



ALGOSHRED TECHNOLOGIES

SYSTEMS THAT TALK · AGREE ONCE HOW YOUR SYSTEMS SHARE DETAILS

Get your systems talking, so your people can stop retyping.

We help your work apps agree one way to ask each other for details, and one shared board where they announce news, so an order or an address can be typed once and reach the apps that need it, written in a way each one can use.

Worth knowing: We connect one link at a time, starting with the ones that hurt most. It's steady work, not a switch you flip.

Say it once. Everyone who needs it can hear it.

Email contact@algoshred.com with the apps one order passes through and where people retype.

Integration & interoperability engineering as a service

Vignesh T.V., CEO & CTO, Algoshred Technologies Pvt. Ltd.

contact@algoshred.com



The problem, and the idea.

We type the same order or address into more than one screen.

The website says it's in stock; the stockroom says otherwise.

An order went missing while one system was down, and nobody noticed for a while.

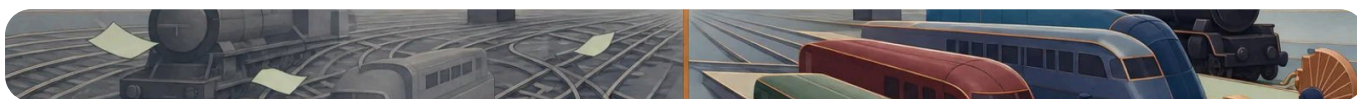
A customer was charged twice, or a parcel went to an old address.

Adding one new tool means rewiring others, and something else breaks.

A partner wants to connect, and all we have is emails and spreadsheets.

Integration means getting your separate work apps to pass details to each other by themselves, in a way each one understands, so people don't have to copy the same thing from screen to screen.

The story, decoded. In our short film, four trains are your work apps: shop, stockroom, billing and delivery. The fifth, the partner's line, is another company's system. Auntie Saule's parcel is an order. Mr. Feathers is anyone copying details between screens. The junction is all your apps together, and each line is one link between them. The plug is one agreed way for apps to ask each other for things (an API), the board is where news is announced once (events), and the shelf is where news waits until an app is ready (a message queue).



01 The same kind of plug for each app

Each app gets the same kind of plug: one agreed, written way to ask it for things, saying how to ask and what it answers. The shop can ask the stockroom what's left and get a reply. A new app gets a plug of its own, so the others aren't rewired to fit it. In the film, "the same plug" means the same kind of plug: each app gets its own.

Experts call this: APIs and API contracts

Worth knowing What each plug offers is agreed between the apps that use it, and helps only if both sides keep to it, so we add checks that test each side against the agreement. A new app still needs setting up and testing on its side, and the apps that want to use it still need setting up to ask it.

02 Say it once, on one shared board

When something happens, like a new order, it is announced once. Every app that needs it reads the board and gets on with its part; the others carry on.

Experts call this: events, publish and subscribe

Worth knowing Apps that hear the news catch up shortly after, not all at the same moment. Announcements make it easy to add listeners and harder to follow one order from start to end, so we add a way to trace each one. A shared board is one more thing to look after. It needs an owner and a plan for when it is busy or down.

03 News waits, repeats get skipped

If an app is busy, the news waits on its shelf until it's ready. If the same news turns up twice, the app checks whether it already acted and skips the repeat. Stuck items go to a lost-and-found shelf, not the bin.

Experts call this: message queues, dead-letter queues, safe repeats (idempotency)

Worth knowing A waiting shelf holds news until an app is ready and keeps stuck items on a lost-and-found shelf instead of dropping them. It can sometimes hand over the same news twice, so each app is set up to spot repeats, and someone still watches the lost-and-found shelf.

04 Each keeps its own app, and they understand each other

Every team runs its own app its own way, and so can a partner company. They share the board, and details are written so both sides read them the same way.

Experts call this: federation, interoperability, data mapping

Worth knowing A converter changes how details are written, not the facts. If details are wrong at the start, they arrive wrong, so we check them at the source too. Each team keeps its own system. A shared view (one screen where teams see the same order) shows only what each side agrees to share, and someone has to keep that agreement current.



One plug and one board, or a knot?

EVERY APP WIRED TO EVERY OTHER

Wired: every new app needs a track to every other.

Returns joins. It needs its own track to every other app, and each one wants its own form.

ONE AGREED PLUG EACH, ONE SHARED BOARD

Agreed: every new app needs one plug of the same kind.

Returns joins. It gets the same kind of plug and reads the one board. The others aren't rewired to fit it.

Who hears the news?

WHAT HAPPENED

A new order

BY HAND

Mr. Feathers copies the order onto a form for the stockroom, another for billing and another for delivery.

ON THE BOARD

ORDER IN goes up once. The stockroom, billing and delivery read it and get on with their part.

Payment taken

Billing rings the shop, then sends a note to delivery, and hopes both got it.

ORDER PAID goes up once. The shop and delivery read it; the stockroom carries on.

Parcel on its way

Delivery fills in a form for the shop and one for the stockroom, and the address is copied again.

PARCEL SENT goes up once. The shop tells the customer, the stockroom ticks the box off, and Returns notes the parcel in case it comes back.

A BUSY DAY

Delivery crew on a tea break

BY HAND

Nobody at Delivery. The form is left on a bench and blows away.

ON THE BOARD

The news waits on Delivery's shelf until the crew is back.

The board rings twice

Mr. Feathers copies it twice. Two parcels.

Delivery checks: already sent? Then it skips the repeat.

A new app wants the news: Returns

One more form on Mr. Feathers' round.

Returns starts reading the board. The shop isn't changed.

Arrived, read, understood Did it arrive? · Can they read it? · Does it mean the same thing?

WHICH DETAIL

Delivery address

CONVERTER AT THE BORDER: OFF

✓✓✗ Their system reads the first line as the town, so the van is booked to a town called Lake House.

CONVERTER AT THE BORDER: ON

✓✓✓ The converter puts the lines in their order. The van heads for the right town.

Customer name

✓✓✗ Their system puts the family name first, so the birthday card greets Auntie by the wrong name.

✓✓✓ The converter puts the names in their order. The card greets Auntie Saule.

Parcel weight

✓✓✗ We wrote kilos; they read pounds, and book a van that's too small.

✓✓✓ The converter writes the weight in their units. The right van turns up.

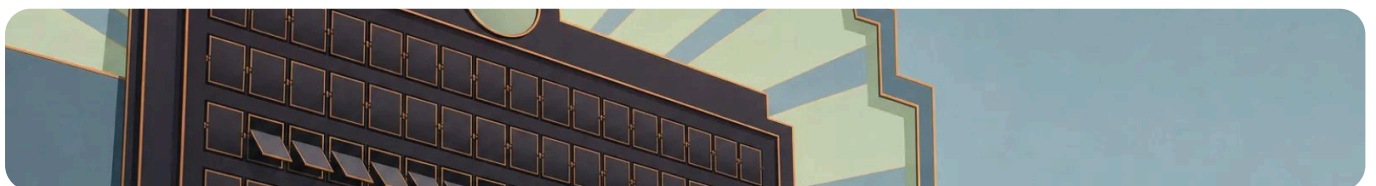
An address typed wrong at the start

✓✓✓ All three checks pass, and it is still wrong: Blue Cake arrives as Blue Cake. The mistake was made before it left.

✓✓✓ All three checks pass, and it is still wrong: Blue Cake arrives as Blue Cake. A converter changes how details are written, not the facts, so we check them at the source.

Worth knowing

A converter changes how details are written, not the facts. If details are wrong at the start, they arrive wrong, so we check them at the source too.





What we help with.

Start with the link that hurts most, then join the rest one at a time.



Trace one order through your apps

We follow one real journey, such as an order, across every app it touches, and draw every link, hand-off and place where someone retypes.

A clear picture of how your apps really pass details, and which links to fix first.

Experts call this: integration review, system landscape, API and event inventory

One shared board for news

Moments that matter, like “order in” or “parcel sent”, announced once, with a written list of who sends what and who reads it (an event catalogue).

New apps can start listening without anyone changing the sender.

Experts call this: event-driven architecture, publish/subscribe messaging, event catalogue

Details both sides read the same way

One agreed way to write common things, such as customers, orders and dates, and converters between each app’s habits and your industry’s standards. A converter changes how details are written, not the facts. If details are wrong at the start, they arrive wrong, so we check them at the source too.

Fewer mix-ups when details cross from one app, or company, to another.

Experts call this: data mapping, shared data model, standards such as HL7 FHIR, ISO 20022 and EDI

A proper door for partners and suppliers

Partner-facing plugs, agreed file exchanges, and a message to the partner’s system when something happens, so a new partner joins along an agreed path instead of a custom one.

Adding a partner follows a known path.

Experts call this: partner APIs, B2B and EDI integration, webhooks

The same kind of plug for each app

Agreed, written ways to ask each app for things. Numbered versions let an old way keep working while apps move to the new one. One front door checks every request before it reaches the app, old apps included.

Changing one app is less likely to break the others.

Experts call this: API design, API contracts, gateways, versioning

Shelves that keep news safe

News that waits when an app is away, gentle retries, a lost-and-found shelf, repeats that are spotted and skipped, and a label to follow each journey. Someone still watches the lost-and-found shelf.

Lost or doubled news is easier to spot and trace.

Experts call this: message queues, retries, dead-letter queues, idempotency, tracing

Each team runs its own app

Teams keep their own apps and plugs, with shared rules at the junction, one screen where teams see the same order, and one sign-in where it helps. Shared sign-in needs both sides to agree whom they trust and to keep that list up to date, and each system still checks what that person is allowed to do.

Teams wait less on one central team to make changes.

Experts call this: federation, API and GraphQL federation, single sign-on

Room for AI helpers on the same tracks

Giving AI assistants limited, logged plugs into your apps, through the same entrance, permissions and checks as everything else, including the openly published way many AI assistants use to connect to apps.

Helpers get only what they are given, and their requests go through the same logged entrance.

Experts call this: agent-ready APIs, tool calling, Model Context Protocol

For your technical team: we work with what you already use, for example REST and OpenAPI, GraphQL and Apollo Federation, gRPC, AsyncAPI and CloudEvents, Apache Kafka, RabbitMQ, NATS, cloud event and queue services, API gateways such as Kong, OpenTelemetry, and standards such as HL7 FHIR, ISO 20022 and EDI.

Capabilities, from tracing one order to a junction your team runs.

Agreed plugs and one shared board, ways of writing details that both sides read the same way, shelves that keep news safe, and room for partners and AI helpers on the same tracks.

Trace and agree

See how details really travel today, then agree how they should.

- A map of your apps, the links between them and who owns each
- A list of every plug and announcement, including forgotten ones
- The right pattern for each link: ask and answer, announce, wait on a shelf, or an end-of-day batch
- Written guides for each plug and announcement

Write it the same way

Details that mean the same thing on both sides.

- One agreed way of writing common details, such as customers, orders and dates
- Converters between each app's habits
- Adapters for industry standards (health records, payments, partner paperwork). Some industries have rules or standards for sharing records, such as health records and payment messages; where yours does, we work to that standard and check it with your compliance team.
- Versions, and a retirement plan for old plugs

Connect and announce

Agreed plugs in front of your apps, and one board for news.

- Plugs in front of core and old apps. Many old systems can be given a proper connection without replacing them; some need changes first, and we say which after we look.
- A shared board with a written list of announcements
- Partner and supplier connections
- One screen where teams see the same order, and one sign-in across teams where it helps

Keep it safe and traceable

Shelves, repeat checks and a label to follow each journey.

- Waiting shelves, gentle retries and a watched lost-and-found shelf
- Repeats spotted and skipped (safe repeats, idempotency)
- A label to follow each journey end to end, with alerts on stuck items
- Access rules, limits and logs for people, partners and AI helpers

WHERE IT BECOMES REAL · ILLUSTRATIVE EXAMPLES

Retail and e-commerce

For example, a new order could be announced once and picked up by the stockroom, invoicing and the customer email, instead of being typed into each.

Financial services

For example, payment messages could be written in the agreed industry format, with repeats spotted so a retry does not pay twice.

Healthcare

For example, a referral sent from a clinic could reach the hospital in a form its system reads, with dates, names and doses meaning the same on both sides.

Logistics

For example, each new carrier could join through one agreed partner door instead of emailed spreadsheets.

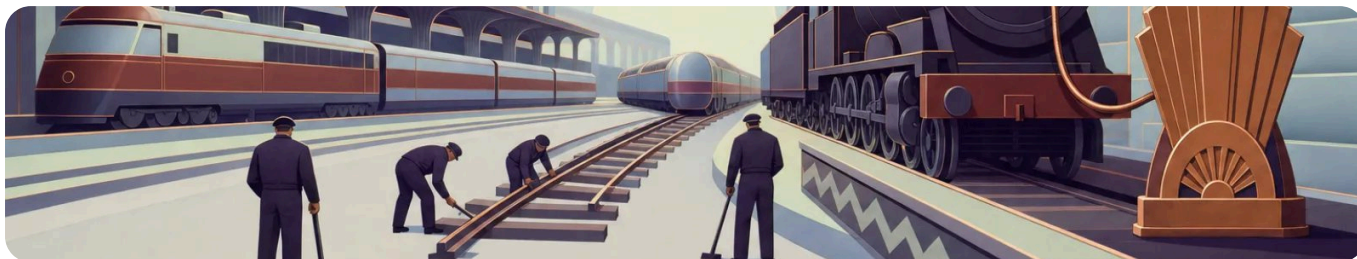
Software companies

For example, a public plug with versions could let customers move to a new version in their own time.

Public sector

For example, departments could share a record through agreed plugs while each runs its own system.

Trace, agree, connect: one line at a time.



01 LISTEN

Listen to the people who retype

We talk to the people who retype, chase and fix mix-ups, and pick one journey that hurts, such as an order from shop to doorstep.

02 TRACE

Trace the journey

We draw every app, link and hand-off on that journey, and mark where details are copied, lost or doubled.

03 AGREE

Agree the plugs and the board

We agree the plugs, the announcements and the shared way of writing details for that journey with you, and who owns each.

04 CONNECT

Connect one line, then the next

We build that journey end to end with one team, with tracing and the shelves in from the start, then move the next links over one at a time while the old ones keep running.

05 YOUR TEAM RUNS IT

Your team runs the junction

Your people own the list of plugs and announcements, the agreements and the check-ups, with templates and training. We can stay on to help with the next line if you prefer.

Ways to start

Trace one order. We follow one real order through every app it touches and draw every hand-off, including the places people retype.

First agreed plug. One busy link redesigned as an agreed plug, built with one team.

Fix one failed hand-off. We follow one failed hand-off end to end, then add a waiting shelf and a watched lost-and-found shelf to that link.

What you get

- A map of the traced journey, with every hand-off and retype marked
- The agreed plugs and announcements for that journey, written down
- A written, shared way of writing common details
- The connected journey, with a label to trace each order
- A lost-and-found shelf with a named owner
- Templates and a check-up routine your team can run



Questions, myths and the next step.

What is integration, in plain words?

Getting your separate work apps to pass details to each other by themselves, in a way each one understands, so people can stop copying them from screen to screen.

What is the difference between integration and interoperability?

Integration is connecting the systems. Interoperability is them understanding each other once connected: the details arrive, can be read, and mean the same thing on both sides.

What is an API, and what does “event-driven” mean?

An API is an app’s plug: every app gets the same kind, one agreed, written way to ask it for things, and what it answers. Event-driven means an app announces “this happened” once, and every app that cares reacts, instead of all of them asking again and again.

Do we have to replace our old systems?

Often not. We can put an agreed plug in front of many of them so they keep working while the rest connect, and move pieces over one at a time if and when that makes sense. Many old systems can be given a proper connection without replacing them; some need changes first, and we say which after we look.

Do we need to buy a big integration platform first?

No. We start from the tools you already have and the connections that hurt most. A new platform comes in only where it helps.

Won’t one central hub become a bottleneck?

It can, and the shared board is a central piece too. The difference is that the board only carries the news: each team keeps its own app and its own decisions, so changes don’t wait on one central team. The board still needs an owner and a plan for when it is busy or down.

What happens when news gets lost or sent twice?

News waits when an app is away, failed hand-offs are tried again gently, items that can’t be delivered go to a shelf someone watches, repeats are acted on once, and each journey carries a label you can follow. These catch many problems, not all of them.

How will we know it’s working?

We agree a few plain measures with you at the start, for example how many hand-offs are still typed in by hand, how often news lands on the lost-and-found shelf, and how long a change in one app takes to reach the others. You keep the numbers and see how they move, so you can judge for yourself whether it’s helping; we don’t promise targets for them.

MYTHS, ANSWERED

“We have to replace our old systems first.”

Often not: many old apps can get the same kind of plug and keep running, like the creaky stockroom train in our story. Many old systems can be given a proper connection without replacing them; some need changes first, and we say which after we look.

“One big central system will fix it.”

It can become the new bottleneck: one central team everyone waits on. A shared board is a central piece too, but it only carries the news; each team keeps its own app and its own decisions. The board still needs an owner and a plan for when it is busy or down.

“If the details arrive, the systems understand each other.”

Arriving, being read and meaning the same thing are three different checks. A converter changes how details are written, not the facts. If details are wrong at the start, they arrive wrong, so we check them at the source too.

“Once it’s connected, it’s done.”

Apps, partners and rules change, so plugs need an owner, versions and a regular check-up. We connect one link at a time, starting with the ones that hurt most. It’s steady work, not a switch you flip.

Still copying by hand?

Email contact@algoshred.com with the apps one order passes through and where people retype.

contact@algoshred.com

WATCH THE FILM

algoshred.com/integration-interoperability