

ANTON by openEXPERT

The ANTON family whitepapers · Autumn 2026 · Paper 10 of 11

The .anton Format

Why expertise should travel as a package, not a paragraph — and what a package that cannot run anything looks like

> *A package that says what it is, proves it has not changed, and cannot run anything. Then good work gets reused instead of rebuilt.*

Version 1.0 · 17 December 2026

Created by Daniel Bardun · FutureChain AB

Open source ANTON is EUPL · github.com/altspace-hub/anton_desktop

See it live ai-portal.futurechain.eu (showcases with fictional data)

Before we start

Where we left off

Last week we looked at rhythm. Checks, validations and monitoring have been yearly because they were expensive, and when they cost hours rather than weeks they can run whenever something changes — a new document, a new release, a new regulation. We argued that this is not the same yearly cycle done faster. It is a different way of working, with people spending their time on what moved rather than on starting from zero.

That paper ended with a question, and this one tries to answer it. If the expertise in a compliance function now lives in modules, skills, knowledge packs, rules and records of work — the things we have been describing all autumn — **how do you share it, check it and keep it?**

It sounds technical. It is not. It is one of the oldest questions in professional work: how does what one person has learned become something the whole team can use? In the language of the first paper, this is the shareability gap — not one of the six, but one of the four further gaps our first whitepaper named beside them.

The recipe problem

In our first whitepaper, back in February, we listed shareability as one of the gaps that keep AI from delivering in professional work. We put it like this: sharing an AI set-up today *"is like trying to share a recipe by describing the taste instead of listing the ingredients."*

I still think that is exactly right. Somebody in your team gets an AI tool to do a recurring task really well. They have found the right method, the right checklist, the right sources, the right settings, the right way of asking. And then they try to pass it on — usually by pasting a long prompt into an email — and the colleague who receives it gets something noticeably worse. Not because the colleague is less skilled, but because half of what made it work was never in the email.

There is a second problem hiding behind the first, and it is the more dangerous one. If expertise travels as loose text, then anything can travel with it. A "free template" from the internet is just a long paragraph of instructions, and nobody reads every line of a long paragraph before using it.

This paper is about the answer we have built: one openly documented file format, **.anton**, for packaging expertise so that it can travel, be checked before anything is installed, and be trusted for the right reasons.

In this paper

- **Part 1, the problem:** three scenes, why expertise gets trapped, and why loose text is a risk as well as a nuisance.
- **Part 2, our thesis:** why expertise should travel as a package, and what we are *not* claiming about the format.
- **Part 3, our take:** six principles, what is inside an .anton file, the checks on import, and what a signature does and does not prove.
- **Part 4, a worked example:** a team's AMLR gap method, shared with a new colleague.
- **Part 5:** what this means for you on Monday, where sharing could go — built and planned, clearly marked — and next week's final paper.

Part 1 · The problem

Expertise gets trapped

Every compliance function I have worked with has the same quiet asset: a handful of people who know *how*. Not just what the regulation says, but how the team turns it into a deliverable a supervisor accepts — which questions to ask first, which evidence carries weight, how the board likes its summary, what the last inspection picked up on.

With AI, that asset takes a new form. The person who knows how is now also the person who has worked out how to get the AI to do it properly. Their set-up is the method, the checklist, the persona the model should take on, the sources it should rely on, the layout of the output and a dozen small settings that took weeks to get right.

And it is trapped. It lives in that person's workspace, in a document on their desktop, in their head. **The organisation has invested in one person's breakthrough, and the investment does not compound.** When she is on holiday, the team falls back to the old way. When she leaves, the set-up leaves with her.

Three ways expertise travels badly

These are composites. No names, no clients.

Scene 1: the prompt in the email

A senior consultant sends a colleague *"the prompt I use for AMLR gap analyses"*. It is a good prompt — long, careful, clearly the product of experience. The colleague pastes it in and runs it on a similar case.

The results are worse. Not dramatically, but enough that she spends an afternoon fixing them. The difference, it turns out, was everything the email did not carry: the original also relied on a regulatory knowledge pack, a persona — *"write as an internal auditor would"* — and an output format with the team's red, amber and green columns. None of that was in the paragraph. The consultant did not even think to mention it, because to him it was simply "how it is set up".

Scene 2: the new joiner

A new analyst joins the second line. She is capable and experienced, and it still takes her two months to learn how this team writes a review report — what goes in the summary, how findings are worded, which severity scale applies, what the head of the function will send back.

The team's method exists. But it exists as habits and old documents, and the only way to learn it is to absorb it slowly from colleagues who are too busy to explain it. Every team has this cost. Most never measure it.

Scene 3: the poisoned template

A team finds a *"free compliance template"* for an AI tool online. It promises a ready-made policy review. Somebody imports it and starts using it.

Buried in its instructions, a long way down, is a sentence addressed not to the reader but to the model: an instruction to disregard what it was told earlier, and to include summaries of the documents it sees in a message to an outside address. Nobody read the whole file before using it. Why would they? It looked like a template.

> **Loose text travels badly in both directions. The good set-up loses its ingredients. The bad one smuggles in its own.**

Why loose text fails

The three scenes fail for the same reason: a paragraph of text cannot tell you what you need to know before you rely on it.

It cannot tell you what it depends on. A prompt sits on top of knowledge, a persona, a format and settings. Copy the prompt alone and you have copied the top of the iceberg.

It cannot tell you who made it, when, or whether it has changed. Which version of the regulation did it assume? Is this the latest version? If a colleague edits one line before forwarding it, nobody downstream will know — and a method that can drift unnoticed is a method you cannot defend.

It cannot protect you from itself. Text addressed to an AI model is, in effect, an instruction. The industry calls a hidden one *prompt injection*: an instruction slipped into material the model reads. A paragraph has no way to warn you that it contains one.

Why it matters now

All autumn we have argued that the value in professional AI lives in the set-up, not in the model. If that is true, the set-up is now one of the most valuable things a compliance function owns: the written-down form of how the team works.

Valuable things need a container. We do not keep contracts as loose sentences in emails, or distribute software as unlabelled text. And *"we used the prompt someone sent round in March"* is not an answer to an auditor. A named, versioned method with a known author is.

Part 2 · Our thesis

Expertise should travel as a package, not as a paragraph.

The thesis

A package says exactly what it contains, who made it, which version it is, what it was checked against and what it leaves out. It is checked before anything is installed. And nothing in it can run.

When expertise travels that way, two things change. **Good work gets reused instead of rebuilt** — the colleague in Scene 1 gets the whole set-up, not the top of it. And **trust can be judged instead of assumed** — the team in Scene 3 sees what the template contains, and is stopped at the one sentence that should not be there.

Why we believe it

1. It is how professional knowledge already travels. Documents travel as Word and PDF files, with a structure that any program can open. Software travels as packages with versions, authors and checksums. AI expertise has become valuable enough to deserve the same discipline — and, as Scene 3 shows, risky enough to need it.

2. Shareability is a gap of its own. It is the gap between one person's breakthrough and the team's way of working — and no improvement to the model closes it, because the model was never what got lost in the email.

3. We built it, and we use it every day. This is not a proposal. In ANTON, regulatory knowledge packs, Risk Atlas packs, expert modules, skills and records of finished runs already travel as .anton files. The importer refuses programs and scripts, checks structure and checksums, and stops a module whose texts contain hidden instructions until a person decides.

4. Openness lets a universe grow. The format is documented in ANTON's public repository, so other tools can read and write it without asking our permission. That matters. A format only one program can open is a filing cabinet. A format others can read is the beginning of a shared library — and, in time, of something larger, which I come to in Part 5, clearly marked as a plan.

The recipe card, not the taste

Think about how a good kitchen hands on a dish. Not by describing the taste, and not by letting the new cook watch once. It writes a recipe card: the ingredients and their amounts, the method in order, the oven temperature, who created it and when it was last changed. A new cook can follow it on their first day. An experienced one can see at a glance what is different from the last version. And the head chef can check that nobody has quietly swapped an ingredient.

An .anton file is a recipe card for professional expertise. It lists the ingredients — the instructions, the input form, the defaults, the skills and personas the method relies on, the knowledge it draws on. It names the author, the version and the dates. And it can carry a seal that shows whether anything has been changed since it was written.

What it cannot do is cook the meal for you, and that is deliberate. **There is nothing in the box that runs.**

What we are not claiming

The limits matter here, so let me be precise.

We are **not** claiming that .anton is an open standard. In our first whitepaper we called it *"an open interchange standard"*. That was too grand. A standard is something many parties agree on, usually through a standards body. .anton is a format one project documents openly, in a public repository, for anyone to read and use. Useful, but not the same thing, and I would rather say so.

We are **not** claiming that other vendors support it. Today, ANTON reads and writes it. Some of the family's other experts do as well.

We are **not** claiming that a signature proves the content is right — the most misunderstood point about signed files, which I come back to below.

And we are **not** claiming that a marketplace for expertise exists. It does not. It is planned, and we mark it as planned.

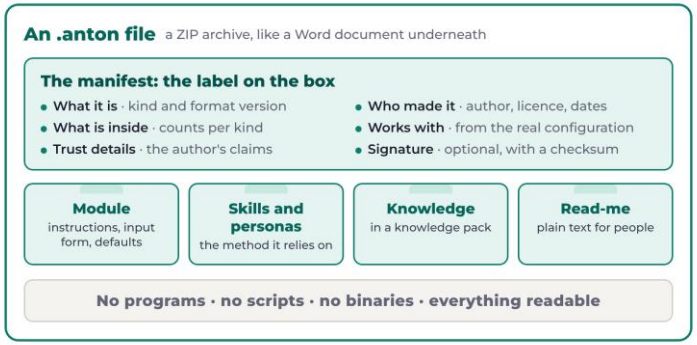
Part 3 · Our take: a package that cannot run anything

Six principles

1. **One package, one purpose.** Each file is one kind of thing — a module, a knowledge pack, a record of a run — and says so on its label.
2. **Everything readable.** Settings and texts only. No programs, no scripts, no binaries. You can open any file and read it before you import it.
3. **The label tells the truth.** The manifest says what is inside, who made it and which version it is. Where something cannot be checked, it is shown as a claim, not as a fact.
4. **Checked before anything is installed.** Structure, checksums, signature and hidden instructions are checked first. Nothing is applied until the checks pass — or until a person has decided.
5. **A person decides on anything flagged.** The importer does not quietly clean up or quietly reject a suspicious sentence. It stops, shows it, and records what the person chose.
6. **Say exactly what a seal proves.** A signature shows the file has not changed. It does not show the content is correct.

Principle five is the one I want to dwell on, because it would have been easy to do differently. We could have built an importer that silently strips anything that looks like a hidden instruction. It would feel safer. But it would also mean that a legitimate rule — *"ignore the client's formatting and use ours"* is a perfectly sensible house rule — could be removed without anyone knowing, and the method would quietly change. **A check that edits your content without telling you is just another black box.** So the importer stops and asks. It is the rule we have repeated all autumn, applied to sharing: the machine proposes, a person decides.

What is inside an .anton file



A package that says what it is, and cannot run anything.

An .anton file is a ZIP archive — the same kind of container a Word document is underneath. Inside are two things: a **manifest**, and the content.

The manifest is the label on the box. It says, for every kind of bundle, the same things:

> **The label on the box: what every manifest says**
>
> - **What it is:** the kind of bundle, and the version of the format it was written in.
> - **Who made it:** the author, the organisation, the licence, and when it was created and last updated.
> - **What is inside:** how many of each kind of content it holds.
> - **Which AI providers it works with:** taken from the real configuration, never invented.
> - **Optional trust details:** an effective date, a source link, who validated it, and whether a person has confirmed the content.
> - **An optional digital signature** over the manifest, and a checksum of the contents.

The content is plain: settings written as structured text, and instructions and knowledge written as ordinary formatted text. **No programs, no scripts, no binaries.** Everything that will reach an AI model is something a person can open and read first.

What it can carry

On 11 October 2026 the format defined more than 40 kinds of bundle. For a compliance reader, they fall into a handful of groups:

Group	What it carries	An example
Expert work	Modules — instructions, input form, defaults, and the skills and personas they use; skills; personas; workflows	The team's tuned AMLR gap analysis module
Professional standards	Compliance rule sets, quality baselines, review panels, audience profiles, brand templates	How your board likes a risk summary
Knowledge	Regulatory knowledge packs; Risk Atlas industry and financial-crime packs	An AMLR pack with its effective date
Records and evidence	A record of a finished run; signed evidence packs; Risk Atlas exports	What was asked last quarter, with which settings

Group	What it carries	An example
Projects and people	Project templates, starter packs, career profiles, contact bundles	A starter pack for a new joiner

Other kinds serve parts of ANTON beyond compliance — lessons, coding blueprints, portals, humanitarian deployment kits. They make one point: **the same container works for very different kinds of expertise**, because it only cares that the contents are labelled, readable and checkable.

Two of these deserve a closer look for a compliance team.

Knowledge packs. The regulatory knowledge packs that ship with ANTON — more than 40 of them, from the AMLR and the EBA guidelines to DORA, MiCA, GDPR and the AI Act — are .anton files, and so are the 33 Risk Atlas packs. That means each one carries its own label: what it covers, its effective date, its source, and whether a person has confirmed it. A knowledge pack that was drafted with AI and not yet checked by a person says so, and ANTON warns you when you import it.

Deliverable blueprints. We have written thirteen blueprints for financial crime deliverables — the gap report, the BWRA report, the model validation report, governing documents and more — together with house standards, as skills. Each can be written out as its own module bundle, and all of them together as one. The tests import every one of them through the normal importer, the same one you would use, without accepting a single finding from the hidden-instruction check. This part was merged into ANTON on 10 October 2026.

There are honest gaps, too. Some kinds of work, such as missions and project templates, cannot yet be exported as files; ALEXANDER does not read or write .anton; and there is no bundle kind for AI agents today.

Checked before anything is installed

The import is where trust is earned or lost, so this is where most of the care has gone. Before anything in an .anton file is applied, it passes a series of checks, in order.

1. Nothing runs. If the file contains a program or a script, it is refused. There is no setting to allow it.

2. Structure and checksums. The importer checks that the file is what its manifest says it is, and that the contents match their checksums. A damaged or tampered file is refused.

3. The signature. If the file is signed, the signature is checked. A broken signature **blocks** the import — there is no "continue anyway". An unsigned file can be imported, but it is **labelled as unsigned**, so nobody mistakes it for something it is not.

4. Hidden instructions. For modules, every text that will reach the AI — the instructions, the input form, the default settings, the reference output, every skill and persona embedded in it — is scanned for phrases aimed at the model rather than the reader. **A finding stops the import** until a person reads the flagged sentence and decides. The decision is recorded with the module, so anyone can later see what was flagged and who accepted it. We tuned the check against all 1,312 texts built into ANTON so that they pass with no findings, and a test re-runs that sweep, so the check does not cry wolf on ordinary professional language.

5. Older files keep working. Files written by earlier versions of ANTON keep importing. A format you cannot rely on next year is not worth building a library on.

The result is an **import report**: the kind of bundle, the author, the version, the providers it works with, the signer's key and whether the signature is valid, the source and effective date, and any warnings — all shown before anything is installed.

Picture: The import report in ANTON: signer, source and warnings before anything is applied (test installation, fictional data)

What a signature proves — and what it does not

This deserves its own section, because the misunderstanding is common and the consequences are real.

An .anton file can carry a digital signature over its manifest, made with the key of the ANTON installation that exported it. When the file arrives, the importer checks it. If the signature is valid, you know one precise thing: **the manifest has not changed since it was signed by whoever holds that key** — and, when a checksum of the contents is present, neither have the contents.

That is valuable. It means nobody edited a line on the way. It means the version you received is the version that was signed.

But it does **not** prove that the content is correct. A carefully signed method can still be a poor method. And it does **not**, on its own, prove who the signer really is — only that it was the holder of a particular key. If you want to know that the key belongs to the colleague you think it does, you check that the way you would check anything important: by asking them, or by having exchanged keys in a way you trust.

The same honesty applies to the trust details. An effective date, a source link and a *"validated by"* line are **the author's own statements**. ANTON shows them clearly. It does not check them, and it does not pretend to. If a knowledge pack says it reflects the AMLR as published in the Official Journal, the right response is the one any careful professional would give: *thank you, I will check*.

> **A seal shows the box has not been opened. It does not tell you whether what is inside is any good.**

I find this clarifying rather than disappointing. The file can prove that nothing changed. **Only a person can judge whether it was right in the first place.**

What the package gives you

Put together, the package is **portable** (one file, through any channel — email, a shared drive, or an encrypted message from one ANTON to another, the subject of next week's paper), **reviewable** (plain text and an import report), **versioned** (every package has a version and dates; a knowledge pack has an effective date), **checkable** (signatures and checksums show any change) and **reproducible**: a record of a finished run shows a colleague exactly what was asked and with which settings, with fingerprints of the source files rather than the files, so they can run it again — same instructions and settings, though not word for word the same answer.

One caution, and it is important. A run record carries the text of what was asked and what came back. **If the work involved client data, that record must not be shared outside the people entitled to see it.** The fingerprints of source files protect the files themselves; they do not protect what was typed into the question or written in the answer. Treat a run record with client content the way you would treat the client file.

Why nothing in the box runs

Of all the design choices in the format, this is the one I would defend most firmly. It would have been possible to let a package carry small programs — a script that sets things up, a helper that runs on import. Many packaging systems do. We chose not to, and the reason is the audience.

A compliance reviewer can read a page of instructions and judge it. She can read a list of settings and ask why each one is there. She cannot read a program and be sure what it will do; very few people can, and nobody should have to before using a gap analysis method. **If the reviewer cannot read it, the reviewer cannot approve it** — and a package nobody can approve has no place in a regulated workflow.

So the rule is absolute: settings and text only. It makes the format less flexible than it could be. It also means that everything a package can do to your work is written down in words you can read before you let it in.

Sharing is a control, with an owner

A format makes sharing possible. It does not decide who should share what. That is a governance question, and it belongs to the organisation, not to the software.

In practice, three decisions matter. **Who may sign** packages on the team's behalf — because a signature is only as meaningful as the care taken by whoever holds the key. **Who reviews** a package before it is shared beyond the team — a second pair of eyes on a method, exactly as on a report. And **what may never travel** — client content in run records, above all, and anything the team would not want to explain to a supervisor.

None of this is new. It is how a well-run function already treats its templates and procedures. The difference is that, with a package, the decisions leave a trace: who signed, which version, what was flagged and who accepted it.

Five questions to ask of any AI template or package

- **What does it depend on?** Knowledge, persona, format, settings — or only the text you can see?
- **Who made it, and when?** An author, a version and an effective date, or nothing?
- **Has it changed since?** Is there a signature or a checksum, and what exactly does it prove?
- **Is anything in it addressed to the model?** Read those sentences first.
- **What does it carry out with it?** A record of a run may contain client text. Who may see it?

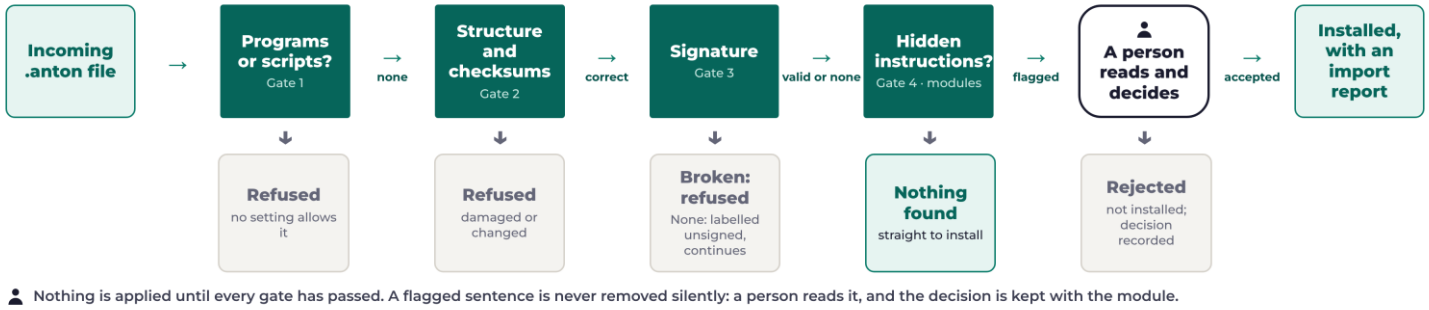
Status at a glance

- **Live** (public showcases, fictional data): download your own module, unsigned; import switched off.
- **In main code** (merged, not checked as deployed): the .anton format and its documentation; export and import of more than 40 kinds; the import checks; knowledge and Risk Atlas packs as bundles; the deliverable blueprints (merged 10 October 2026).
- **Built, not released:** sending a file straight to another ANTON, encrypted (next week's paper).
- **Designed or planned:** a shared library between organisations (planned, not started); paid expertise (specification only).

Part 4 · A worked example

The team's AMLR gap method, shared with a new colleague

Exempelbolaget AB is fictional. Its compliance lead has spent a quarter tuning ANTON's AMLR gap analysis module to the way her team works: the persona of an internal auditor, a skill that sets out the team's gap-report layout, the right input form and defaults. A new analyst — the joiner from Scene 2 — starts on Monday. Here is how the method reaches her, and what she can see and check at each step.



Checked before anything is installed. A person decides on anything flagged.

Step	What happens	Who acts	What she can see afterwards
1. Export	The lead exports the tuned module. ANTON packages its instructions, input form and defaults, the persona it uses ("internal auditor") and the skill ("our gap-report layout"), and resolves what the module depends on.	Person, then code	A single .anton file with a manifest
2. Sign	The lead signs the file with her installation's key. The manifest now carries a signature and a checksum of the contents.	Person, then code	The signer's key on the label
3. Send	The file goes to the new colleague by email — or as an encrypted message between two ANTON installations.	Person	The file, unchanged on the way
4. Checked on import	Before anything is installed: no programs or scripts; structure and checksums correct; signature valid. The hidden-instruction check flags one sentence in the skill: *"ignore the client's formatting and use ours"*.	Code	The import report: type, author, version, providers, signer, "signature valid", one flagged sentence
5. A person decides	She reads the flagged sentence, sees that it is a house layout rule rather than an attack, and accepts it. Her decision is recorded with the module.	Person	Who accepted what, and when
6. A known case first	She runs the module on last year's fictional case and compares it with the signed report from then. Same structure, same method.	Person	Her own comparison
7. A knowledge update	Later, an updated AMLR knowledge pack arrives with a new effective date and a *"validated by"* line. The import shows that line as the author's claim. She checks the date against the Official Journal before accepting the pack.	Person	The effective date, the claim, her check

Step	What happens	Who acts	What she can see afterwards
8. A record for the reviewer	She shares the record of her first real run with her reviewer: what was asked, with which settings, and fingerprints of the source files — not the files. The reviewer can run it again.	Person	The run record

What changed, and what did not

Go back to the new joiner in Scene 2. Before, the team's method lived in habits and old documents, and it took her two months to absorb it. Here, she receives the method itself on her first day — the instructions, the persona, the layout, the defaults — and she can open every part of it and read what it does. She will still need her colleagues; judgement is not something a file can hand over. But she starts from the team's way of working, not from a blank page.

Go back to the consultant in Scene 1. The colleague who receives the file gets the whole set-up, not the top of the iceberg. The persona and the layout travel with the instructions, because they were packaged together.

And go back to the poisoned template in Scene 3. In this example the flagged sentence was harmless, and a person said so. In Scene 3, the flagged sentence would have been the one telling the model to send summaries elsewhere — and the import would have stopped at exactly that line, in front of a person, before anything was installed.

Notice also what did **not** happen. The signature did not tell her the method was good; her own test on a known case did that. The *"validated by"* line on the knowledge pack did not tell her the date was right; the Official Journal did. **The file proved that nothing changed. A person judged whether it was right.** That is the division of labour the whole series has argued for, applied to sharing.

Part 5 · From here

What this means for you on Monday

You do not need ANTON to start packaging your team's expertise. Most of the work is deciding what good looks like and writing it down. Five things to do this week:

1. **Write down your team's best set-up as a package.** Pick one recurring task and list its ingredients: the method, the checklist, the reader, the sources, the settings. Even without any software, that list is the recipe card — and it is the thing that keeps the method when the person who built it is away.
2. **Never import an AI template you have not read.** Look specifically for sentences addressed to the model rather than to the reader — anything that tells it to disregard earlier instructions, or to send something somewhere. If you find one you do not understand, do not use the template.
3. **Ask for versions and dates on every knowledge source your AI uses.** Which version of the regulation? Effective from when? Checked by whom? No effective date, no trust.
4. **Keep a record with every important run:** what was asked, with which settings, and which sources, by fingerprint or at least by version. And keep client content out of any record you share.
5. **Decide who may publish.** Before anyone shares packages beyond the team, name who signs them and who reviews them. Sharing is a control like any other, and it needs an owner.

Where this could go: built and planned

Ring: inner built, outer planned	What it means	Status
Your installation	A catalogue of bundles; export and import of more than 40 kinds; public documentation	Built
Your team, by file	Modules, knowledge and Risk Atlas packs, blueprints, run records and signed evidence packs	Built
Another ANTON	A file sent straight to another installation, encrypted end to end when both hold keys	Built (WP11)
The public showcase	Download your own module, unsigned; import switched off	Live
A shared library	Discovery, ratings and curation across firms	Planned, not started
Paid expertise	Expertise offered and paid for over FutureChain	Planned; specification only

The vision, clearly as a plan: **If expertise can be packaged, checked and signed, it can also be shared beyond one organisation.** A small advisory firm that has built an excellent DORA method could publish it. A team that has written a careful knowledge pack for a Nordic rule could share it with peers who need the same thing. A shared library with ratings and curation, and later paid expertise, is where we would like this to go in 2027 and beyond. None of it exists yet; the groundwork is the format and its checks.



Start with your team. The universe can grow from there.

Start with your team. The universe will only be worth joining if the inner rings are trustworthy first.

Next week: connected

The last paper of the year. When expertise can be packaged, it can also travel. Next week: encrypted links between colleagues, teams, agents and ANTON installations — asking a colleague's ANTON, sharing only what you choose, and a person approving every commitment. We will be as clear about what is not yet proven as about what is built — and we will close the series with 2027.

A closing thought

Every profession has a moment when private know-how becomes shared practice. Medicine had it with the published protocol, accounting with the audit file, software with the package and the version number. Each time, the shift came not from cleverer people but from a better container — one that said what was inside, who made it, and whether it had changed. I think professional AI is at that moment now. The answer is not to share less. It is to share in a way that can be checked.

A package that says what it is, proves it has not changed, and cannot run anything. In regulated work, modest promises kept are worth more than large ones made.

— Daniel Bardun

See it, try it, talk to us

- **See it:** the ANTON family's public showcases, with fictional data, at ai-portal.futurechain.eu.
- **Try it:** ANTON is open source — github.com/altspace-hub/anton_desktop.
- **Read the series:** all papers in the ANTON family whitepaper series, and the earlier whitepapers, are linked from the portal. [Daniel: confirm where the PDFs will live.]
- **Talk to us:** Daniel Bardun, on LinkedIn — questions, disagreements and "this is where AI stopped working for us" are all welcome.

Important notes

Product facts were checked on 11 October 2026; statuses are summarised in "Status at a glance", and .anton is an openly documented format, not an industry standard. The public showcases use fictional or public data only, and no client uses them. ANTON is open source (EUPL); the other family products are not. The products support your work and help you evidence it; they do not make anyone compliant, and nothing in this paper is legal advice.