Dweve
Weaving Intelligence

ENGLISH EDITION · JULY 2026

MARKETING BROCHURE · 20 PAGES

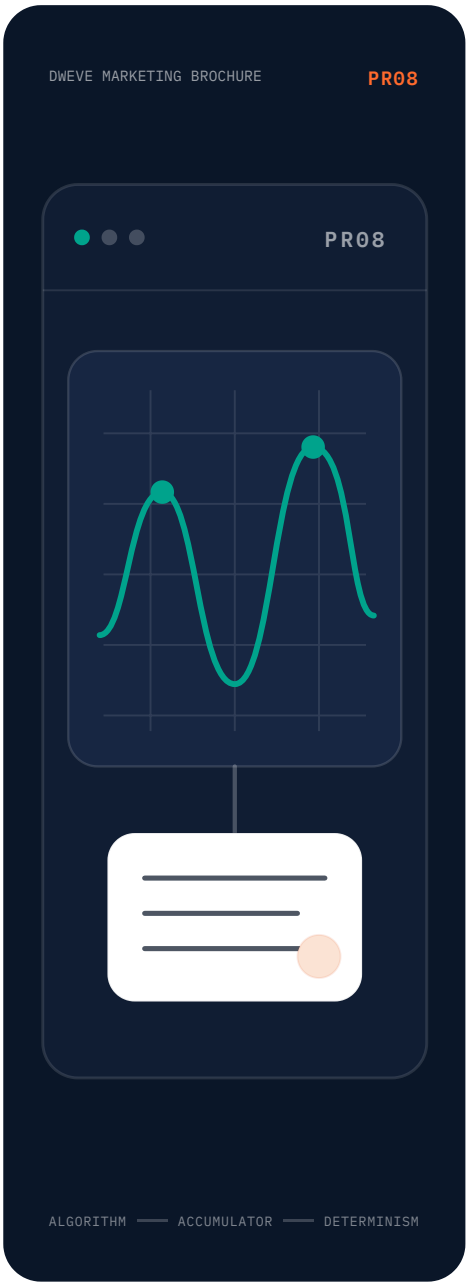
Core: the product brochure

A standalone visual introduction to Core, the job it owns, how it works, where it fits, and how to prove it.

VISUAL BRIEFING

PRODUCT LEADERS, BUYERS, ARCHITECTS, DELIVERY TEAMS, AND TECHNICAL EVALUATORS

ENGLISH



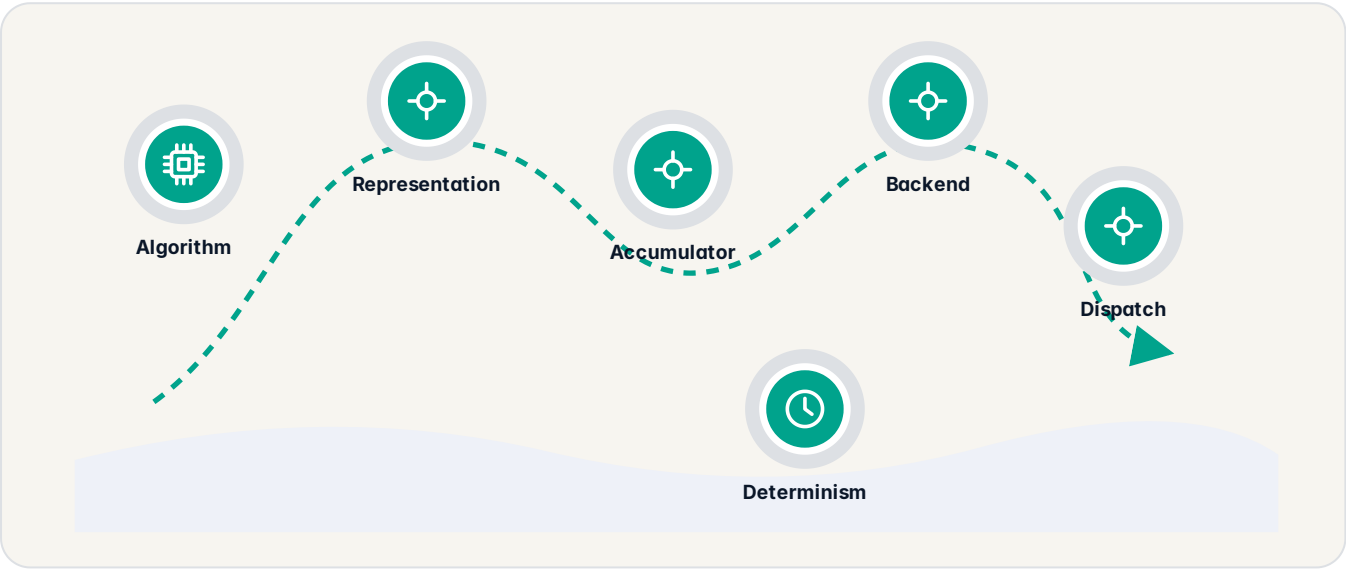
Decide whether Core belongs in the selected workflow.

PR08

THE PROPOSITION

The complete execution cell is the product

The implementation registry spans more than 2,900 operations, 76 operation categories and 25 numerical formats. Each path is determined across nine dimensions: algorithm, datatype, width, packing, accumulator, backend, instruction set, dispatch policy and determinism mode. The visual below fixes most of those dimensions so a small representative slice remains readable.



How to read this visual: follow one complete execution cell through the proposition.

MECHANISM

Read the promise against the operating reality

The implementation registry spans more than 2,900 operations, 76 operation categories and 25 numerical formats. Each path is determined across nine dimensions: algorithm, datatype, width, packing, accumulator, backend, instruction set, dispatch policy and determinism mode. The visual below fixes most of those dimensions so a small representative slice remains readable.

The actual implementation registry spans more than 2,900 operations, 76 operation categories, 25 numerical formats and nine execution dimensions. A complete cell also records datatype, width, packing, accumulator, backend, instruction set, dispatch policy and determinism mode. This visual fixes most of those dimensions to remain readable.

- Algorithm
- Representation
- Accumulator
- Backend

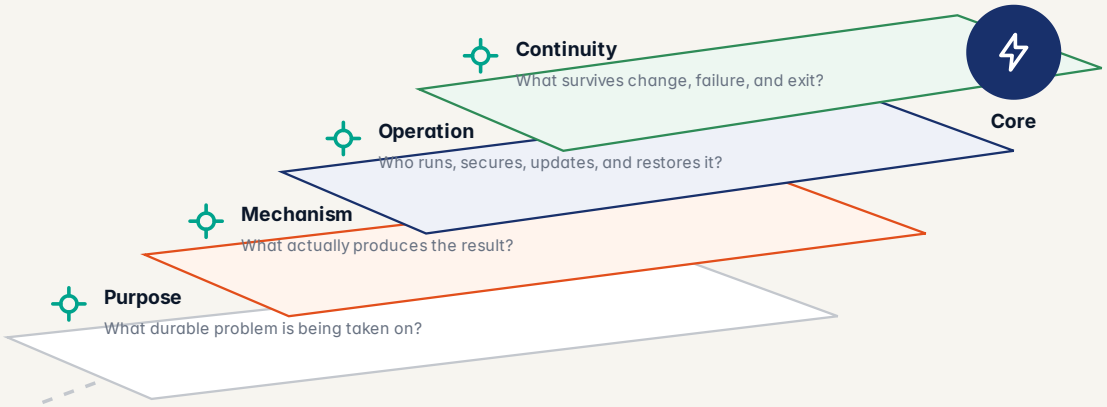
QUESTION TO CARRY

Which part of this proposition changes the outcome people receive today?

DEEPER VIEW

Read the proposition through four connected lenses

A serious decision connects purpose, mechanism, operation, and long-term responsibility.



How to read this visual: follow one complete execution cell through deeper view.

DECISION

Connect the mechanism to the responsibility

Most AI platforms cover a slice. One framework defines the model, another library handles the numbers, another system trains it, another tool compresses it, another runtime serves it, and another vendor translates it for the processor underneath. Every missing capability becomes another product to buy. Every handover becomes another place where cost, behaviour, security, and responsibility can drift apart. Dweve Core implements the complete machine beneath AI as one coherent system. Every operation. Every representation. Every machine.

The purpose is not to place a European label on a foreign technical foundation. The purpose is to give European organisations control over the actual machine that builds and runs their AI: the framework, the training engine, the inference system, the numerical model, the hardware paths, and the operational record. European origin does not automatically create compliance. It creates control.

- Purpose
- Mechanism
- Operation
- Continuity

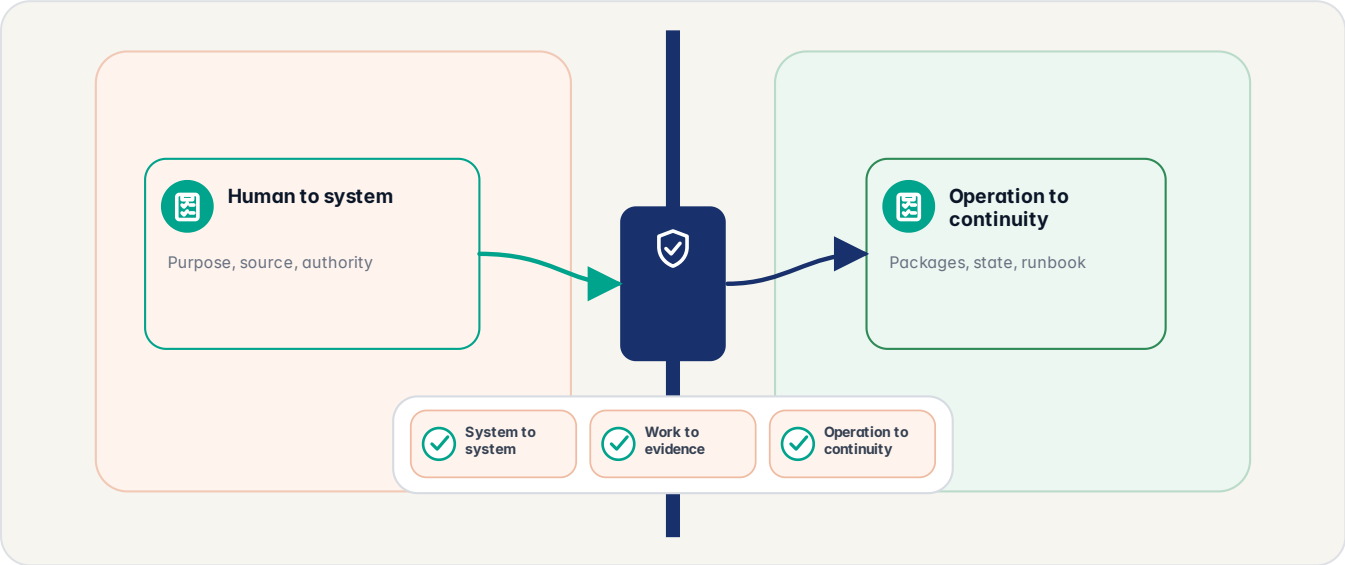
QUESTION TO CARRY

Can purpose, mechanism, operation, and continuity be reviewed together?

BOUNDARY MAP

Interfaces are promises between responsibilities

A clean boundary makes change, testing, replacement, and accountability easier to reason about.



How to read this visual: follow one complete execution cell through boundary map.

MECHANISM

Inspect what must survive the hand-off

Hard mode selects exact and reproducible paths where supported. Seeded mode permits controlled variation while preserving repeatability for the same seed. Creative mode permits ordinary sampling where variation is part of the task. The organisation does not need separate model deployments merely because the required execution contract changed. The mode can change per request. Variance becomes an explicit choice rather than an accidental property of whichever runtime happened to receive the model.

Hard uses exact and reproducible paths for the supported operation set and target combinations covered by the hard-mode contract; the purpose is bit-identical replay where the implementation and backend guarantee it. Seeded uses controlled variation; the same seed reproduces the same supported stochastic decisions, and different seeds permit deliberate variation for testing, simulation, and comparison. Creative allows temperature, top-k, top-p, and other sampling behaviour where variation is desired.

- Human to system
- System to system
- Work to evidence
- Operation to continuity

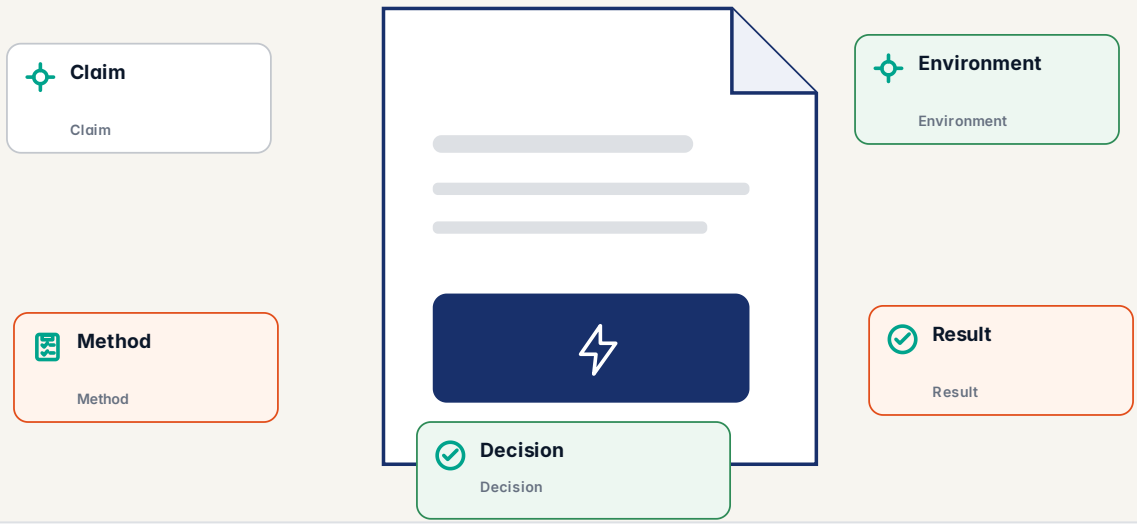
QUESTION TO CARRY

Which interface carries meaning, authority, failure, and evidence across the boundary?

EVIDENCE DESIGN

A claim becomes useful when the reviewer can test it

The brochure makes the claim visible. The proof still runs against the selected workload, environment, and acceptance criteria.



How to read this visual: follow one complete execution cell through evidence design.

DECISION

Turn the claim into something reviewable

Core's claims should be tested on the customer's hardware. The scalar path provides a reference. The optimised path provides the performance. The test suite connects them. The operation benchmarks expose the selected kernel and backend path. Dispatch tests confirm that the expected implementation was chosen. Coverage tests confirm that the supported operation and numerical categories are present. Hard-mode replay tests compare the resulting execution across the targets covered by the contract.

Core can record the operation and execution path used for supported work. This is not a claim that you should read an AI model's private thoughts. It means the system can preserve the visible work around the answer instead of asking you to trust a sentence produced by a black box.

- Claim
- Method
- Environment
- Result

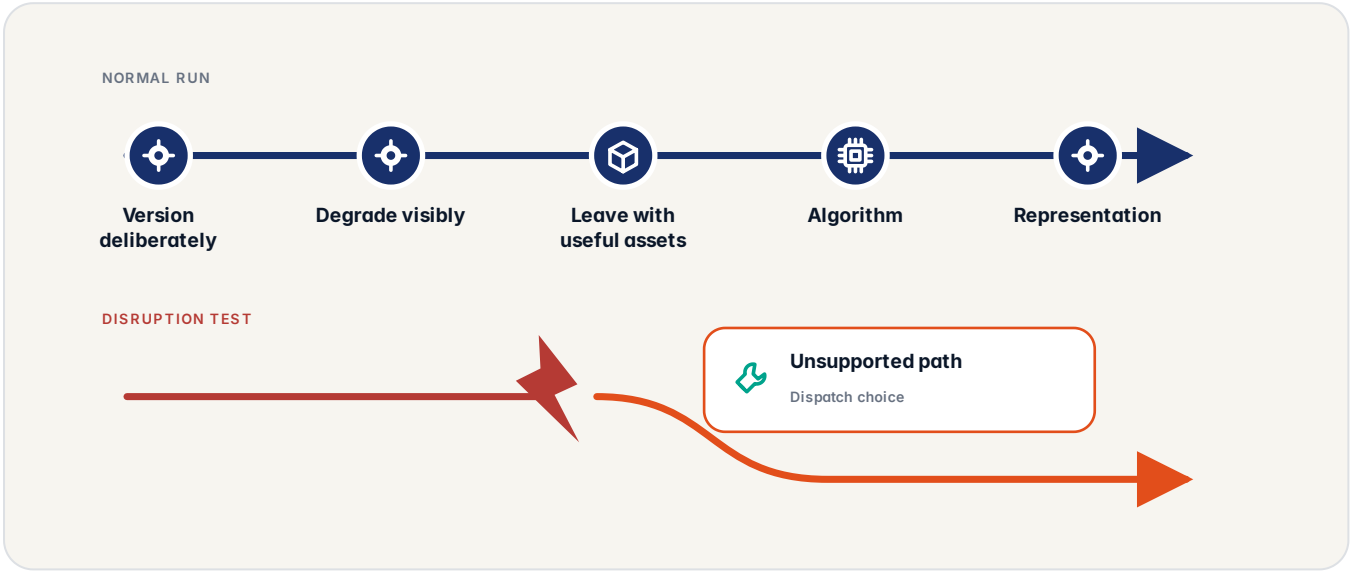
QUESTION TO CARRY

What would a reviewer need to reproduce or challenge the result?

CONTINUITY

Long-term fit includes change, failure, and exit

Evaluate the normal day and the difficult day before increasing organisational reliance.



How to read this visual: follow one complete execution cell through continuity.

MECHANISM

Test the difficult day as well as the normal day

Core covers the machine-learning stack from the lowest-level bit operations to complete model structures. The significance is not simply the number of functions. Each operation participates in the same type, representation, backend, dispatch, testing, and determinism systems. The operation surface includes bit primitives, arithmetic, vector operations, reductions, transforms, tensor operations, matrix multiplication, convolution, normalisation, activation, attention, losses, optimisation, training support, inference support, graph-level operations, and model-level composition. A binary operation and an FP64 operation are not two unrelated libraries.

Core includes a real binary neural-network training system. The output is not a conventional model compressed by an unrelated deployment tool. The target representation participated in the learning process. This distinction matters because efficient execution begins in the model lifecycle, not only at the final export step. A binary model can train in its target representation through quantisation-aware methods. A full-precision teacher can supervise a low-bit student through distillation. Progressive precision can move a model down through numerical stages.

- Version deliberately
- Degrade visibly
- Leave with useful assets
- Algorithm

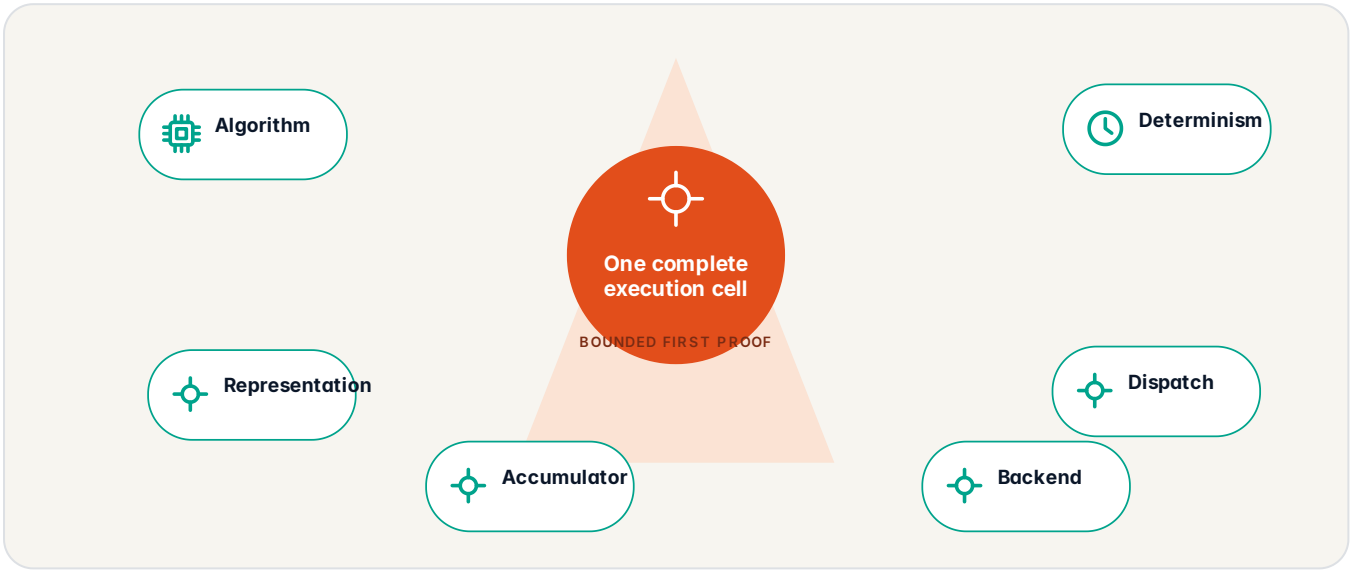
QUESTION TO CARRY

What must remain usable through change, failure, supplier loss, or exit?

USE AND FIT

Where Core can earn its place

Start with responsibilities that have a real owner, useful sources, and a consequence worth improving.



How to read this visual: follow one complete execution cell through use and fit.

DECISION

Separate useful scope from attractive distraction

When useful AI can run on the machine in front of you or on servers controlled nearby, less work has to travel to a distant data centre. Not every AI task will always run entirely on one laptop. But Core gives Dweve far more freedom to keep work local than a system built only for remote accelerator clusters. Your files can remain on your device for supported local tasks. An organisation can keep processing inside its own data centre. A hospital or public service can run a system inside the environment where its rules already apply.

A machine that uses the heaviest calculation for every task needs more memory, more electricity, and more expensive hardware. Core can use smaller and lighter ways of representing work where those methods are appropriate. The savings depend on the model, task, and hardware. The principle stays the same: use the machine the work actually needs.

- Algorithm
- Representation
- Accumulator
- Backend

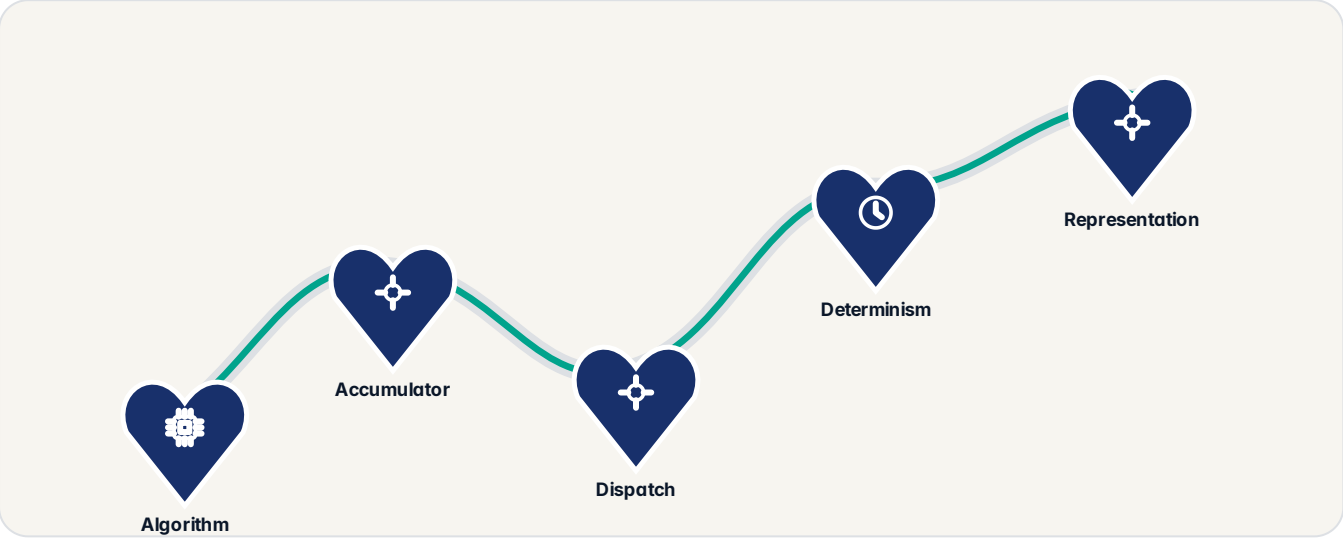
QUESTION TO CARRY

Where does this earn a place, and where should it deliberately not be used?

THE PATH

How Core moves from question to decision

A focused engagement should make the next decision easier, not create a larger promise.



How to read this visual: follow one complete execution cell through the path.

WHY THIS SHAPE WORKS

The workflow, operating boundary, evidence, and decision criteria are considered together from the start.

MECHANISM

Make every stage earn the next one

Scalar code is often described as a fallback. In Core it also serves as the universal implementation and reference target. The scalar path is not the embarrassing code left behind after optimisation. It is part of the machine's proof structure. It keeps the operation available where specialised instructions are absent. It provides a stable comparison point for optimised kernels. It supports hard determinism and reference checks where the relevant operation contract requires them.

Binary matrix multiplication on AVX-512 is not treated as floating-point matrix multiplication with smaller values. Packed-bit operations use the bit instructions available to the machine. Integer and fixed-point operations use the correct accumulator and overflow behaviour. GPU paths use the execution structure of the target. Scalar implementations remain present as the reference and universal floor. That creates an accountable answer to a simple question: what code actually performed this operation on this machine? Core can name the path.

- Algorithm
- Accumulator
- Dispatch
- Determinism

QUESTION TO CARRY

What evidence must exist before the scope is allowed to grow?

THE VALUE PICTURE

Core seen from three responsibilities

Good AI work must make sense to the person using it, the team operating it, and the people reviewing it.

How to read this visual: follow one complete execution cell through the value picture.

PRIMARY READER

Product leaders, buyers, architects, delivery teams, and technical evaluators

The first conversation.

DECISION SUPPORTED

Decide whether Core belongs in the selected workflow.

The next useful step.

DECISION

Read the same outcome from several responsibilities

A licensed team may operate Core on organisation-controlled clusters or workstations. A regulated programme may require a fully isolated estate. A product may move part of its inference to the edge, while a browser application may need a local execution target. Core keeps those licensed operating postures inside one machine without claiming that every backend supports every cell. Core is not a managed-service product.

Most AI products are only the part you see. You open the app, type a question, and receive an answer. Behind that simple screen, the work may pass through software, computers, and services owned by several different companies, and the app works only while that whole chain continues to work. Dweve built Core differently. Core is the complete AI machine underneath Dweve's products: the many small operations, calculations, methods, and hardware paths required to make the AI work. That is why Dweve can decide how the work should run instead of sending every task through the same distant system.

- Algorithm
- Accumulator
- Dispatch
- Representation

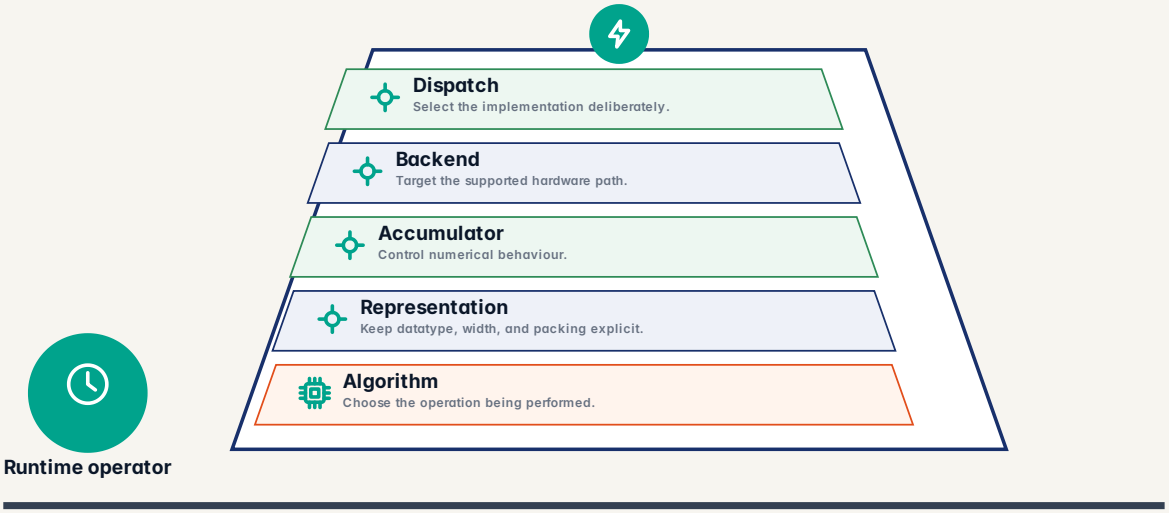
QUESTION TO CARRY

Can the same result satisfy its user, operator, and reviewer?

THE OPERATING MODEL

The control model around Core

A credible proposition connects the visible result to the less visible responsibilities underneath it.



How to read this visual: follow one complete execution cell through the operating model.

MECHANISM

Name the controls behind the simple interface

An organisation should be able to change a technical constraint without losing the model. Core keeps those changes inside one machine. The representation may change. The packing may change. The accumulator may change. The selected kernel may change. The processor may change. The computational responsibility stays with Core. A binary operation on a CPU should still be the same operation when it is deployed through a different instruction set. An integer model should not require an unrelated inference system because it no longer resembles the framework's preferred floating-point path. A browser deployment should not mean rebuilding the application around a new computational model.

Many systems expose a quantised datatype but hide the arithmetic model behind it. Core separates the value representation from the way intermediate results are accumulated. This is not a global framework switch. The choice belongs to the operator. That makes mixed numerical models explicit, inspectable, and compilable. A packed binary input may use integer population counts. A low-bit integer operation may accumulate into Int32. A fixed-point operation may use a wider fixed-point accumulator.

- Algorithm
- Representation
- Accumulator
- Backend

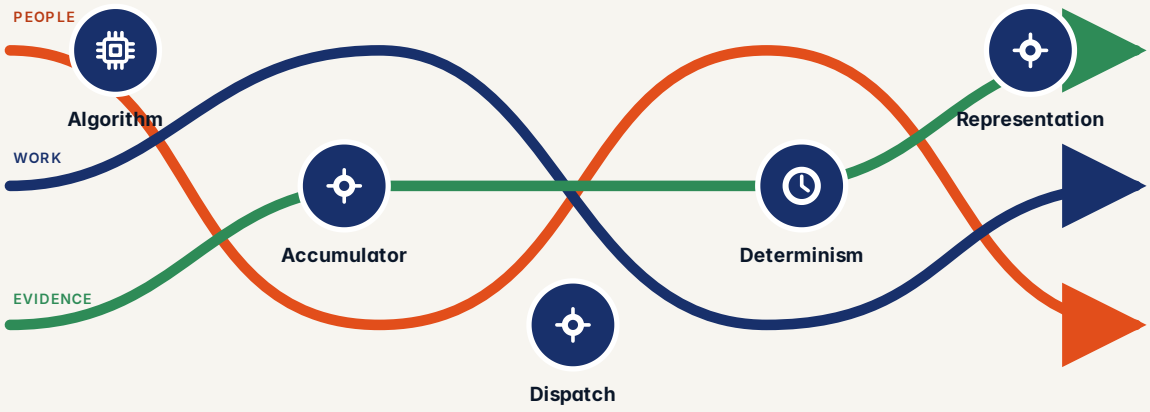
QUESTION TO CARRY

Who owns each control when the interface appears simple?

THE RESPONSIBILITY FLOW

People, Core, and evidence move together

The work becomes easier to govern when each stage leaves a clear hand-off.



How to read this visual: follow one complete execution cell through the responsibility flow.

DECISION

Keep every hand-off attached to an owner

Most AI infrastructure has a preferred numerical centre. Everything close to that centre works well. Everything outside it becomes an extension, a fallback, an export path, or a research feature. Core has no single privileged datatype. Binary and ternary computation are real. Low-bit integer computation is real. Fixed-point computation is real. FP8, FP16, BF16, FP32, and FP64 are real. Storage precision and accumulator precision are separate decisions. A model can mix representations according to what each operation requires rather than forcing every part of the system through one numerical assumption.

Packed binary words, ternary encodings, bit planes, quantised blocks, scalar layouts, vector layouts, and backend-specific layouts each change how the kernel must move data. Core does not hide those differences behind a claim that one generic kernel supports everything. The supported implementations are tuned for the packing they actually receive.

- Algorithm
- Accumulator
- Dispatch
- Determinism

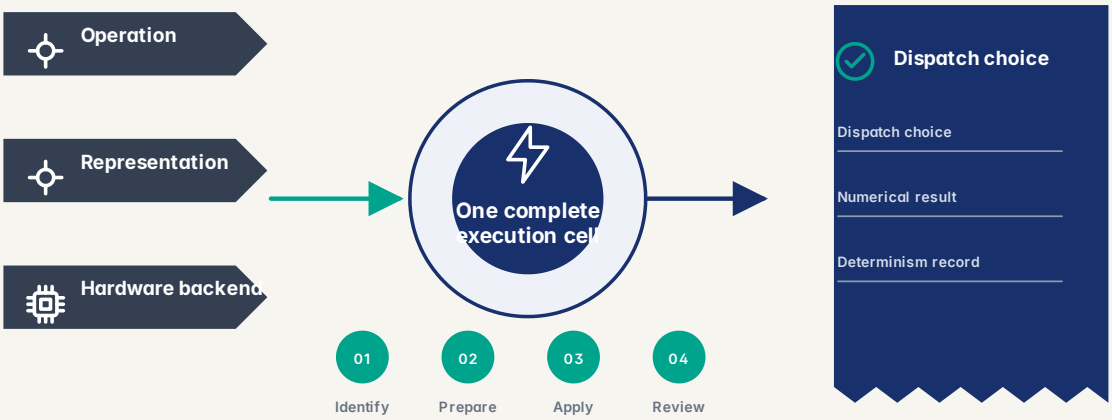
QUESTION TO CARRY

Is every hand-off visible, bounded, and assigned to an owner?

SOURCE TO RESULT

Core: from authoritative source to result

The work stays explainable when source identity, transformation, decision, and hand-off remain visible between systems.



How to read this visual: follow one complete execution cell through source to result.

MECHANISM

Keep the basis visible as the work changes form

Operate licensed Core on your own servers in your own data centre. Your operators hold the keys; source escrow and source rights remain separate unless the agreement expressly includes them.

Coverage would mean less if the model still changed systems between development and production. Core keeps model authoring, numerical representation, training, optimisation, inference, dispatch, serving, benchmarking, and replay inside the same computational machine. Quantisation-aware training does not have to be bolted onto a model after training. Knowledge distillation does not have to live in a separate experimental stack. Progressive precision does not have to end in an export into another runtime. The model can be trained according to the representation it is intended to use and executed by the same system that understood those choices during training.

- Identify
- Prepare
- Apply
- Review

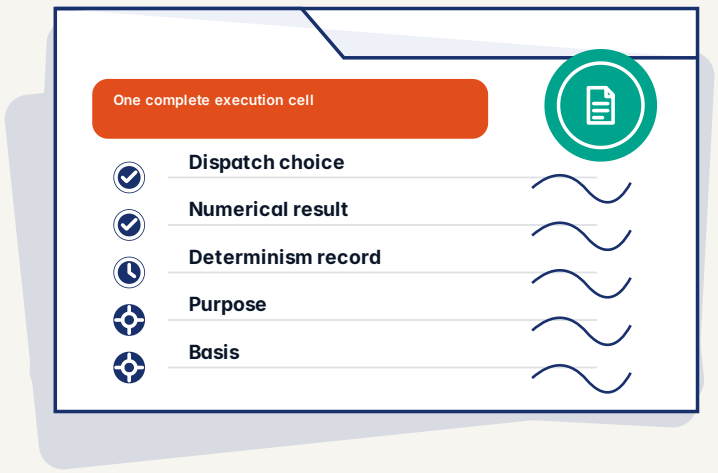
QUESTION TO CARRY

Can the result still be connected to the exact basis that shaped it?

EVIDENCE PACKET

The evidence Core leaves with the work

Capture enough to explain the result, inspect authority, investigate failure, and make the next decision without rebuilding the story from memory.



How to read this visual: follow one complete execution cell through evidence packet.

DECISION

Preserve the record needed by the next question

Core is available through Rust and Python. Both surfaces describe Core models. They do not define separate execution architectures. The language is a front door. The machine behind it remains Core. Rust provides direct control, static types, native integration, standalone binaries, thread and memory control, and access to the system without a language runtime in production.

Core contains many different ways to perform the work. It can use very small, efficient representations where they are enough. It can use larger and more precise calculations where the task needs them. It can choose a different path for a different kind of operation. The complete workshop is already inside the machine.

- Purpose
- Basis
- Authority
- Action

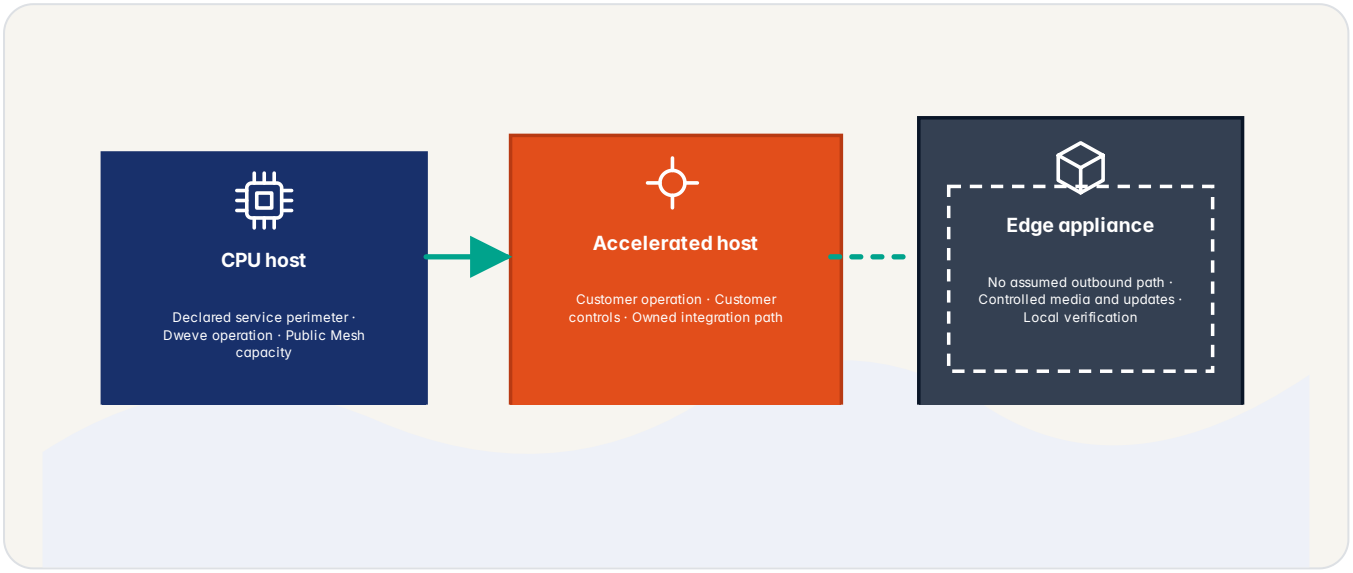
QUESTION TO CARRY

Does the retained record answer the next difficult question without reconstruction?

DEPLOYMENT CHOICE

Choose the operating posture for Core

Location is only one dimension. Decide who operates, who holds keys, which dependencies exist, how updates arrive, and what continuity requires.



How to read this visual: follow one complete execution cell through deployment choice.

MECHANISM

Compare operating responsibility, not location alone

Infrastructure decisions should not force a model rewrite. Core keeps the model, operator surface, numerical design, and execution contract inside the same system while the deployment posture changes. The selected backend may differ. The organisation does not need to reconstruct the AI stack around it.

An AI stack can appear complete while every important responsibility belongs somewhere else. The organisation may own the model file. It does not own everything the model requires in order to exist. The model framework relies on a numerical library. The numerical library relies on an accelerator runtime. The accelerator runtime relies on a hardware vendor. The quantisation pipeline relies on another tool.

- Begin through managed Fabric
- Run on customer infrastructure
- Close the environment
- Algorithm

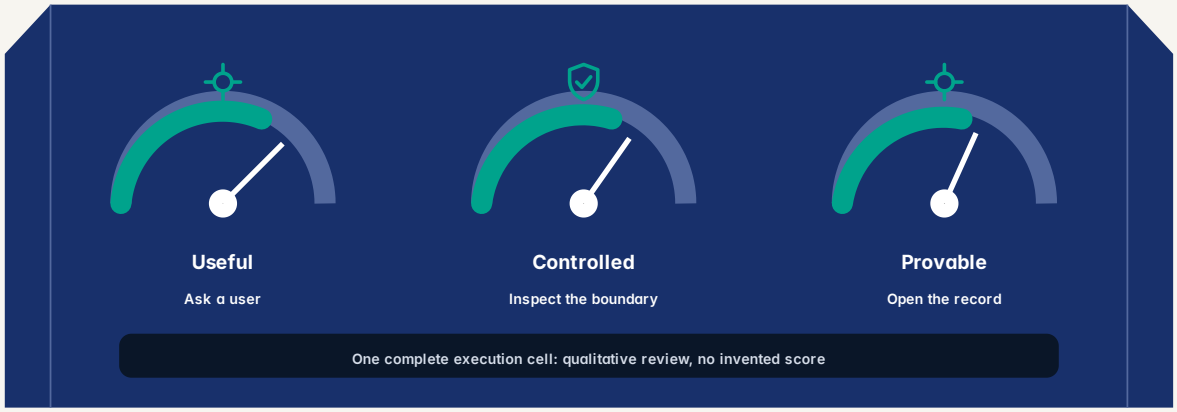
QUESTION TO CARRY

Who holds infrastructure, keys, updates, logs, recovery, and the exit path?

WHAT TO LOOK FOR

Signals that Core is earning trust

Progress is a useful outcome under understood control, supported by reviewable evidence.



How to read this visual: follow one complete execution cell through what to look for.

DECISION

Use signals that can change the next decision

A creative request can allow variation. A test can use a fixed seed. A supported hard-deterministic operation can use the strict path required for replay. The product does not have to accept accidental variation merely because the underlying AI software was never designed to control it.

AI infrastructure is usually described through APIs. Core is better understood through implementations. The real unit of coverage is not an operation name. It is the complete execution cell: algorithm x datatype x width x packing x accumulator x backend x instruction set x dispatch policy x determinism mode. Core implements the supported matrix across more than 2,900 operations, 76 operation categories, 25 numerical formats, multiple packing models, CPU and GPU backends, browser targets, scalar reference paths, and specialised deployment targets. Every supported cell is real code. Not a capability flag. Not a generic wrapper over one preferred representation. Not an export into somebody else's runtime.

- Useful
- Controlled
- Provable
- Algorithm

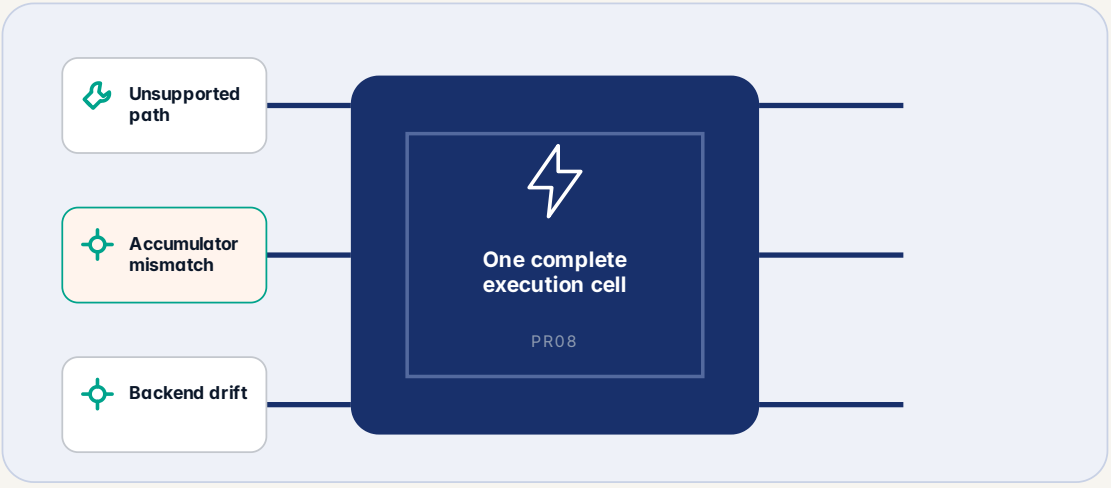
QUESTION TO CARRY

Which signals would change the next decision rather than merely impress the room?

HONEST LIMITS

Test how Core fails before the first success demo

Visible failure is more useful than a polished result whose limits remain unknown.



How to read this visual: follow one complete execution cell through honest limits.

MECHANISM

Design the failure before presenting the success

When a production result changes in a fragmented stack, responsibility is distributed. The framework blames the exporter. The exporter blames the runtime. The runtime blames the kernel. The kernel blames the driver. The driver blames the hardware. Core brings those responsibilities inside one system.

Core is designed and built by Dweve in the Netherlands. That matters because the machine underneath the product determines where data can run, which outside companies are required, and who can be held responsible when something fails.

- Unsupported path
- Accumulator mismatch
- Backend drift

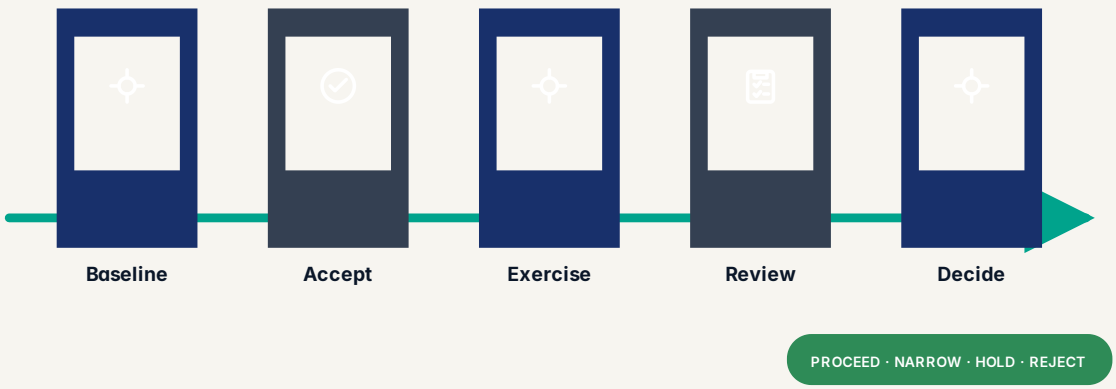
QUESTION TO CARRY

What happens when a source, permission, target, model, or supplier is unavailable?

ACCEPTANCE PATH

Close the Core proof with a decision

Agree the decision path before the demonstration so a strong or weak result can change what happens next.



How to read this visual: follow one complete execution cell through acceptance path.

DECISION

Close the proof with an owned decision

The inference server relies on an exported representation. The production deployment relies on all of them remaining compatible. A change in hardware can force a new runtime. A change in precision can force a new training path. A change in deployment can force a new serving system. A new regulatory requirement can expose gaps nobody designed the original stack to answer. The problem is not simply that there are too many products. The problem is that each one covers only part of the computational space.

The best path can differ by operation, datatype, packing, tensor shape, alignment, policy, and available instruction set. Core's dispatcher resolves the implementation at runtime according to the current host and execution contract. Dispatch is not a one-time declaration that the whole model belongs to one device. It is a decision attached to the operation. The selected path can be recorded and verified. The same process may use an AVX-512 binary kernel for one operation, an AVX2 integer path for another, a scalar exact path for a deterministic check, and a GPU backend for a larger floating-point workload.

- Baseline
- Accept
- Exercise
- Review

QUESTION TO CARRY

Are success, counter-metrics, failure cases, and stop conditions agreed first?

ADOPTION

Adopt Core through owned choices

Move from one bounded result to a repeatable operating capability without making the scope vague.

How to read this visual: follow one complete execution cell through adoption.

MECHANISM

Expand only what proved reusable

Core keeps changes in numerical width, packing, hardware and deployment inside one implementation system. An algorithm does not become a different product because it moves from an x86 server to ARM, a GPU, a browser or a specialised target. The operation remains named, inspectable and owned throughout the move.

That means supported Dweve workloads can run on modern laptops, office computers, servers, edge devices, and other machines without making a dedicated AI graphics card the minimum requirement. Larger jobs may still use more powerful hardware. The difference is that the whole system was not designed around one expensive processor from the beginning. The computer is chosen for the work. The work is not forced to follow one computer vendor.

- Prove one workflow
- Own the operation
- Repeat the pattern
- Add responsibility

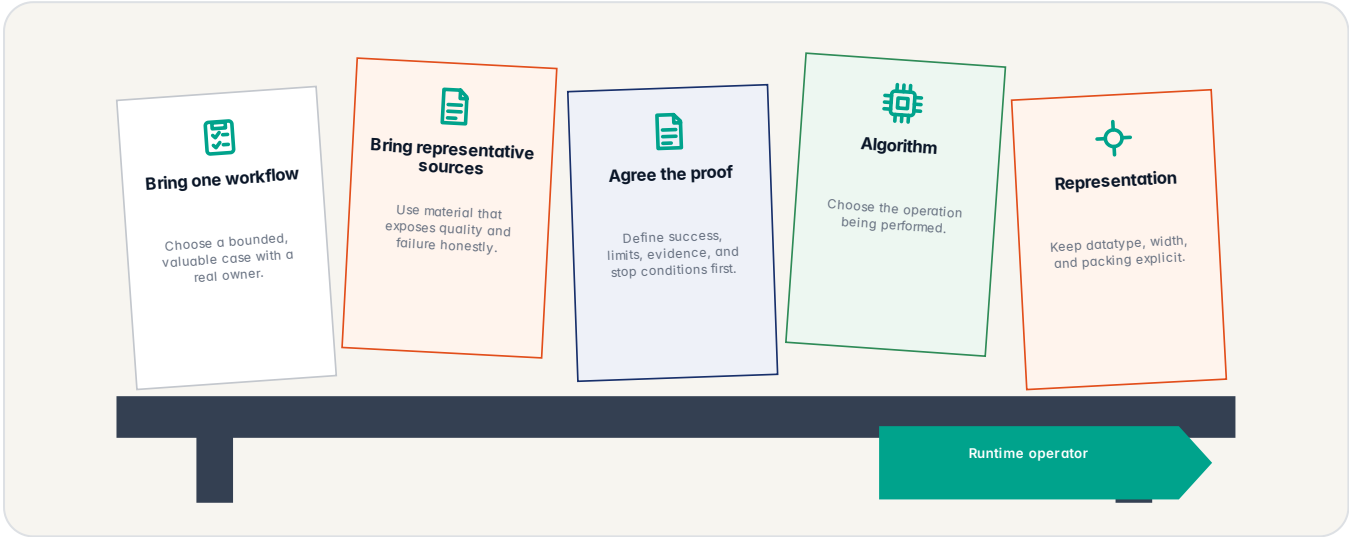
QUESTION TO CARRY

What can be reused safely, and what requires a new decision and proof?

WHAT TO PREPARE

What to bring to the Core working session

A small amount of real context produces a better conversation than a broad catalogue of hypothetical features.



How to read this visual: follow one complete execution cell through what to prepare.

KEEP IT BOUNDED

Sensitive production data is not required for the first conversation. Bring representative material and the rules that define a useful result.

DECISION

Bring enough reality to make the session useful

Some Dweve features can operate locally because the machine they need is already present. When the feature and its required data are available on the device, Core does not need to contact a remote service merely to perform its calculations.

A broad API is easy to advertise. Broad machine-level ownership is much harder. Core does not count an operation as covered merely because a generic implementation compiles on another target. The supported paths are implemented and tuned for their actual combination of operation, representation, packing, backend, and instruction set.

- Bring one workflow
- Bring representative sources
- Agree the proof
- Algorithm

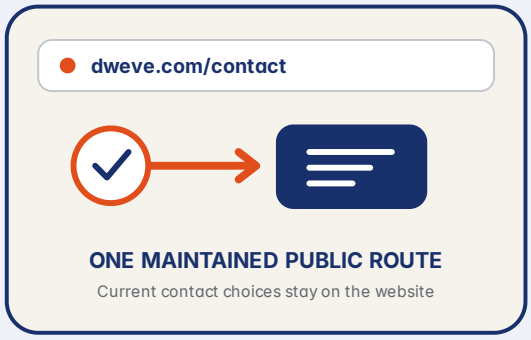
QUESTION TO CARRY

Which real sources, rules, owners, and examples will make the first session honest?

THE INVITATION

Continue the Core conversation

Scope a briefing, demonstration, or proof around the workflow and decision that matter to your organisation.



CONTINUE ONLINE

dweve.com/contact

Current route and contact choices

START A CONVERSATION

Continue through the current contact page

Scope a workflow or ask a technical question, then use the contact page to begin.

START	**dweve.com/contact**	Use the maintained public contact route
BRING	**One bounded question**	Workflow, audience, decision, and desired evidence
CHOOSE	**Briefing · demonstration · PoV**	Select the smallest useful next step
VERIFY	**Current scope and terms**	Confirm availability, pricing, and responsibilities before commitment

MECHANISM

Turn the brochure into one bounded next step

Core does not treat one accelerator ecosystem as the definition of AI and everything else as a lesser port. Its operations have scalar reference implementations and target-specific paths across supported CPU, GPU, browser, and deployment backends. The runtime selects an implementation according to the available machine and the policy attached to the work.

Core contains thousands of operations and many ways to run them. You do not have to select any of them. The Dweve product makes those decisions according to the task and the rules around it.

- Start: dweve.com/contact
- Bring: One bounded question
- Choose: Briefing · demonstration · PoV
- Verify: Current scope and terms

QUESTION TO CARRY

What is the smallest useful next conversation Dweve should prepare?