

Find. Manage. Trust.

Paper 3 of 4 — can you believe what the record says?

What do we know about the pump tagged P-101?

It is a fair question and it sounds like a simple one.

Somewhere there is an answer to every part of it. What it is. What it was specified to be. Which drawings show it. What the vendor actually supplied. Whether anybody has changed it since.

That information exists. It is in a P&ID somebody drew, a model somebody built, a plot plan, a purchase order, a vendor submittal, and a spreadsheet that may or may not still be current. Getting to it means knowing where each of those lives, and who to ask.

And then there is the question underneath, which nobody asks out loud until something goes wrong:

How much of it can be relied on?

PlantViZN begins from the asset instead.

Find

Start anywhere. A tag, a document, an item picked off a drawing.

From P-101, the documents that show it: the P&ID, the three-dimensional model, the plot plan, the vendor submittal. Each authored somewhere different, by somebody different, in an application the person asking may not own — and they never have to know that.

But a pump is not an island, and the question almost never stops at the pump.

What is upstream of it. What it feeds. Which valves isolate it. What else belongs to the same system, and what those things are in turn.

And the piping is only half of it. A pump has a control story as well — what permits it to start, what trips it, what protects it at low flow, which instruments participate in its startup and its shutdown. That story lives in instruments and logic rather than in pipe, and it is usually the part that decides whether somebody can safely touch anything.

Then the question that follows every one of those answers: **where is it in the plant?** Not which document mentions the instrument — where it physically sits, so somebody can locate it in a model or a scan and then go and stand in front of the real thing.

An engineer asking about P-101 is within about ten seconds of asking about something connected to P-101 — physically, functionally, or both — and then about where that thing is. The moment any of those steps means opening a different application and starting again, the trail goes cold.

And it is not one person asking. The engineer planning a modification. The constructor working out how to reach it and what has to come out first. The operator who needs to know what isolates it before a shutdown,

and what trips it during one. The inspector who has to find it, confirm it is what the record says it is, and write down what they saw.

And the reviewer — whose job is precisely to ask whether a proposed change makes sense in the system it belongs to, and who has never really been able to. Reviewing on paper means checking whether a drawing is correct on its own terms, because checking it against the P&ID means finding the P&ID, and checking what else is on that line means tracing it across a dozen sheets. So the review stops at the edge of the drawing, not because anyone is cutting corners, but because the context was never reachable in an afternoon.

Five roles, five different weeks, five different reasons — asking versions of the same question about the same pump. And each of them, today, solves it the same way: ask somebody who knows, or walk out and look.

What the record actually holds

Find P-101 and it is the same pump everywhere it shows up — in the P&ID, in the model, in the vendor data — even though none of those systems were built to talk to each other. They are matched by what they describe rather than by what each one happens to call it, and what comes back is its discipline, the models it appears in, and what each source says about it.

Where it sits on a line, that line carries its own facts: size, specification, the equipment it runs from and to.

That connection is what turns the record into something you can walk. From the pump to the line it belongs to. From the line to the equipment at either end. From a specification to every line drawn on it — and so to the question nobody could previously answer without a week and a wall: **if this changes, what else does it touch?**

And nobody has to go and open the file to get the answer. The details that matter — spec, material, rating — are pulled out of the source documents and attached to the asset itself.

The challenge was never that the information did not exist. It is that reaching it depended on knowing where somebody else chose to put it.

Find the asset. Find its context.

Manage

Find gets you to the evidence. It does not tell you what the asset *is*.

That is a decision somebody has to make, and it is what Data Manager is for. The scattered evidence — a value from the model, a rating from the P&ID, a figure from the vendor submittal — gets consolidated into one authoritative statement of the asset.

That statement is the datasheet.

Which matters more than it sounds, because a datasheet is not a document PlantViZN found. It is a document PlantViZN produced, from sources that never agreed to talk to each other. And once it exists it does two jobs at once: it is the answer to *what is this*, and it is the definition of what a complete answer looks like for anything else of its class.

And most projects do not start with a model. They start with a process flow diagram and an equipment list in Excel, built before anyone opens a CAD tool. PlantViZN takes that list in from day one, matches its columns to the right properties, and keeps it as the asset's source of truth. When the P&ID and the model catch up, they add to that record. They do not replace it.

Alongside it, the ordinary working verbs. Properties maintained and corrected as better information arrives — and because a correction to the record is an engineering act rather than a typing one, changes can be routed for approval rather than simply taking effect. Views onto a set of assets, built for a project or across an account, so each discipline sees what it cares about. Datasheets, reports and extracts produced from the registry rather than assembled by hand.

Sources keep their own structure and their own authority. Nothing is absorbed. What changes is that the asset now has a considered statement about itself, rather than six partial ones scattered across systems that have never been compared.

Existing information stays where it belongs. The asset gets an answer.

Trust

Now there is something to check against — and this is the part worth saying plainly, because it is the argument for having done any of the preceding work.

Assembling the set was the hard part. Getting information out of systems that never agreed to share it. Giving things identities that survive leaving the tool that made them. Attaching the documents. Establishing the relationships. That is the work of years, and it is the part that is genuinely difficult.

Once the set exists, applying rules to it is comparatively easy.

Which is the whole point. The difficulty was never writing a rule — anybody can write a rule. It was having a population coherent enough for a rule to mean anything. **Data quality is what a well-assembled set makes possible**, not a feature bolted on beside it.

And writing one need not be a specialist act either: in the Validation Hub, someone can describe a rule for spec compliance in plain language and get a draft back to check.

An engineering record can carry missing values, incomplete properties, inconsistent data, conflicting representations, documents that should exist and do not, and relationships that no longer agree. None of it announces itself. It sits quietly until somebody makes a decision on it.

So the registry does not only collect and consolidate. It measures — and it measures against the datasheet Manage produced.

Coverage — does the asset have what its datasheet says it should have?

Consistency — do the sources agree with each other?

Conformity — does it meet the specification it is supposed to meet?

No new vocabulary. Three questions any engineer already asks, asked automatically and at scale.

Consistency is the one that finds things nobody was looking for. A pump that appears in the P&ID and in the model is described twice, by two people, working from the same intent at different times — and on a real project it is described more often than that, in whichever authoring tools each discipline happened to use. Every platform holding data on the asset is compared against every other one that does. Where those descriptions disagree, the disagreement surfaces — and a device appearing in a different order in one of them is not a formatting discrepancy. It may be an engineering inconsistency that has been sitting there for years.

Some conflicts resolve automatically, where mapped properties give a reliable basis for reconciliation. Others need engineering judgement — and those become **issues**.

An issue is the action half of data quality: an owner, a status, a root cause, and somewhere to record what was decided. Until it closes it holds the asset's score down rather than quietly disappearing into an inbox. And a conflict that somebody chooses to ignore earns no credit — dismissing a finding is not the same as resolving it, and the score knows the difference.

But the part that matters is where issues are consumed. They are composed here, against the assembled record, by whoever is accountable for its quality. They are actioned at the desktop, in the drawing, by the designer who can actually fix the thing — **who never has to open this at all.** The work comes to them, in the tool they already use, and the number of people who need an account here stays independent of the number of people doing the work.

Validate → Detect → Resolve → Track → Score

An objective basis for trust

Information quality does not have to remain a matter of opinion.

Each asset is scored on the three lenses against a versioned template — which is, in effect, the datasheet specification for its class. The score carries a per-criterion breakdown, so it says *what* is wrong rather than only *how much*. Unresolved issues hold it down. As problems are cleared the picture improves, and because scores are captured over time the direction of travel is visible as well as the position.

Scores roll up to the level the question is being asked at: an asset, an area, a discipline, the whole project.

And they are measured against where the project actually is. A record early in design is not held to the standard of one at handover. Each phase of a project carries its own expected Coverage, Consistency and Conformity — so the number answers *are we where we should be by now*, not *are we finished*, which is the only version of the question that is useful while work is still going on.

At the end of that sequence sits a **handover gate**: a threshold the project is measured against before the record changes hands, with the exceptions listed rather than averaged away.

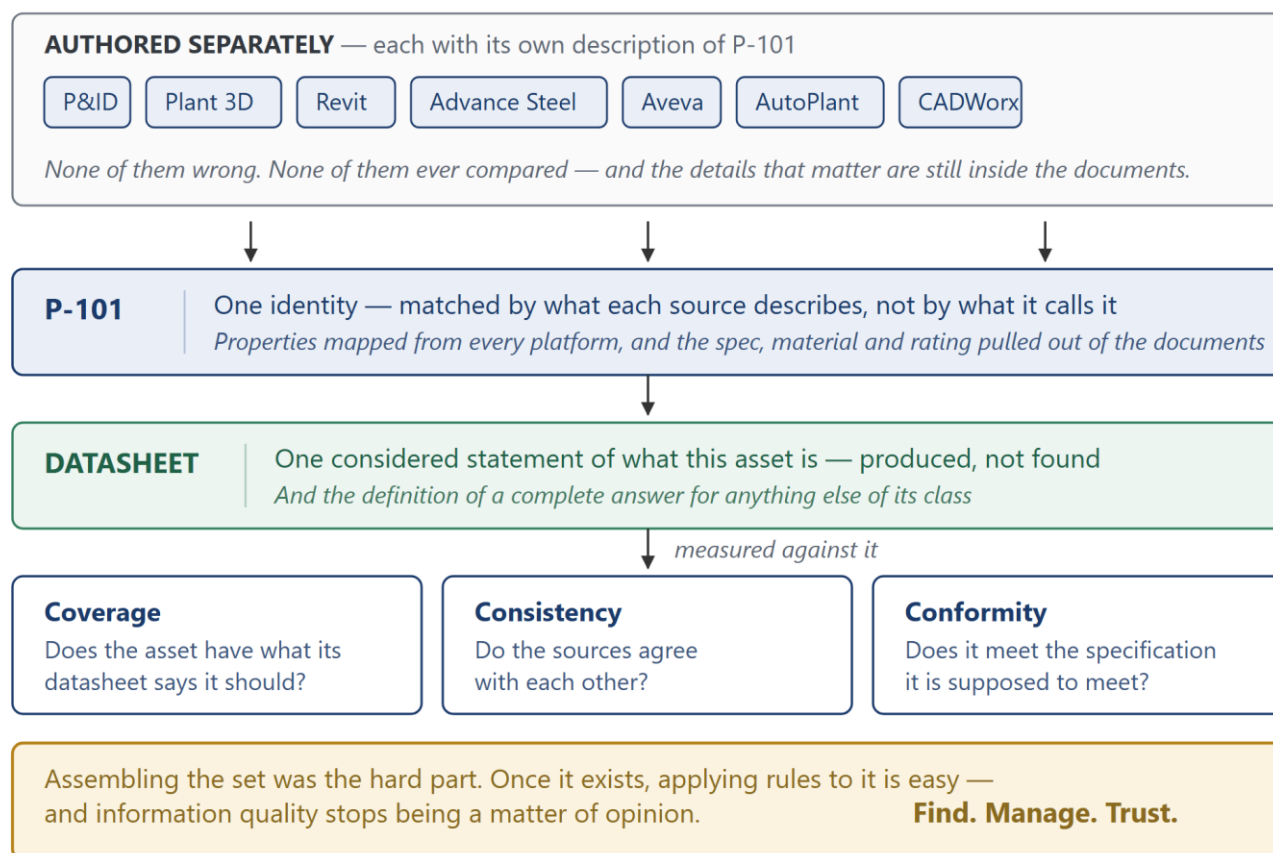
Which turns this:

“We think the information is pretty good.”

into a number that can be examined, argued with, and improved. That is a different conversation to have at handover.

One asset, many representations

Nothing is absorbed. The asset gets an answer — and the answer can be measured.



One asset, many representations

What changed, and who changed it

A score says what the record looks like now. It does not say how it got that way.

So each asset also carries its own history: what changed, when, who changed it, and what the value was before. Not a log of everything that happened to the project — a timeline belonging to the asset, so the question *when did this become what it is* has an answer that does not depend on anybody remembering.

This is a small thing that turns out to matter enormously, and it is the thread the next paper picks up.

The registry that can be deleted

There is one more property, and for an owner deciding whether to depend on any of this, it is the one that matters most.

The registry can be destroyed and rebuilt from the files.

Un-elevate a project and it returns to plain ACC — no registry, no repository, nothing but documents in a folder the customer already owns. Elevate it and the registry is built again from those documents, including the datasheets and reports PlantViZN produced itself.

Every other asset registry has to be *trusted*, because it is the only copy. This one does not have to be, because it is derived. Remove it entirely and the customer holds their own Autodesk platform and their own documents, exactly as if they had never adopted anything.

Which is worth saying plainly, because it inverts how this category usually works: **the data was never hostage**. Not as a promise. As a property of how it is built.

Where this is reached from

This is a connected environment. The aggregate — the registry across an estate, the validation, the scores, the datasheets — is reached by logging in to the service. That is the normal shape for a registry, and it is a different surface from the drawing.

Most of the traffic runs upward. What AssetLink establishes at the workstation travels up into this, and the assembled view is built from it.

Issues run the other way. They are composed against the aggregate and delivered to the desktop, so the people who do the work receive it without coming here. That distinction matters for anyone counting seats: the accountable few work in the registry, and everybody else keeps working where they already are.

What This Is Not

It is not the authoritative source for anything. Source systems remain authoritative for what belongs to them.

It is not a replacement for an owner's operational or maintenance systems.

It does not require every source to adopt one object model, or one schema that anticipates every future class.

And it is not a document management system. The documents stay where they are. What changes is that the asset knows about them.

The Larger Idea

Contractors change. Authoring applications change. Disciplines use different tools, and the systems around an asset keep evolving.

The asset remains.

Information becomes useful when people can find what they need. It becomes more valuable when somebody has done the work of turning scattered evidence into a considered answer. And it becomes trustworthy when that answer can be measured, its inconsistencies surfaced, and its progress tracked.

Find. Manage. Trust.

But trust has a shelf life, and this is where it begins to leak.

Everything above describes a record at rest — assembled, consolidated, checked, scored. A facility is not at rest. It is modified constantly, in small increments, by people under time pressure, and every one of those changes is a chance for something verified last year to quietly become wrong.

The pump gets a minimum flow bypass at two in the morning.

Will the record still be true afterwards — and will anyone know why it was done?

That is where the last paper begins.
