that its setup still matches the purpose. Otherwise the business can end up trusting a description of the control that no longer matches the configuration.

[Audit changes](#)

**Emma:** Those old and new values are useful when you need them. They're also information the company is choosing to retain. I'd rather decide how long that history is needed than leave it accumulating simply because nobody discussed the end of its life. Does setting a retention period settle that, or is there another operational step?

[Audit changes](#)

[Define retention policies](#)

**Alex:** The policy defines which old data qualifies for deletion, using the configured period and any filters. But it also has to be applied. Business Central supports scheduled processing through a job queue and manual application. The person setting up the policy needs the relevant setup permission, while the user running the scheduled work needs permission to read and delete the data. Those are different responsibilities. So I'd check both the intended retention rule and whether the processing actually runs. I wouldn't infer that old entries have been removed just because a period is filled in on a setup page.

[Define retention policies](#)

**Emma:** There's a real balance there. Keep too little and the person investigating a delayed question may lose the history they need. Keep everything indefinitely and we've stopped making a deliberate choice. I wouldn't settle that by copying a number from another company's setup. The review process we agreed earlier should help tell us what history is actually useful.

**Alex:** Yes, with the company's applicable requirements considered as well. There isn't a universally correct period we can prescribe from the feature description. What I like about your approach is that it starts with the question the history must answer. For a change that requires prompt verification, we can examine how that review works and what evidence needs to remain afterward. For a log kept mainly for later investigation, the useful window may be different. The setup should reflect those decisions. Retention isn't just housekeeping added at the end; it affects what this monitoring can tell us later.

[Audit changes](#)

[Define retention policies](#)

**Emma:** Then I'd start smaller than my first shopping list of fields. Pick a change we genuinely care about, agree who checks it, and see whether the information helps them reach an answer. If the recipient can't tell what to do next, adding twenty more alerts won't fix that.

**Alex:** And I'd count a quickly verified, legitimate change as a successful use of the feature, not a disappointing alert because nothing was wrong. The point is to make the review informed. Equally, a quiet inbox doesn't tell us on its own whether no changes happened or whether our notification path needs attention. That's why the controlled test and the log matter together. We can choose a modest scope, understand the signal, and know what observation would make us revisit the setup. That's more dependable than promising that every important event will somehow announce itself.

[Audit changes](#)

**Emma:** That's a good place to finish. Today's useful discoveries weren't all about adding more machinery. Sometimes the next step was to find an interface that already exists, or make the basis of a calculation clearer. And sometimes a small distinction changed the decision: a calendar horizon isn't spare capacity, and an available correction or notification still has a specific job and boundary. The direct sources and deeper technical notes are in the episode notes. Thanks for listening to Central Perspectives from ITrators. We'll be back with more Business Central developments worth understanding.

[Discover APIs with the API Overview page](#)

[Allocate revenue and costs to multiple general ledger accounts](#)

[Set up shop calendars](#)

[Make drop shipments](#)

[Audit changes](#)