

The integration layer is not a *separate project.*

Most mid-market planning initiatives stall at the same place: getting data out of the ERP, the HRIS and the CRM and into the planning model, reliably, every night. That work is usually bought as a second product, built by a second team, and scheduled as a second project. EACONNECT puts it inside the planning platform. This article explains what that means in practice — connectors, visual pipelines, monitoring, and the unglamorous discipline of keeping data in step when source systems change.

AUDIENCE	CONNECTORS IN THIS ARTICLE	WHERE IT LIVES	READING TIME
Leaders of mid-size enterprises	NetSuite · Xero · Sage · Acumatica · Workday · Snowflake	Integrations, inside EACONNECT Planning	11 min read

01 Why planning projects stall at integration

A planning tool is only as current as its last data load. In most mid-size companies that load is a person, a spreadsheet and a Tuesday.

The pattern is familiar. The planning tool is selected and licensed. The implementation partner builds the model. Then someone asks how the general ledger will get into it every month, and the answer is one of three things: a manual export, a connector that covers one system and not the others, or a separate integration platform with its own licence, its own skill set and its own timeline. The planning project waits on the integration project, and the business waits on both.

ROUTE 1

Manual exports

Cheap to start, expensive to run. Somebody exports, somebody loads, and the report is correct for about a day. Errors are invisible until a number looks wrong.

ROUTE 2

Vendor connectors

Most planning tools connect natively to one or two ERPs. The HRIS, the CRM and the data warehouse are "on the roadmap" or handled by route 3.

ROUTE 3

A separate iPaaS

Boomi, Workato, Celigo and their peers do the job well — for a second licence, a second console and a second team who understand the mappings. The seams are now the customer's to own.

None of these is wrong. Each is the rational choice given the products on offer. The argument of this article is that the products on offer split a single problem — *get the right data into the model on schedule and know when it fails* — into two purchases, and that a platform which keeps it whole removes most of the cost.

02 What "built in" means here

Inside EAConnect Planning there is an Integrations area with its own Build, Manage and Monitor sections. It is the same product, the same tenant, the

same login as the reports and the planning grid. The destination of every pipeline is a planning table in the same workspace.

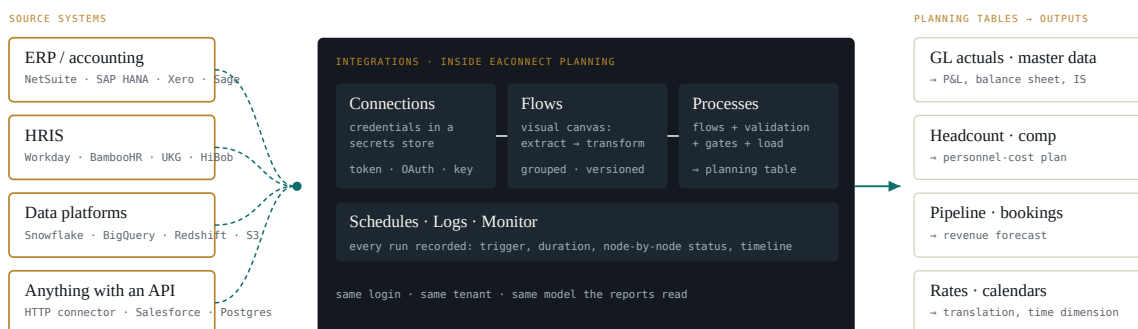


Figure 1. The integration layer sits between source systems and planning tables, inside the same product. There is no export-to-file-then-import step between "integration" and "model".

Two consequences follow that are easy to miss. First, the target of a pipeline is a typed planning table, not a file — so a load either conforms to the table's definition or fails visibly. Second, the process engine that runs data loads is the same engine that later runs budget workflows: the skill a team builds wiring up the ERP transfers directly to governing the planning cycle.

03 Connectors, and making a connection

A connection is a named set of credentials to one system. Creating one is the first and shortest step; the longest part of day one is usually on the source

system's side.

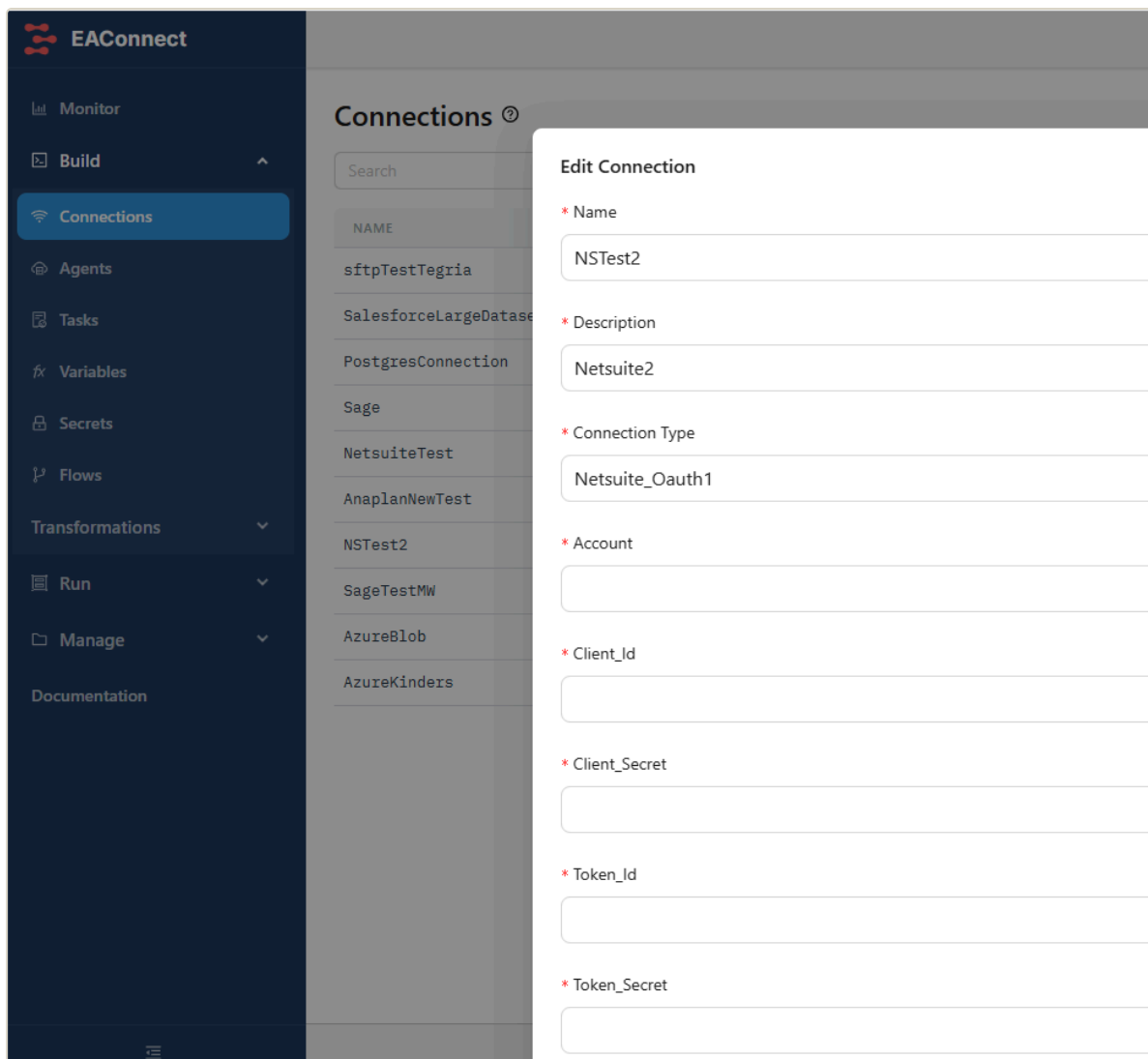


Figure 2. The Connections list — SFTP, Salesforce, Postgres, Sage, NetSuite, Anaplan, Azure Blob — with a NetSuite connection being edited. Token-based authentication: account, client ID and secret, token ID and secret. Secrets are held in the platform's secrets store, not in the flow definition.

The catalogue, at the time of writing

CATEGORY	SYSTEMS	TYPICAL USE IN PLANNING
ERP / accounting	NetSuite, SAP HANA, Xero, Sage, Acumatica, Microsoft Dynamics 365 Business Central, Restaurant365, QuickBooks Online (via HTTP connector)	General ledger actuals, chart of accounts, entities, exchange rates
HRIS	Workday, BambooHR, UKG, HiBob, Namely	Headcount roster, compensation, start and end dates for personnel-cost planning
Data platforms	Snowflake, BigQuery, Redshift and other databases; data lakes; GCS, Azure Blob, S3, SFTP	Reading from an existing warehouse rather than operational systems; bulk history
Applications and generic	Salesforce, Anaplan, Postgres; a generic HTTP connector for any REST API	Pipeline for revenue forecasting; migration from a prior planning tool; anything else
The list grows; treat it as a snapshot. In practice the HTTP connector covers most systems that expose an API, at the cost of the mapping being yours to define.		

Where day one actually goes

Creating the connection in EACConnect takes minutes. Obtaining the credentials takes as long as your ERP administrator needs to create an integration record and issue a token — for NetSuite that is a defined procedure with role permissions attached. Plan for it, assign it, and do not measure the integration platform by a delay that belongs to the source system.

04 Visual pipelines: flows and processes

A pipeline is a directed graph of steps — extract this, transform that, validate, load — where each step runs when the steps before it succeed. EAConnect exposes that graph on a canvas at two levels: flows, which move data, and processes, which orchestrate flows with checks and branching.

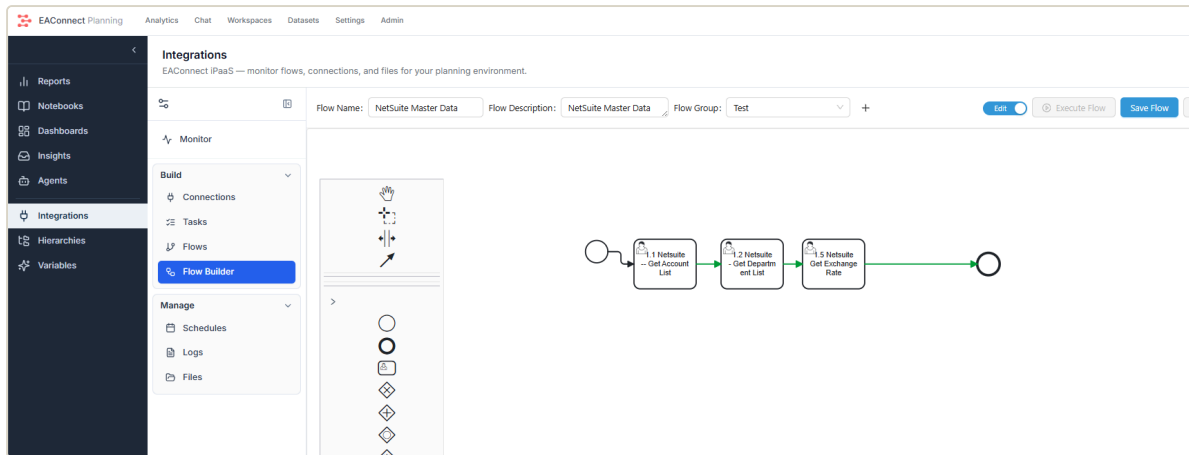


Figure 3. Flow Builder. A NetSuite master-data flow of three connector tasks in sequence — account list, department list, exchange rates. Flows carry a name, description and group, and can be executed from the canvas while being built.

Each node in a flow is a connector task with its own configuration: which object, which fields, which filters, which target. Flows are grouped and versioned, and — the detail that most shortens the build — they can be run on demand from the same screen, so the loop of "run it, look at the rows, adjust the filter" takes seconds rather than a deployment.

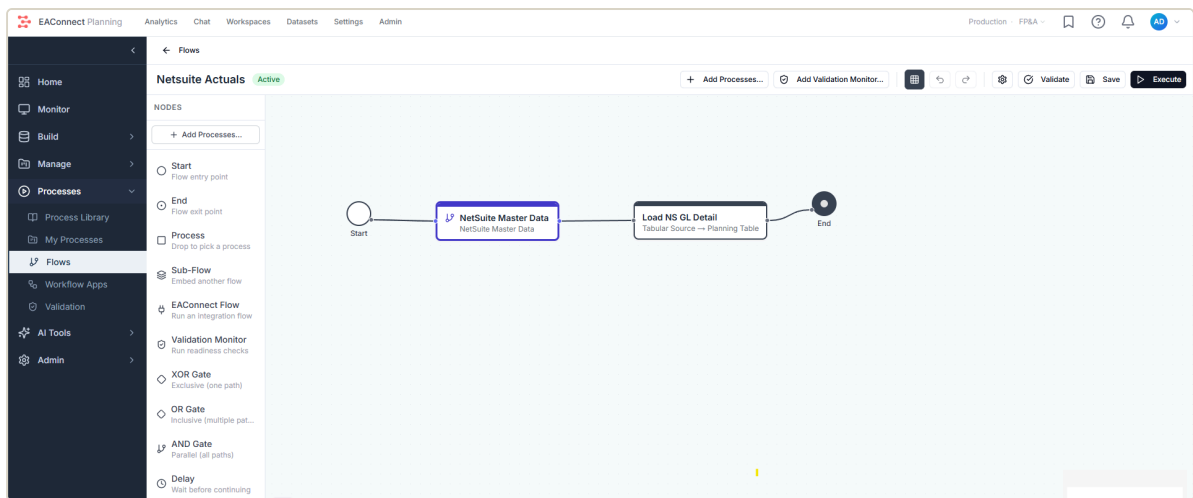


Figure 4. A process. It runs the master-data flow, then a load step that takes the tabular source and writes it to a planning table. The palette on the left shows what else a process can contain.

ANATOMY OF A PROCESS GRAPH

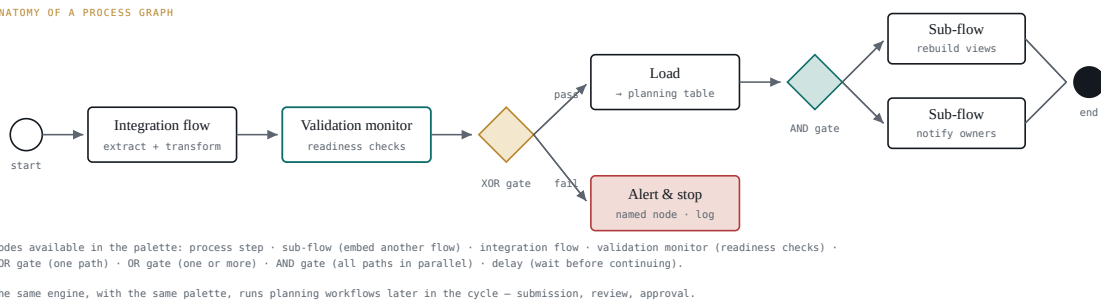


Figure 5. What a process graph can express. The validation-gate-alert pattern in the middle is the one that matters for reliability: a load that would corrupt a plan is stopped before it is written, at a named node.

The word "DAG" — directed acyclic graph — is engineering vocabulary for a simple idea: steps with dependencies and no loops, so the engine always knows what can run next and what must wait. What it buys a planning team is legibility. A month-end load is no longer a script somebody wrote; it is a picture the finance manager can read, with each box's status shown against it after every run.

05 Running it: schedules, monitoring, failure

The test of an integration layer is not the first successful run; it is the two-

hundredth, and what happens on the morning one of them fails.

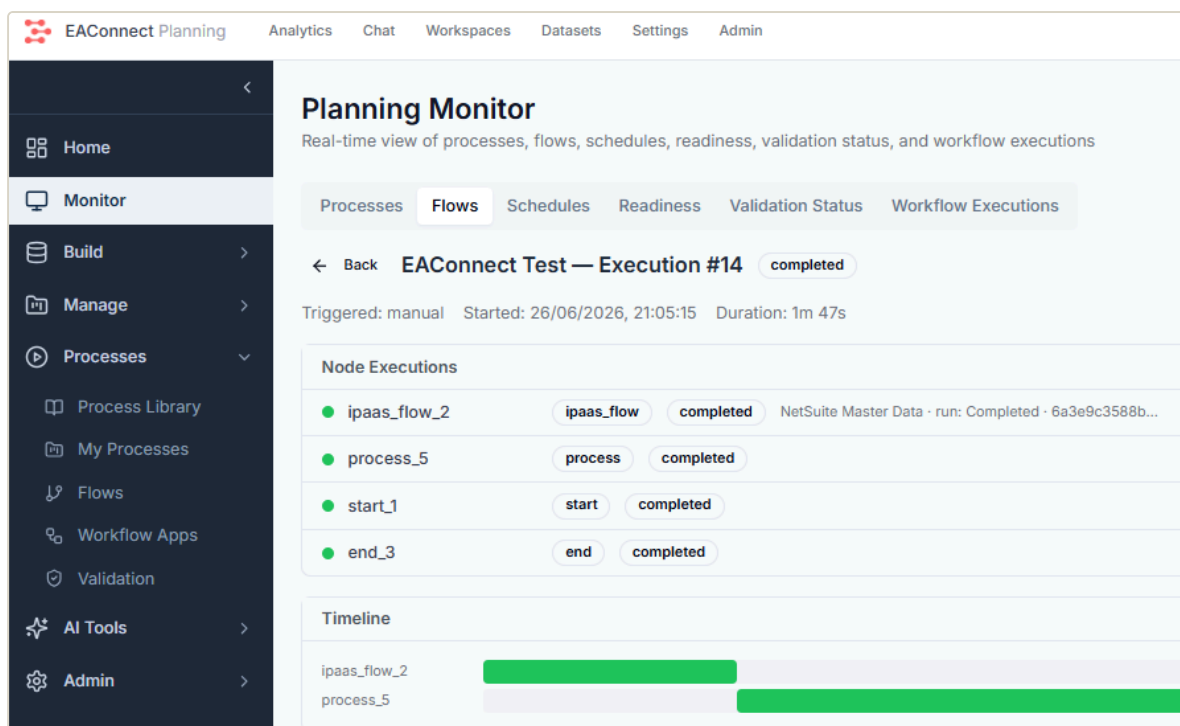


Figure 6. The Planning Monitor. Execution #14 was triggered manually, ran 1 minute 47 seconds, and completed. Each node — the integration flow, the load process, start and end — shows its own status; the timeline shows where the time went.

What a run leaves behind

- **A record** — trigger type, start time, duration, and completion state, per execution.
- **Node-level status** — when something fails it fails at a named step, with a log, rather than as a silent gap in a report three weeks later.
- **A timeline** — which step consumed the time, so the slow source (a rate-limited API, a large history pull) is obvious.

Scheduling and retries, as practice

Nightly is the normal cadence for general-ledger actuals; hourly is possible for operational data where the source API permits. The schedule is a setting on the process. For retries the practitioner's advice is the same on any platform: retry transient failures (a timed-out call, a rate limit) a small number of times with a delay, and do not retry structural failures (a missing field, a rejected credential) — those need a person. The validation monitor and the alert branch in Figure 5 are how a process encodes that distinction.

Honest time expectations for the NetSuite actuals pipeline

Source-side setup: minutes to a day, depending on your ERP administrator. Connection in EAConnect: minutes. Master data plus GL detail flow, including test runs: an afternoon for someone who knows their ERP. Process with validation and a nightly schedule: the same day. Historical backfill: governed by data volume and the source API's rate limits — plan for multi-year history to run overnight.

06 Keeping data in step: change capture and schema drift

Two problems appear only after the first month, and they are where most home-grown integrations quietly rot. This section explains them as practice; the platform mechanisms that address them are noted where they have been shown.

KEEPING A TABLE IN STEP WITH ITS SOURCE

Full reload

truncate and re-pull everything
simplest · correct by construction
cost grows with history · fine for master data

Incremental by watermark

pull rows changed since the last run
(a modified-date or sequence) · cheap
misses deletes unless the source flags them

Change data capture / replication

keep a synchronized copy of the source table:
inserts, updates and deletes · most faithful
needs a source that exposes change

Rule of thumb: full reload for small reference tables;
incremental or replication for transactional history.

FOUR KINDS OF SCHEMA DRIFT · AND THE RIGHT RESPONSE

A column is added at the source

usually harmless · detect, log, ignore until mapped · never fail the load for it

A column the load depends on is renamed or removed

a hard failure · stop before writing · alert a named owner · the validation gate's job

A type or format changes (text to number, date format)

the dangerous one · loads may "succeed" with wrong values
type checks at the typed table boundary catch it

A code list changes (new department, merged entity, new account)

not a schema change, but the most frequent · versioned hierarchies absorb it
unmapped members are quarantined, not dropped

What EAConnect provides for this, as shown: typed planning tables as the load target;
validation monitors that run readiness checks before a process proceeds;
a data-replication capability under Settings; versioned hierarchies for code-list change.

Figure 7. The two after-month-one problems. Neither is solved by a connector; both are solved by discipline at the boundary between the pipeline and the planning table — and by making failures loud.

Change capture: how much to move, how often

The naive approach re-pulls everything every night. It is correct and, for master data and small ledgers, entirely adequate. It stops being adequate when the general-ledger detail runs to millions of rows and the source API meters requests. The alternatives are to pull only what changed since the last run — which depends on the source exposing a reliable "modified" marker and on deletes being flagged — or to maintain a replicated copy of the source table that receives every insert, update and delete. EAConnect offers a data-

replication capability alongside its flows; for most mid-market general ledgers the practitioner's recommendation is incremental loads for detail with a periodic full reconciliation, and full reloads for everything small.

Schema drift: when the source changes under you

Source systems change. A NetSuite administrator adds a custom field; a Xero tenant renames a tracking category; an HRIS upgrade changes a date format. Most integrations discover this when a report is wrong. The right design has three properties: detect the change at the boundary (the target table is typed, so a type change is a rejected load, not a corrupted plan); distinguish the harmless from the fatal (an added column is logged, a removed one stops the process); and route the fatal case to a person, at a named node, before anything is written. In EAConnect the validation monitor is the node that carries these checks, and the alert branch of a process is where the failure lands. The most frequent form of drift — a new department, a merged entity — is not a schema change at all, and is handled by the versioned hierarchies described in the platform's hierarchy article rather than by the pipeline.

What we are not claiming

This section describes the practice and the mechanisms EAConnect has shown: typed tables, validation monitors, replication and versioned hierarchies. Automatic repair of a drifted mapping — inferring that a renamed column is the same column — is a design choice on which reasonable platforms differ, and we describe it here as a design question rather than a feature.

07 Where integration sits in the four weeks

EAConnect's implementation claim is a single ladder: first value in hours, AI-assisted builds in days, a governed go-live in four weeks. Integration is the

third week of that ladder, and the reason the fourth week is a board pack rather than a data-cleaning exercise.

HOURS → DAYS → FOUR WEEKS → ONE LADDER

WEEK 1 · HOURS TO FIRST VALUE

Files in, statements out

chart-of-accounts audit
GL export · budget file loaded
P&L, dashboard, planning grid

→ prove the outcome before
touching an integration

WEEK 2 · DAYS PER BUILD

Model and drivers

hierarchies · calendars · currency
driver formulas · entry forms
AI-assisted from plain English

→ the business decisions that
no platform makes for you

WEEK 3 · THIS ARTICLE

Live source systems

connections · flows · processes
validation · nightly schedule
historical backfill overnight

→ the report from week 1 now
refreshes itself

WEEK 4 · GOVERNED GO-LIVE

Workflow and board pack

submission · review · approval
broadcasting · audit log
formatted pack to the board

→ the cycle runs on the
platform, not on email

Implementation services are time-and-materials; a typical baseline engagement is in the low five figures and is spent on weeks 2 and 4 – decisions and governance – not on week 3.

Figure 8. The four-week ladder. Integration is deliberately the third week: by then the model exists and the team has seen the outputs on files, so mapping decisions are made against something real.

The ordering is the point. Teams that start with integration spend the first month arguing about field mappings for a model that does not yet exist. Teams that start with files spend an afternoon proving what the model produces, and then wire in the sources with a clear picture of what each table is for.

08 A practitioner's view for leaders

Opinions from people who have built integrations on Boomi, MuleSoft and SnapLogic and inside planning platforms. Not product claims.

Count the seams, not the connectors

Every vendor has a connector list. The question that predicts cost is how many products the data crosses between the ERP and the board pack. Each crossing is a mapping to maintain, a failure to diagnose across two consoles, and a reconciliation between two copies of the truth.

Own the credentials and the schedule

Whoever holds the ERP integration credentials and can see the run log owns the integration. Keep that inside finance systems, not with an external partner, or month-end will depend on someone else's ticket queue.

Make failures loud and specific

The cheapest integration failure is the one that stops at a named node with a log at 02:14. The most expensive is the one that loads wrong numbers successfully. Validation gates at the table boundary are worth more than any retry policy.

Decide the mappings, then automate them

Which subsidiaries, which accounts, which posting periods count as closed, how post-close adjustments are treated: these are finance decisions. An integration platform makes them cheap to change; it should not be expected to make them.

Start on files, integrate third

Prove the model on exports in week one. Wire the sources in week three, once you know what every table is for. Teams that reverse this order spend a month mapping fields into a model that does not exist yet.

Do not buy a second platform to feed the first

A separate iPaaS is the right choice for a company running dozens of application-to-application integrations. For feeding a planning model from three to six sources, it is a second licence, a second console and a second team for a problem the planning platform should already solve.

EACConnect Planning · Research library · Hours, Not Months

Evidence: EACConnect Integrations screens — Connections, Flow Builder, Processes, Planning Monitor (Sept 2026). Connector catalogue as stated by EACConnect at time of writing. Change capture and schema drift described as practice.

2,422 words · 11 min read