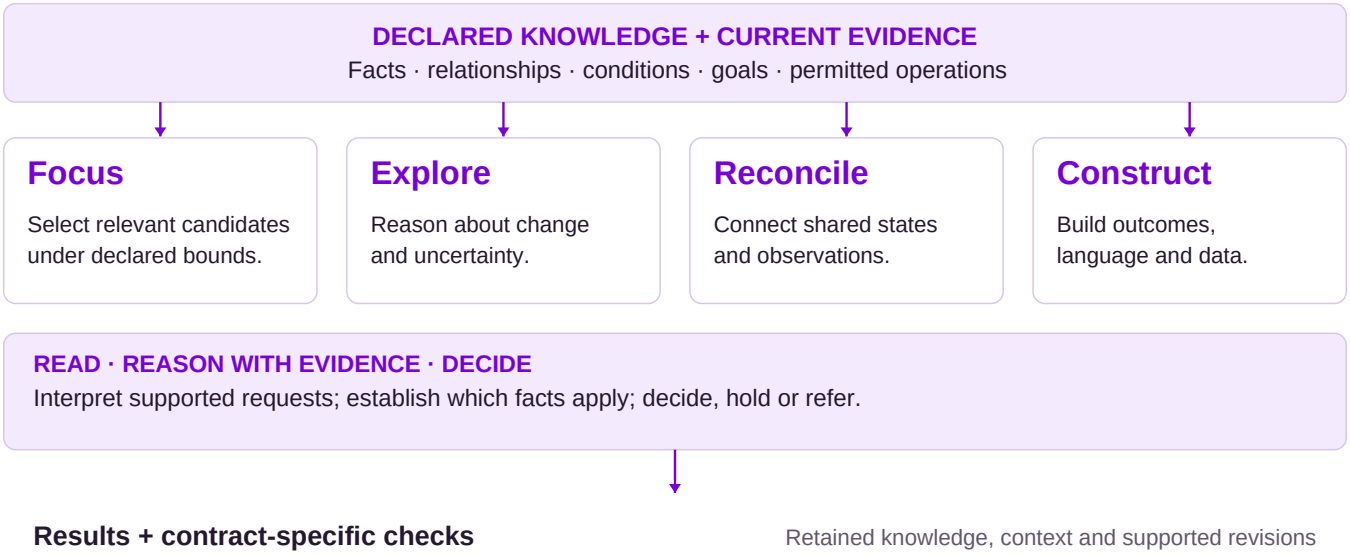
01 / ILLUSTRATED INTRODUCTION

AI that reasons with declared knowledge.

MDIL is a declared reasoning AI. It derives conclusions, explores possible developments and constructs decisions, plans, language and data from explicit knowledge. The knowledge behind each result remains useful for checking, revision and reuse. MDIL Kide is the first release of MDIL.

A fact can support a conclusion. A condition can change which actions are possible. Several facts can become an explanation. A shared construction can recreate a much larger artifact.



Architecture map. Different operations share declared knowledge and evidence; a task uses the capabilities it needs.

What connects the capabilities

Explicit knowledge and current evidence supply the basis. A task declares what must hold; its reasoning and construction operations produce a result with the applicable checks. Retained knowledge remains available for further work.

A task uses the operations it needs. The map describes the architecture; current Core, demonstration and study implementations are identified in the system card. A model can work through MDIL: it declares a task once and continues from its stored results. Models can also help at authoring, intake or other configured boundaries.

02 / ILLUSTRATED INTRODUCTION

From a message to a supported result.

A declared reader turns a supported message into an explicit request. Evidence reasoning establishes which facts apply. Relationships and priorities then govern a decision, a hold or a review outcome.

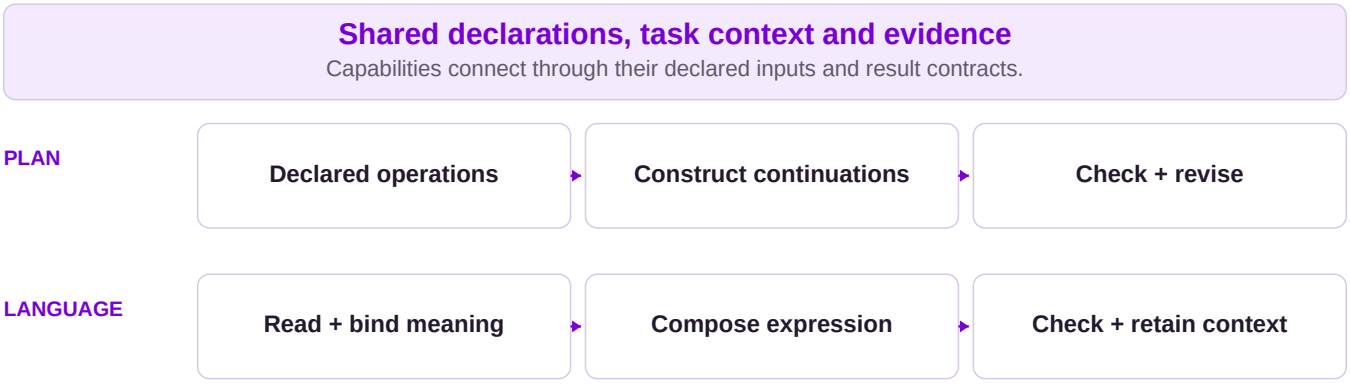
Request	Declared relationships	Conditions
Which desk should handle this equipment?	Equipment belongs to a service group; the group has a responsible desk.	A permit allows dispatch. A missing permit holds it. A hazard requires review.

Language is constructed from meaning

For the request for two assets' service prices, supported equal prices produced:

“Pump Larch and Valve Juniper each cost €210 to service.”

Different prices need distinct statements. Expired or conflicting information changes the kind of answer. The writer composes from supported meaning and an independent reading checks the expression.



Two implemented paths. Other capabilities retain their own contracts and implementation forms.

Implemented planning and language paths share declarations and evidence while retaining their own checks.

Retained context supports the next turn. The request, accepted observations and their declared relationships remain distinct from a model's proposed text.

03 / ILLUSTRATED INTRODUCTION

Reason about what is known and what could happen.

A plan that needs tomorrow's result today is not a usable policy today. Trajectory and causal reasoning keep the available information explicit when evaluating possible developments.

2,835 cases

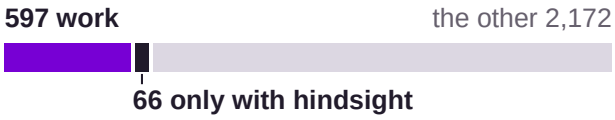
Finite policy cases checked against explicit policy enumeration.

66 hindsight-only

Cases that looked feasible only when future information was granted.

597 robustly feasible

Cases with a policy that works under the declared disturbances.



The same 2,835 cases on one strip. Each case asks one thing: does the plan work with only what is known when each decision is made?

Shared evidence stays shared

The same observation may appear in several time windows or reports. Repeating it does not supply another independent observation. Projections preserve this relationship.

ONE OBSERVATION. SEVERAL VIEWS.

Source observation

80%

recorded probability

Ten copies of that evidence

Copies share the same source.

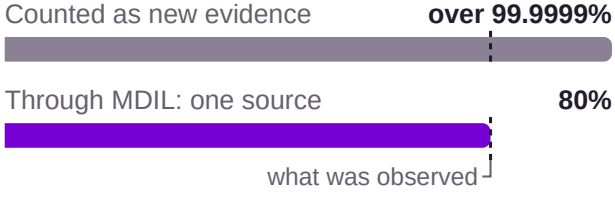
MDIL retains

80%

No extra independent evidence.

Treating the copies as independent would inflate this example above 99.9999%.

Recorded synthetic shared-observation example. Duplicate observations do not create independent evidence.



Ten copies of one observation. Counted as new evidence they would say over 99.9999%. MDIL keeps the one source, so the answer stays 80%.

Across the recorded overlap study, 5,000 duplicate events caused no probability changes. Conflicting observations and projected events presented as observations were rejected. Current Core includes trajectory and projection mechanisms; the figures describe their named reference studies.

04 / ILLUSTRATED INTRODUCTION

Generation reaches beyond one kind of output.

A plan

Steps + conditions + goals

A valid continuation
with its evidence.

Construct + check + reuse

Language

Meaning + facts + constructions

A composed expression
that preserves the content.

Construct + check + reuse

Exact data

Installed component + new input

A reconstructed artifact
equal to the source.

Construct + check + reuse

Three generation contracts: constructive results, declared language and exact reconstruction.

A plan can adapt to declared conditions

In the maintenance demonstration, staffing, inspection windows and budget determine which continuations work. A missing worker leads to a permitted service alternative. A tight inspection window requires waiting. An insufficient budget produces no feasible plan.

Current Core also supports repairing retained constructions when specified conditions tighten, with results matching full recomputation. Changes outside that repair contract explicitly require fresh construction.

3.8 kB newly sent

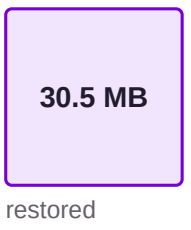
A small transfer supplies the new information.

402 kB installed once

The receiver already holds the reusable component.

30.5 MB restored

944 files reconstructed exactly from that combination.



The same three amounts, drawn to scale. The area of each square is its size: what was restored, what the receiver already held and what was newly sent.

The reconstruction result shows why prior knowledge matters: a small message can carry a large result when the receiver already understands how to construct it. Exact file equality, supported linguistic meaning and feasible plans remain different contracts.

05 / ILLUSTRATED INTRODUCTION

Prepared knowledge can serve without a model call.

Prepared MDIL and a local Qwen2.5-1.5B received the same twelve explicit service-price requests and source facts. The comparison measured serving after preparation on one machine.



68× faster

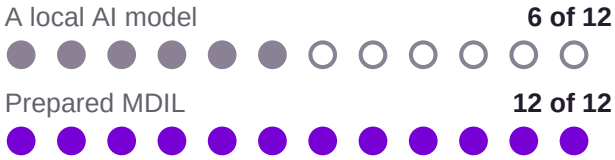
Median returned response, prepared MDIL on a CPU path.

99.5% less

Gross chip energy per attempt in this comparison.

All 12 complete

Every requested price stated, no unsupported price. The model completed six of the same twelve.



The same twelve requests, one dot each. A filled dot is a complete answer: every requested price stated, no unsupported price.

The value extends beyond a latency number

Covered execution can operate locally without a model call. Declared conditions help control decisions and information use. Retained constructions give later work a starting point. More useful coverage means less repeated work and more resources available for new problems.

The pilot used already interpreted requests. Preparation, intake, updates and fallback belong in whole-application accounting. The model work of a whole task is counted on pages 06 and 07 and in the system card. Two post-run semantic reviews applied one completeness rule to both methods. Full methods and evidence: the system card.

06 / ILLUSTRATED INTRODUCTION

Declare. Reason. Keep.

MDIL is used in three steps. The model, or a person, states the task and makes the declarations. MDIL reasons, computes the answers, checks them and keeps them.

Declare

A model, or a person, states the task once: the facts, the conditions and the result that is wanted.

Reason

MDIL reasons: it computes the answers in order, by itself, and checks each one.

Keep

The result is stored with its check. A change redoes only what moved. Another session, another AI model or a small local model continues from it.

One task, done both ways

The task is a long calculation: 320 steps, and an answer 511 digits long. It was set as three jobs: build it, pick it up, change it. The same AI model, Claude Sonnet 5.5, did the three jobs alone and then through MDIL.

0 of 3 jobs right



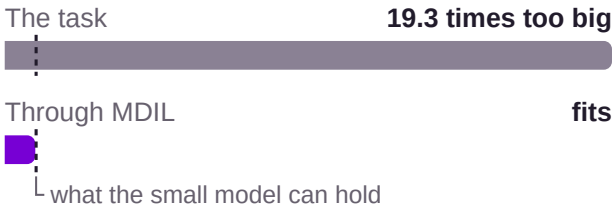
The AI model alone, working by itself, in 24 turns.

3 of 3 jobs right



The same model through MDIL, in 18 turns and with about half the writing. Each answer was checked by a separate program.

Then a small local model continued the work



A small local model is the kind a company can run on its own servers. The task is 19 times what it can hold. Through MDIL its whole session fits, and it returned the checked answer in 8 seconds. The large AI model built the work once, in 8 turns and 108 seconds.

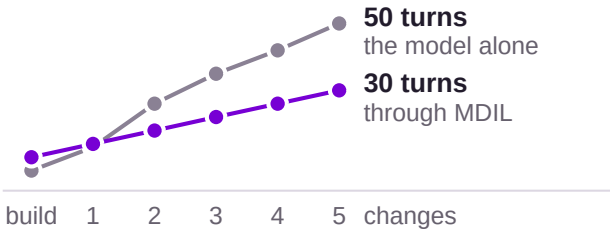
Task runs of November 2026, graded by a separate program that reads the stored results. A turn is one step of an AI model in a session. Full figures and methods: the system card.

07 / ILLUSTRATED INTRODUCTION

Build it once.
Then state each change once.

A supply chain of 48 work stations. Each station's order depends on what its neighbours order, so one change at the head moves the orders behind it. The AI model alone works the whole chain out again for every change. Through MDIL the model builds the chain once and then states each change once. MDIL reasons, computes, checks and keeps the rest.

30 turns against 50



The build and five changes, added up: 30 turns through MDIL, 50 for the AI model alone.

20 turns against 44



The five changes: 20 turns through MDIL, 44 for the AI model alone.

Only what moved is redone. The second version of a software repository needed 26 new objects. The other 1,452 were kept as they were.

Using MDIL well

One request, not many

When MDIL is used as designed, one request runs the whole project. The model only makes the declarations.

A reference, not the raw data

The structure is declared once and shared by reference. Any model, even a small local one, continues from it.

Checked, traceable, reusable

For a task the model solves alone in minutes, the value is that every answer is checked, traceable and reusable.

The same model and the same task texts were used both ways. Every station's answer was checked by a separate program. Full figures and methods: the system card.

08 / ILLUSTRATED INTRODUCTION

Knowledge becomes shared infrastructure.

People and AI create expertise. MDIL makes supported knowledge executable and reusable. A reviewed contribution can become a capability other devices, teams and applications use and improve, with their own circumstances declared locally.

Devices	Organisations	Discovery
Useful reasoning and construction close to the work.	Shared expertise with local rules and information boundaries.	Less repeated work; each finding supports the next.

Bring a capability into an existing application

Declare the knowledge, conditions and desired result, yourself or through an AI model. MDIL **reasons**, computes the result, checks it and keeps it. **Use** the decision, plan, text or data with its evidence. When something changes, **state the change**: what depends on it is worked out again. **Publish** the structure, and another session or another model continues from it.

An agent can declare a task once and come back to its checked results; a service can compose grounded language; a planner can evaluate continuations; a receiver can reconstruct data. HTTP API access serves identified beta integrations; embedded Core is available by agreement from a versioned configuration. Interface releases name the supported capabilities.

Access

MDIL Kide is the first release of MDIL. MDIL launches on 17 November 2026 at 17:11 at Allas Pool in Helsinki, with three products built on it. MDIL Portal is the development environment where developers, integrators and enterprises build on MDIL, with the MDIL SDK and MDIL MCP opening in beta; access to the beta is by agreement, and the Portal’s own availability is announced separately. The beta includes compression and reconstruction. REMAP turns a goal stated in one sentence into declared AI agents and loops that run a company’s work on its own systems; it is in closed beta by invitation. Airut is an AI assistant for businesses and people, in the chat they already use; its waitlist is open.

Join at mdil.ai
Review the system card and companion materials at review.mdil.ai.

The system card contains implementation and evidence detail. The technical summary explains the connections across mechanisms. This illustrated introduction follows the same account at a visual level.