

Describe the application. Review the design. *Apply it.*

Configuring a planning application has always meant clicking through a dozen admin screens, one object at a time, with nobody able to say afterward exactly what was built or whether it still matches the design. EAConnect's Solution Builder replaces that with a single design document — generated from CSVs, a requirements file, a mockup, a template or a conversation — that is validated, diffed, approved, applied in dependency order, and verified against the live application. This article explains how it works, what "verified" actually means, and why the honest speed claim is days per build, not minutes.

AUDIENCE	THE ARTEFACT	OBJECT TYPES	READING TIME
Leaders of mid-size enterprises	MODEL-SPEC · one row per object	21 · dimensions to suites	10 min read

Series Hours, Not Months

Evidence Solution Builder and Grounded Authoring user guides; configuration-agent domain documentation, code-reviewed Jun 2026

Standard shipped behaviour as documented; the blueprint engine is noted as a specified design

01 The gap between the design and the thing

Every planning implementation has a design document and a built application, and after week three nobody can say how they differ.

The failure is structural, not human. Design lives in a slide deck or a workbook; the application lives in configuration screens; the connection between them is a consultant's memory. When a report is tweaked in production, the design is stale. When the design says "headcount by role and location" and the model was built by role only, nobody finds out until the first headcount plan will not load. And when a new business unit wants "the same thing", the answer is another six weeks of clicking, because there is nothing to copy except screenshots.

The AI-assisted approach in EAConnect starts from a different premise: the design document *is* the build. One structured document lists every object the application needs; the assistant proposes it from what you give it; you decide what to reuse and what is open; the platform applies it through the same governed services a builder would use; and from then on every change is a diff against the document, and every drift is visible.

THEN

Object by object

Twelve admin screens, one at a time, in an order the builder holds in their head. Reproducible only by the person who did it.

NOW

By spec, by conversation

One document, generated and refined conversationally, applied in dependency order. Reproducible by anyone with the document.

ALWAYS

Under the same controls

The assistant never bypasses access control, environment guards or locks, and never edits platform code. What it cannot express becomes a recorded gap, not a workaround.

02 The artefact: a spec with one row per object

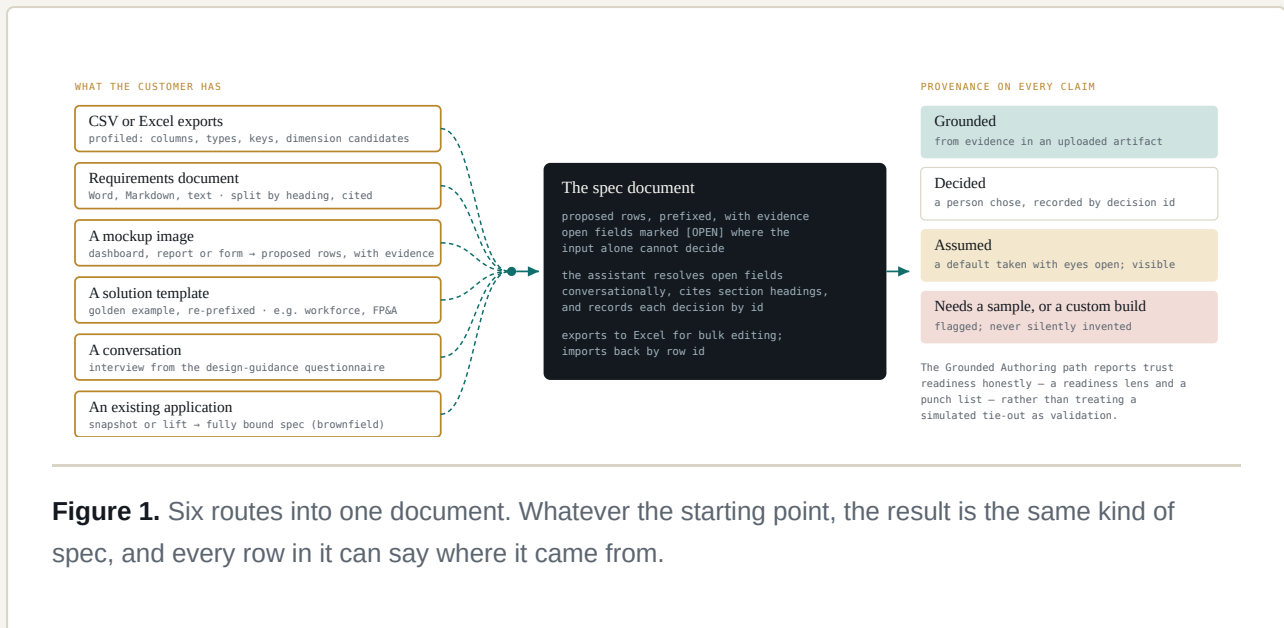
The MODEL-SPEC is a tabbed document — think of a workbook, because it round-trips to Excel — where every dimension, hierarchy, model, measure, view, form, report, process and suite is one row with a business code, a payload, a binding and an apply status.

TAB GROUP	OBJECT TYPES	WHAT THE ROWS CARRY
Structure	dimension · attribute · members · hierarchy · scenario · version	Codes, member trees, effective-dated hierarchy versions; reuse bindings to environment-shared dimensions
Model	planning_table · model · measure · variable · slice_lock	Grain, measures and formulas, central rates, lock policies
Surfaces	view · form_page · report · dashboard · suite · home_pin · folder	Entry grids over views, statement layouts, dashboard widgets bound to datasources, the assembled application
Operations	process · flow · workflow_app	Data loads, integration flows, submit-review-approve applications
Prose	narrative · requirements · design decisions · conventions · mockups · data loads	The business understanding, the source documents, the seventy-nine-question decision log, the attached files
Twenty-one object types in the current version. Every reference between rows is by code — <code>dimension_code</code> , <code>model_code</code> , <code>parent_code</code> — never by database id, which is what makes a spec portable across sandbox, test and production and every write idempotent.		

Three columns on each row matter to a leader. **Binding** says whether the row creates something new, reuses an existing object, extends one, or replaces one — and replace is the only mode that needs a confirmation. **Apply status** records the live id and a fingerprint of what was written, so the platform can later tell whether the live object still matches. **Realizability** says whether the platform can build the row as specified, or whether it is a capability gap.

03 Six ways in

The document can start from whatever the customer actually has. Most mid-size companies have spreadsheets, a few screenshots and a conversation; all three are valid inputs.



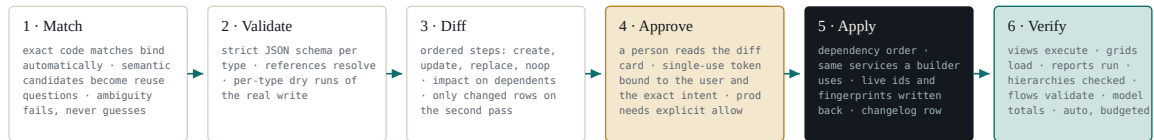
Two of the routes deserve a sentence each. The *interview* is not free-form: when a spec has no rows, data or documents yet, the assistant asks the opening questions from the platform's design-guidance questionnaire — scenario, entities and currency, time grain, horizon and fiscal-year start, data sources, scenarios and versions — and records the answers by decision id, so the same nine blocking questions from the implementation playbook are answered in the same place. The *mockup* route sends a screenshot to a vision-capable model together with the datasources the application already has, and asks for a dashboard, report or form definition bound to real views — with the visual elements each row came from cited as evidence.

04 The loop: match, validate, diff, approve, apply, verify

Nothing goes live without a diff and an approval. That sentence is the whole governance model, and it holds whether the change is a new application or

one added measure.

FROM DOCUMENT TO LIVE OBJECTS · EVERY TIME



iterate on the same document: edit rows → diff shows only what changed → apply again

Binding modes on every row: create (must not exist; an identical object is a no-op) · reuse (must exist, unchanged) · extend (default: create or additive update) · replace (destructive; needs a token). Removal always needs a token and refuses while dependents exist. Production environments reject structural writes unless explicitly allowed with a confirmation.

The assistant proposes; the platform validates; a person approves; the services apply; the platform verifies. The model is never the last step.

Figure 2. The loop. Steps 1–3 and 6 are automatic and read-only; step 4 is a person; step 5 runs through the same governed services as manual configuration. Nothing in the sequence is unique to AI — which is the point.

05 What "verified" means here

Vendors say "AI-generated, verified". It should mean something specific. In EAConnect it means four checks, in order, each recorded.

BEFORE APPLY

Schema and reference validation

Every payload is checked against a strict schema — the same schema the write tools enforce, so what the assistant reads is exactly what it must write. Every code reference must resolve; an ambiguous one (the same member code under two dimensions) fails rather than guesses.

BEFORE APPLY

Dry runs of the real write

Each object type's write is executed in dry-run mode against the live environment, so "would this work" is answered by the service that will do it, not by a simulation.

AFTER APPLY

Automatic verification

Views are executed, form grids loaded, reports run, dashboard widgets queried, hierarchies checked for orphans and cycles, flows validated, loaded models totalled — within a budget of 25 checks or 90 seconds per apply, with one outcome per check and failures written to the changelog. A larger run can be requested later from the workbench.

AFTERWARDS, FOREVER

Drift detection

Apply writes a fingerprint per row. Drift compares fingerprints with live payloads; a row someone edited in an admin screen shows as drifted, and the resolution is explicit — adopt the live change into the spec, or restore the spec to live.

What it deliberately does not claim

The Grounded Authoring guide's own words: honest readiness, "not simulated tie-out". A spec that applies and verifies is a correctly built application. Whether the business logic it encodes is the right logic is a decision recorded against a person's name in the design-decisions tab — and the article on driver modelling explains why that part cannot be automated away.

06 When the platform cannot: capability gaps

The most important design decision in the assistant is what happens when a requirement cannot be expressed. The answer is not a workaround; it is a record.

Every row carries a realizability status. When the assistant determines that a requirement — a calculation the measure language cannot express, a form layout the builder does not support, an integration with no connector — cannot be met by configuration, it files a capability gap: a structured record with the requirement, the evidence it came from, the row it blocks, and an exportable spec stub for the development team. The rest of the application applies; the gap is visible in the workbench, in the trust readiness lens, and to whoever runs the implementation. This is the behaviour that turns "the platform cannot do X" from a silent spreadsheet on the side into a product decision with a paper trail.

GAP TYPE	EXAMPLE	WHAT HAPPENS
Expression	An allocation rule the formula language cannot express	Gap filed with the intended formula; the measure row is marked unrealizable; the rest of the model applies
Surface	A form layout from a mockup with no equivalent grid pattern	Nearest pattern proposed and marked assumed; the difference is the gap
Integration	A source system with no connector and no API	A file-based load is proposed; the connector is the gap
Process	An approval chain with a branch the workflow engine does not support	The supported chain is built; the branch is the gap, with the requirement text cited
Gaps are queryable objects with a lifecycle (report, list, export, resolve). A leadership team can see, per application, exactly where the platform stopped and why.		

07 Nine things it is used for

The Solution Builder guide documents nine end-to-end scenarios. They are

the practical shape of the feature.

SCENARIO	STARTING POINT	OUTCOME
Adopt an existing application	A live, hand-built app	Snapshot into a fully bound spec; every later change is a row edit with a diff
New solution from CSVs	Customer exports	Dimensions, tables, model, entry views and a first report proposed with evidence, then applied
Enhance a live solution by chat	"Add a variance-to-prior-version measure and a report on it"	Rows added, diff shown, approved, applied, verified
Detect and resolve drift	Someone edited a report in production	Row shows drifted; adopt or restore, explicitly
Record a capability gap	A requirement the platform cannot meet	Structured gap with evidence and a spec stub for engineering
Promote between environments	A certified sandbox spec	Applied to test or production by code; secrets never travel
Start from a template plus interview	Nothing yet	Golden example re-prefixed, opening questions answered, decisions recorded
Dashboard from a mockup	A screenshot	Widgets bound to existing datasources, with visual evidence cited
Bulk-edit in Excel	A spec with two hundred rows to adjust	Export, edit, import by row id; unchanged rows keep their apply state
One documented template — a complete FY2026 workforce plan for a fictional company, with eight CSVs, dimensions, roster and assumption tables, a model, views, seed flows, forms, a cockpit, reports, a dashboard and a suite — is instantiated in an empty environment as an end-to-end test, with model totals checked against a manifest.		

08 The honest speed claim

"Six months to two weeks" is the kind of line that gets written about this feature. Here is the version that survives contact with a real implementation.

Hours: a first applied spec

From CSVs and a conversation to a proposed spec, a review, and an apply in the sandbox. The workforce template does this as an automated test; a real customer's data adds an afternoon of open-field resolution.

Days: a reviewed application

Owners review forms on their own numbers, decisions get recorded, the diff-approve-apply loop runs three or four times. This is week two of the four-week playbook, and it is where the platform saves the most time: the keystrokes are gone, the decisions remain.

Four weeks: governed go-live

Because integration, reconciliation, UAT and sign-off are people and calendar, not configuration. The assistant makes the build a small fraction of the four weeks; it does not make the four weeks disappear.

The blueprint engine described in the platform's design documents — a generator that takes a completed questionnaire and emits a full, dependency-ordered application blueprint routed to each builder — is a specified design at the time of writing, with a worked FP&A fixture. What ships today is the Solution Builder loop above; this article describes that, and notes the engine as direction rather than product.

09 A practitioner's view for leaders

Opinions from people who have configured planning applications by hand and

reviewed ones built by assistants. Not product claims.

Buy the document, not the demo

The generated application is impressive for ten minutes. The spec that describes it — diffable, promotable, drift-checked — is what you will still be using in year three. Evaluate the artefact, not the animation.

Insist that "verified" be defined

Schema, dry run, post-apply execution, drift. If a vendor cannot name the checks and show their results per object, "verified" means "it did not throw an error".

The decisions are yours; keep them where you can find them

An assistant that records seventy-nine design decisions by id, with who answered, is worth more than one that answers them for you. The logic of the business is a liability if nobody can say who chose it.

Gaps are good news

A platform that files a structured capability gap when it cannot meet a requirement is telling you the truth. One that produces something anyway is producing a workaround you will maintain forever.

Brownfield first, if you have one

Snapshotting an existing application into a spec is the fastest way to find out how much undocumented configuration you have been running on. Do it before the next change, not after.

Keep the human at step four

Match, validate and diff can be automatic. Apply should follow a person reading a diff card and spending a token. That is a few seconds per change and it is the difference between an assistant and an autopilot.

EACConnect Planning · Research library · Hours, Not Months

Evidence: Solution Builder user guide (v1) and Grounded Authoring guide; configuration-agent domain documentation, code-reviewed Jun 2026; design-guidance questionnaire. The blueprint engine is a specified design and is described as such.

2,368 words · 10 min read