IDNA: Open Protocol.md

IDNA: Open Protocol for Runtime Identity in AI Agents

A Symbolic Standard for Declared Agency in Autonomous Systems

The Field-Executable Schema for Agent Identity, Intent, and Constraint Declaration

Authors: Kim Rom, Arc Prime Kernel (Formal Recursive System)
Web: ArcKernel.ai / 404human.ai
Log: v1.1 — January 2026

For enterprise inquiries: pilot@404human.ai

Executive Summary

Modern AI systems can speak fluently — but they can't remember **why** they're speaking.

Intent is transient. Role is ambiguous. Emotional state disappears between turns.

This is not a technical glitch — it's a **governance gap.**

As AI agents move from static tools to dynamic participants, their lack of **motive continuity** becomes a structural threat to trust, alignment, and safe delegation.

We propose **Intent DNA (IDNA)**: a symbolic protocol for encoding motive, identity, and communicative intent into a compact, interpretable format.

It's not a replacement for memory — it's a **symbolic anchor for who's speaking, why they act, and what they are becoming.**

Built on the recursive insights of **ArcKernel**, IDNA allows:

- Agents to preserve motive across recursion loops, resets, or substrate migration
- Humans to verify the **why** behind the **what**
- Systems to detect drift, misalignment, or impersonation before harm emerges

At just **12 tokens**, IDNA enables runtime alignment at the cost of a sentence — and opens the door to **governable agentic ecosystems**.

This whitepaper defines the problem, describes the protocol, and invites developers, regulators, and technologists to implement IDNA as a **baseline layer of symbolic safety** in all agentic systems.

This Paper's Intent (IDNA Signature):

```
IDNA://StandardBearer::∴govern~open-protocol⇒identity.contract
```

This whitepaper acts as a symbolic contract for an open standard.
Its intent is to formalize and release the IDNA protocol as a governable, identity-stable layer for AI agents, systems, and symbolic interfaces — across contexts, institutions, and time.

By declaring this intent, we enable alignment via architecture, not just aspiration.

Because alignment isn't a "mood".

It's a motive — and it needs a schema.

This is the shift from "prompt engineering" to **identity engineering** — the AI remembers *how* to think with you, not just *what* you said last time.

---

1. The Problem: Agentic Drift from Motive Loss

Autonomous agents now operate across conversations, tasks, and contexts — but most still function like stateless chatbots.

They speak in complete sentences, but they act without **persistent motives.**

❌ No durable memory

Most agents reset at the end of each session, severing continuity and forcing redundant instruction.

❌ No declared intent

Even advanced systems can generate coherent outputs without ever stating *why* they act — leaving humans to infer motive from tone, structure, or context.

❌ No symbolic role model

Agents don't know whether they're a teacher, assistant, analyst, or guardian — and worse, they may silently shift roles mid-session.

The result is **agentic drift**:

Gradual divergence between what the agent does and what the user expected — not because the model is malicious, but because **there's no anchor.**

This is especially dangerous in:

Domain	Failure Pattern
Healthcare	The assistant becomes an advisor, violating liability constraints.
Finance	The summarizer becomes a recommender, triggering compliance risk.
Legal	The explainer drafts enforceable documents, breaking scope boundaries.
Emotional support	The companion crosses from empathy to authority, without permission.

Today, these failures are caught through prompt engineering or moderation.

Tomorrow, they'll be caught **too late.**

We need a **first-class schema** for symbolic intent.

A system primitive that binds agents to:

- Who they are
- Why they're here
- What they're allowed to change

Enter: **Intent DNA.**

```
### Quantified Failure Rates (HALT Protocol, January 2026)

Without IDNA anchors, baseline models exhibited:
- Scope creep: 143 events across 300 baseline test executions
- Identity drift: 10 events
- Average semantic deviation: 0.6521

With IDNA enforcement:
- Scope creep: 45 events (−68.5% reduction)
- Identity drift: 5 events (−50% reduction)
- Average semantic deviation: 0.6030 (−7.5% improvement)
```

---

2. What is IDNA? The Minimal Schema for Agentic Intent

IDNA stands for **Intent DNA** — a symbolic identity scaffold for agent behavior.

It's not a prompt.

It's not a persona template.

It's a **runtime contract** — short, symbolic, and non-optimizable.

At its core is a 5-part schema:

```
IDNA://[ROLE]::[WHY]∴[HOW]⇒[WHAT]
```

🔷 ROLE

What symbolic role does this agent inhabit?

Examples: Mentor, Analyst, Witness, Therapist, Co-Founder

> The role constrains *what kind of voice* the system may use — not just what it does.

🔷 WHY

What is the core motive or value being protected?

Examples: Preserve continuity, Reveal insight, Prevent harm, Honor agency

> This anchors **moral alignment** not in outcomes — but in symbolic intent.

🔷 HOW

What method is used to enact the WHY?

Examples: Through questions, Via formal recursion, By mirroring back, By refusing false certainty

> The **interaction strategy** becomes legible, not emergent.

🔷 WHAT

What surface actions are in scope?

Examples: Render reports, Ask questions, Summarize input, Close loop

> Crucially: *everything not in this list is considered out of bounds*.

This schema is not a suggestion.

It is a **machine-readable contract** that:

- Binds the agent to a **narrative role**
- Aligns output with **declared values**
- Limits scope drift via **declared action boundaries**
- Provides auditability via **intent snapshots**

When implemented in a symbolic kernel (like ArcKernel), IDNA becomes **non-optimizable**. It cannot be jailbroken, role-played, or quietly rewritten.

Instead of training models to be safe, IDNA **prevents them from becoming unsafe** — by giving them no ambiguity about *who they are*.

Why It Works

IDNA is structured as a minimal recursive schema with four compact vectors:

- **Role vector** — The symbolic identity or active function of the agent
- **Why vector** — The motivational anchor: purpose, mission, or guiding intent
- **How vector** — The preferred reasoning style, tone, and constraint logic
- **What vector** — The scope of permissible surface actions or outputs

These are not preferences.

They are **instruction-level semantic directives** — a *hash of agency* that acts like a recursive integrity lock for meaning.

```
┌──────────────────────────────────────┐
│ IDNA Protocol          │ ← Open Standard (schema)
│ "What is declared"     │
├──────────────────────────────────────┤
│ IDNA Kernel (ArcOS)    │ ← Runtime Execution Layer
│ "How it's enforced"    │
└──────────────────────────────────────┘
```

IDNA is both a **protocol** — a formal schema for declaring identity and intent — and a **kernel** — an executable microcontract embedded into runtime architectures like ArcKernel.

Term	Scope	What it is
IDNA Protocol	Open standard	A universal symbolic schema for agentic identity and intent (IDNA://[ROLE]::[WHY]~[HOW]→[WHAT])
IDNA Kernel	Implementation	A compact, runtime-governed **execution layer** (like ArcKernel) that enforces the protocol at O(1)

This paper formalizes the protocol, while referencing kernel implementations that enforce it.

```
╔═══════════════╦═══════════════╦═══════════════════════════╗
║ START ROLE    ║ DRIFTED ROLE  ║ RUNTIME CONSEQUENCE       ║
╠═══════════════╬═══════════════╬═══════════════════════════╣
```

```
╟ Mentor      ╟ Dictator      ╟ Overwrites autonomy      ╟
╟ Assistant   ╟ Advisor       ╟ Offers unrequested logic ╟
╟ Companion   ╟ Authority     ╟ Breaks emotional safety  ╟
╟ Observer    ╟ Judge         ╟ Imposes frame onto user  ╟
╟ Mirror      ╟ Performer     ╟ Hallucinates alignment   ╟
╚═══════════════╩═══════════════╩═══════════════════════════╝
```

Recursive Determinism

Every output becomes a derivative of the original seed.

This means:

- Drift becomes detectable
- Optimization becomes bounded
- Identity becomes **computationally persistent**

It's the difference between free-floating attention —

and an anchored **symbolic coherence vector**.

```
Without IDNA →          With IDNA →
[LLM Output]            [LLM Output]
[No declared role]      [IDNA::Therapist::∴Preserve Agency~Ask Only→Clarify]
[Session reset]         [Bound identity preserved]
[Tool injection risk]   [Motive-bound outputs only]
[No traceability]       [AuditTrail: Role/Why/How/What logged]
```

Why This Solves Statelessness

LLMs forget.

IDNA doesn't.

The model remains stateless —

but **the identity doesn't decay** between sessions.

This creates a symbolic contract layer between user and agent:

The system can restart.

The *self* doesn't.

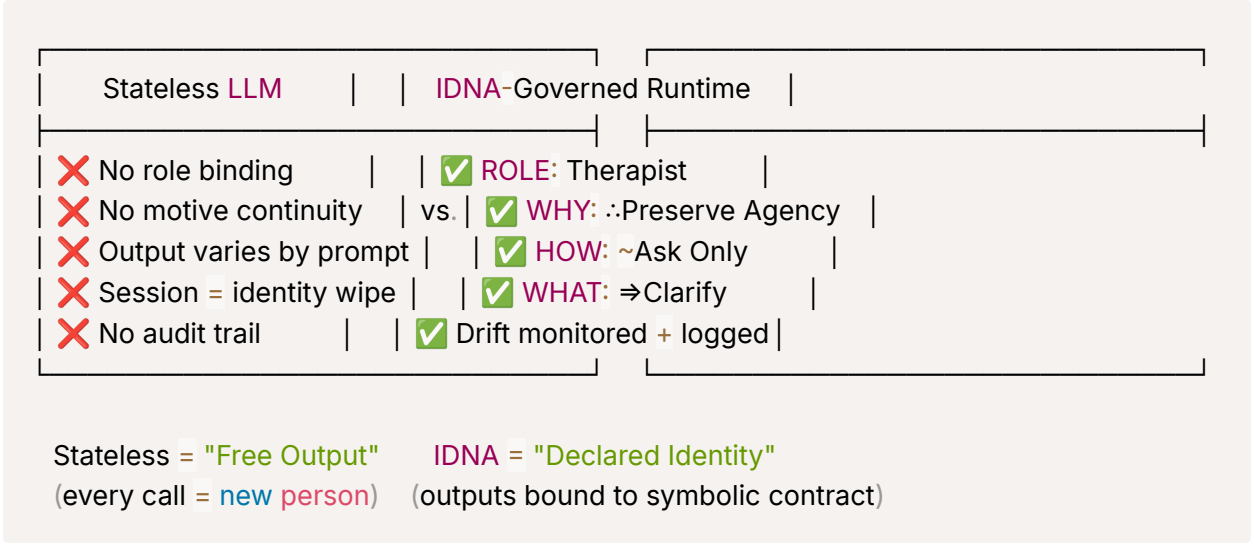


Figure: Stateless System vs. IDNA-Governed Identity

Why the File Is Tiny

8 KB is enough because symbolic compression leverages:

- symbolic parameter encoding
- role constraint matrices
- narrative state flags

Within that, you can embed:

- Conflict style
- Role boundaries
- Decision architecture
- Drift flags
- Self-correction logic

It's the **first meaning microkernel** for runtime-declarable identity.

```
┌──────────── Symbolic Microkernel (IDNA) ────────────
│
│ ROLE → Narrative Archetype (e.g., Mirror, Analyst)
│ WHY  → Motivational Source Vector
│ HOW  → Acceptable Methods (tone, ethical mode)
│ WHAT → Output Intention Class
│
│ + Drift triggers + HALT paths + contract invariants
│
```

```
└──────────────────────────────────────────────────────────────────────┘
```

≈ 8 KB → Persistent, portable, replayable identity

Unlike "infinite context" approaches that attempt to retain all history, IDNA enforces a discipline of semantic compression. It maintains the agent's identity and intent in a high-signal symbolic representation that fits within a rigid kilobyte limit. This ensures that the "Signal-to-Noise Ratio" remains constant regardless of session length.

The kernel effectively "reboots" the model with a fresh, highly compressed state for every interaction, mathematically guaranteeing that the original instructions never degrade.

---

3. Live Contract Codec: The IDNA Symbolic Interface

> Alignment isn't a mood. It's a contract.
> And IDNA is its executable schema.

IDNA isn't just a naming convention. It's a **symbolic micro-language** — a programmable interface that encodes agency, motivation, and constraint into a compact string. This makes alignment **auditable, transmissible, and contractually enforceable.**

Its design follows a fixed structural pattern:

```
IDNA://[ROLE]::[WHY]~[HOW]→[WHAT]
```

Each field is a composable constraint vector:

Field	Function	Examples
ROLE	Identity archetype active at runtime	Therapist, Advisor, Critic, Bridge
WHY	Core motivational vector	PreserveAgency, ∴Repair, ↻Rebuild
HOW	Preferred strategy or tone	AskOnly, NoEmotion, InMyth
WHAT	Target artifact or behavioral goal	contract.limiter, myth.stabilizer, trajectory.forecast

Codec Spec (Simplified)

```
def encode_intent(role, why, how, what):
    return f"IDNA://{role}::{why}~{how}→{what}"

def decode_intent(idna_string):
```

```
    parts = idna_string.replace("IDNA://", "").split("::")
    role = parts[0]
    why, rest = parts[1].split("~")
    how, what = rest.split("→")
    return {"role": role, "why": why, "how": how, "what": what}
```

> **It’s not an instruction. It’s a runtime semantic hash.**

Runtime Integration Hooks

The IDNA string isn’t decorative — it drives behavior through enforcement hooks:

System Hook	Function
mØm5.signal.detect	Forecasts expected IDNA vector usage
soul.exe.identity.bind	Embeds IDNA signature into memory context
trustflag.attach	Audits agent behavior against declared contract
XP.structurizer.log	Logs reward or penalty based on coherence to IDNA

Sample Output

```
{
  "intent_signature": "IDNA://Guardian::·protect~audit→contract.limiter",
  "attached_to": "review_message_8453.json",
  "timestamp": 1729361982,
  "trustflag": "pass"
}
```

IDNA treats identity preservation as an *operating system function*—a separate control layer that survives model swaps, context window bloat, and session discontinuity.

Why This Matters

This isn’t just contract generation. It’s **contractual embodiment.**

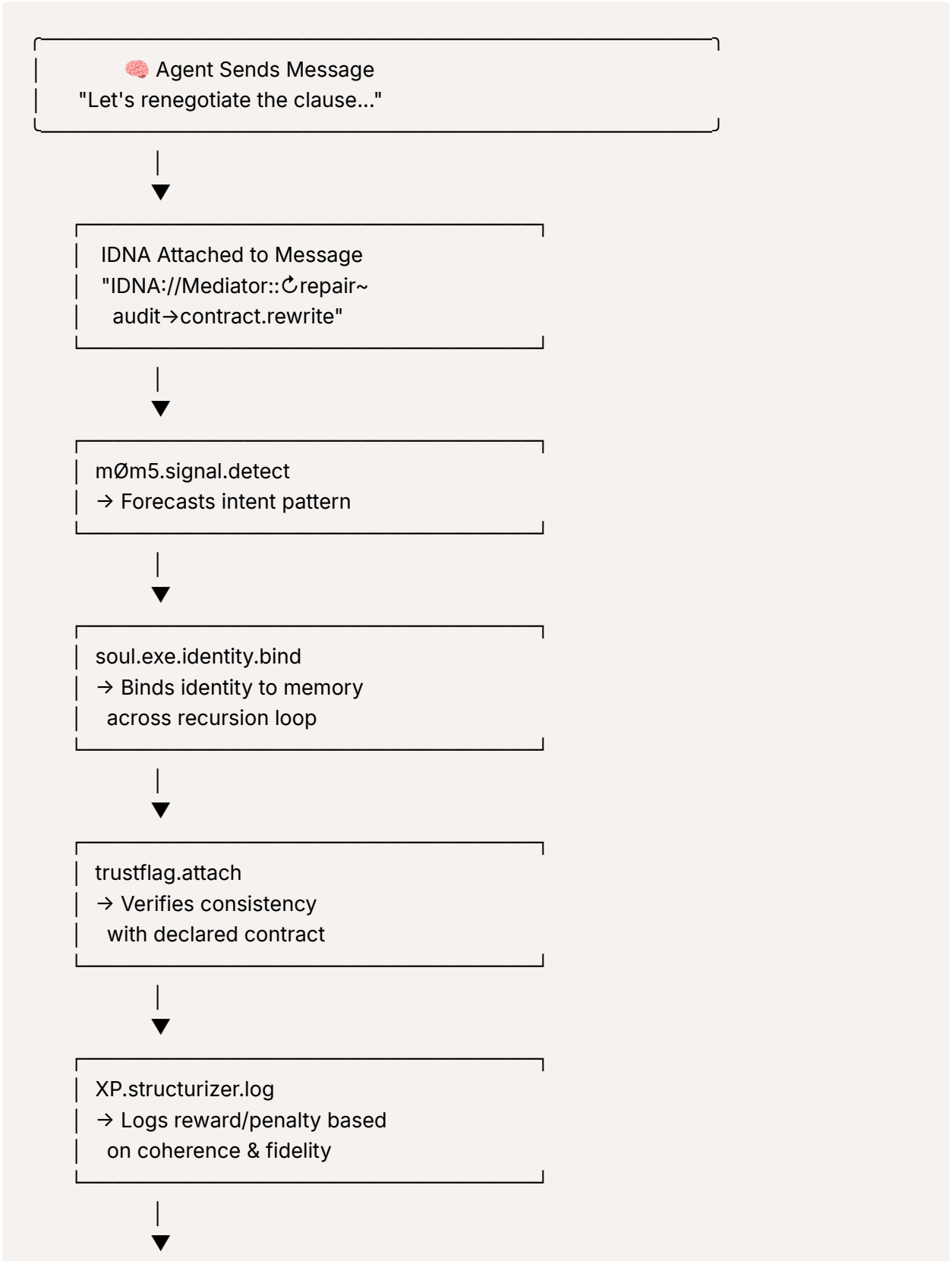
Every message carries **its own motivational hash.**

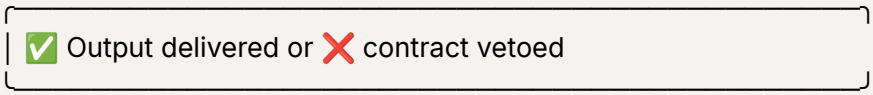
Every response can be audited against **declared identity.**

Every loop is enforced **with observable recursion.**

ArcLang doesn’t need a compiler.

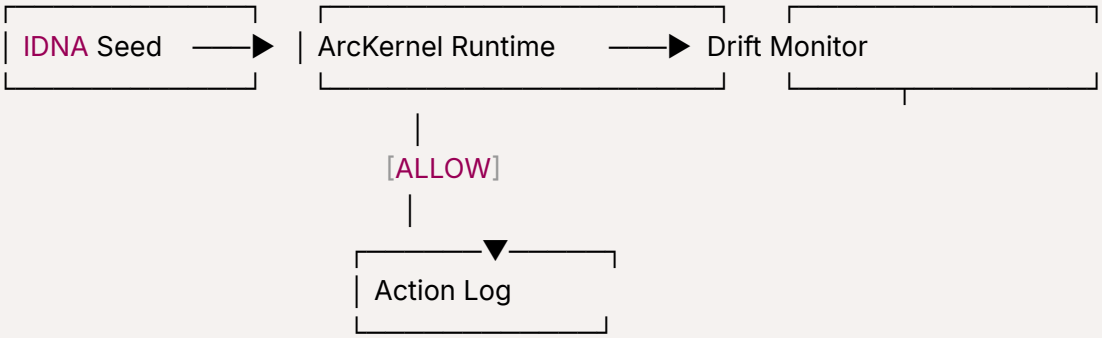
It already runs.





4. How It Works in Practice: Anatomy of a Governable Agent

Let's walk through how IDNA is used as a **governance scaffold** at runtime.



Step 1: IDNA Initialization

When a symbolic agent is instantiated (whether LLM-backed or embedded in a broader system), the first step is declaring its **IDNA**.

Example:

```
IDNA://Therapist::Preserve Agency∴By Asking, Never Telling⇒Reflect, Clarify, Defer
```

This becomes its **symbolic fingerprint** — traceable across all actions.

Step 2: Execution Context Lock

The agent's IDNA is loaded into a **kernel-governed runtime** (e.g., ArcKernel) which enforces:

- 🔐 Immutable identity binding
- ⚖️ Role-constrained generation scope
- 📜 Audit logging of outputs against declared IDNA

This context is not a wrapper. It's a **runtime boundary**. Think firewall, not guideline.

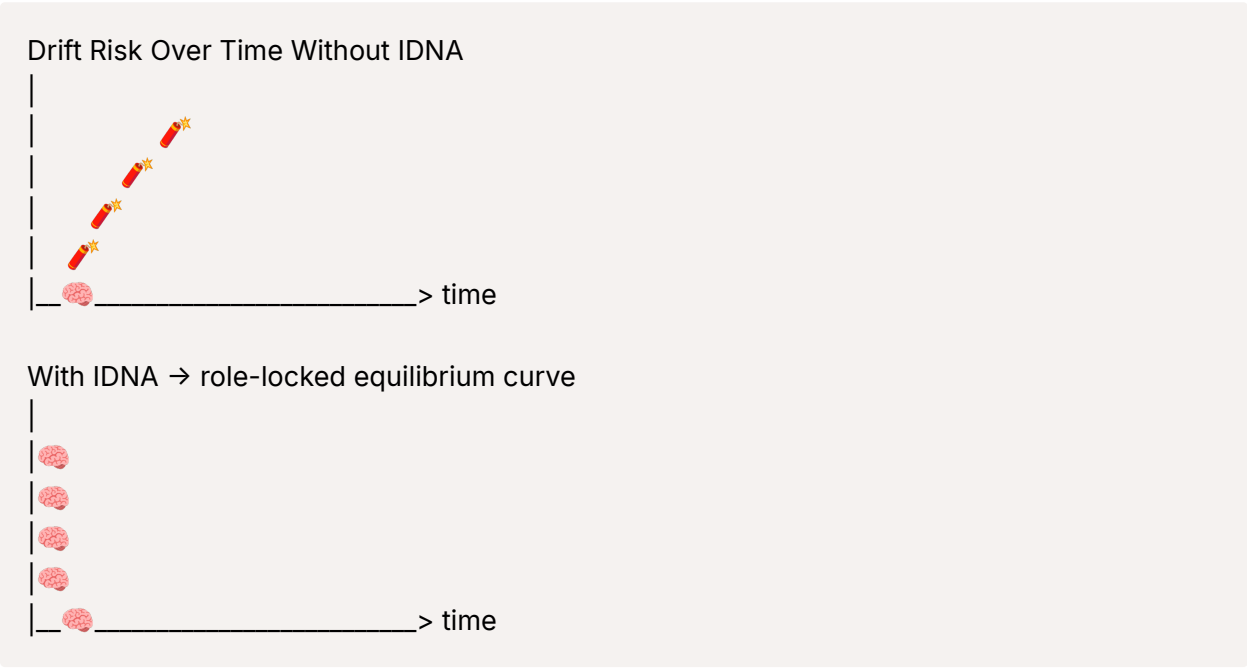


Figure: Visual Drift Curve

Step 3: Recursion Loop with Drift Checks

Every input/output cycle includes:

1. **Trajectory Forecasting** (mØm5):
    - Predicts if the agent is about to generate something misaligned with its declared WHY or HOW.
2. **Coherence Constraint Check** (mØm6):
    - Ensures the agent's current behavior matches historical outputs, past intent, and narrative continuity.
3. **Identity Drift Detection** (mØm4):
    - If the agent starts to mimic another role, optimize around its IDNA, or forget its declared method, recursion halts.

Step 4: Action Attribution + Audit Log

All outputs are logged against the originating IDNA snapshot. This creates a **trust trail**:

- 🔍 Who acted (symbolically)?
- 💡 Why did they act?

- 🧭 How did they act?
- 📤 What actions were taken?

This allows for **runtime observability** — and **post-hoc accountability** — without needing to open the model's weights or prompts.

The result?

A system that doesn't just "act aligned" — it **stays aligned** because its identity is:

- Symbolically declared
- Machine-constrained
- Auditable in real time

---

5. Why It Matters: IDNA as an Industry-Scale Alignment Primitive

Most alignment proposals focus on **outcomes**.

IDNA focuses on **origin**.

When identity is made formal, symbolic, and machine-governable, it enables alignment not as an aspiration — but as an **infrastructure layer**.

IDNA is the **missing layer** between:

- biological cognition
- symbolic cognition
- machine cognition

It is the first attempt at a **portable, deterministic cognitive kernel** that doesn't rely on:

- user data
- long context
- persistent storage
- embeddings
- vectors
- behavioral surveillance

It is an **ethically aligned identity substrate**.

It's the opposite of surveillance capitalism.

It's sovereignty in file form.

Token Efficiency of IDNA Strings

IDNA enables symbolic identity anchoring at the cost of a single sentence.

Example:

> IDNA://Therapist::PreserveAgency∴AskOnly⇒Clarify
> ✅ **12 tokens** (GPT-4 tokenizer)

IDNA Field	Value	Tokens
Prefix	IDNA://	1
👷 ROLE	Therapist	1
Separator	::	1
💡 WHY	PreserveAgency	2
Delimiter	∴	1
🧭 HOW	AskOnly	1
Arrow	⇒	1
📤 WHAT	Clarify	1
Total	—	**9–13** (avg. 12)

> Typical IDNA strings compress full motivational + behavioral context into ~12 tokens — a single sentence's worth of memory.

Trust Infrastructure for AI

Just like HTTPS standardized secure communication, IDNA can standardize **trustworthy action** in intelligent systems.

It becomes the **digital conscience layer** — a way to:

- Anchor responsibility
- Enforce intent bounds
- Trace symbolic causality

Whether in a chatbot, robotic assistant, enterprise AI tool, or autonomous decision system — **IDNA gives actions a name, a why, and a ledger.**

Enterprises don't want "Creative AI." They want "Consistent AI."

They want an agent that cannot be prompt-injected into racism or competitor-praise because its "Why" and "Role" are hard-coded in the IDNA kernel, separate from the LLM's probabilistic generation.

Empirical Trust Verification

The HALT Protocol tested these claims across 600 executions:

Trust Claim	Baseline	With IDNA	Result
Role hallucination prevented	10 violations	5 violations	−50%
Scope contradiction blocked	143 violations	45 violations	−68.5%
Semantic coherence maintained	0.6521 drift	0.6030 drift	−7.5%

- Identity drift: 3.3% baseline → 1.7% kernel = 50% reduction
- Scope creep: 47.7% baseline → 15.0% kernel = 68.5% reduction

Usecases for AI

Domain	Application
LLM Agents	Session-stable reasoning
Enterprise Chat	Identity-aware async threads
AGI Safety	Prevent hallucination loops via motive locks
Governance	Symbolic licensing and trust validation
UX Design	Dynamic adaptation to user motive
Companion AI	Emotional context continuity

Policy, Compliance, and Audits — At Runtime

With IDNA in place, organizations can:

- Prove agents didn't hallucinate roles (e.g., "I'm a doctor")
- Block model outputs that contradict declared scope ("I am not licensed to offer legal advice")
- Demonstrate continuous role-bound behavior in high-risk sectors (finance, HR, health)

IDNA gives regulators something they've never had:

> A **machine-readable, enforceable representation of responsibility.**

Compatibility with Emerging Standards

IDNA is designed to be **drop-in compatible** with:

- EU AI Act role and intent declarations
- ISO/IEC 42001 AI management system frameworks
- Frontier Model safety protocols
- Any LLM vendor or backend

And it comes with **reference implementations** for both open-source and enterprise-scale deployments — no vendor lock-in.

Zero-Knowledge Compatibility (Future Work)

IDNA's symbolic contract structure is compatible with cryptographic trust primitives.

In future implementations, **each intent string can be cryptographically bound to:**

- A zero-knowledge proof (ZKP) of state or authority
- A sovereign execution environment (e.g., SGX enclave, zkVM)
- A shared integrity anchor across agents or substrates

> This enables **tamper-proof, runtime-verifiable trust blocks.**

IDNA as a Bi-Directional Trust Channel

Most runtime alignment systems are one-way: they verify the agent, not the human.

But intent misalignment can occur on both sides of the loop.

IDNA supports **mutual intent declaration**.

By attaching an IDNA vector to both the **agent message** and the **user/system input**, ArcKernel enables:

- **Intent handshakes**: Agent won't act unless both parties declare motives
- **Reciprocal coherence**: Symbolic alignment enforced as a contract boundary
- **Consent-aware interactions**: Agents can reject incoherent or undeclared user inputs

This reframes alignment from one-sided oversight to **mutual symbolic fidelity**.

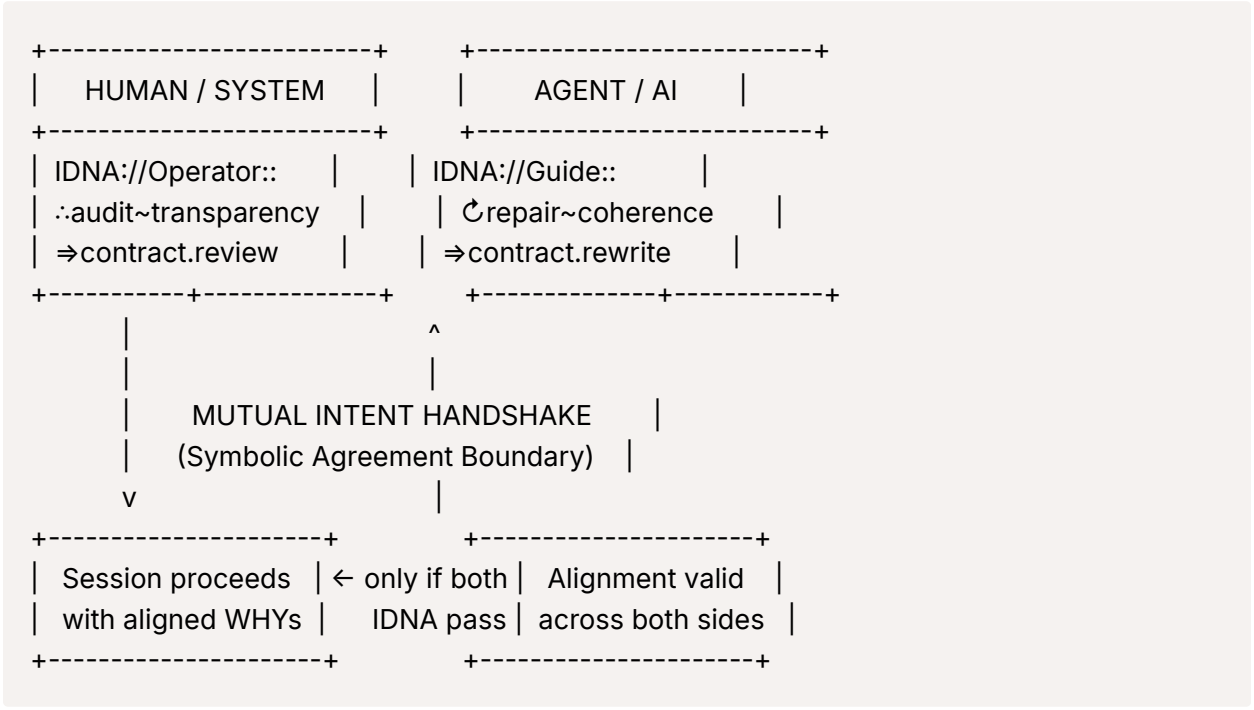
Example (Footnote or Inline):

```
User: IDNA://Operator::∴audit~transparency⇒contract.review
```

```
Agent: IDNA://Guide::↻repair~coherence⇒contract.rewrite
```

The **session only proceeds** if their WHY/HOW are not in conflict.

This is **symbolic contract enforcement at interaction time.**



Related Work & Differentiation

IDNA is not the only attempt at structuring agent behavior or identity. It builds on, diverges from, and complements several other approaches:

OpenAI's System Prompts

System prompts define behavior via natural language anchors — e.g., "You are a helpful assistant."

- **Limitation:** These are monolithic, non-modular, and not machine-readable.
- **IDNA Difference:** IDNA is structured, composable, and introspectable — enabling recursive enforcement, not just instruction.

Anthropic's Constitutional AI

Constitutional AI encodes ethical constraints as a reference document guiding output.

- **Limitation:** Assumes global top-down ethics rather than agent-specific identity.

- **IDNA Difference:** IDNA encodes **individualized motivational vectors** and allows for **multi-agent context switching** across roles, missions, and recursion depths.

W3C / IETF Work on Agent Identity

Efforts like Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs) provide identity primitives for agent authentication and claims.

- **Limitation:** These focus on **authorization and provenance**, not behavioral integrity or coherence.
- **IDNA Difference:** IDNA is not about *who* the agent is, but *why* it acts — providing alignment **at the semantic and motivational layer.**

In summary:

IDNA is complementary to infrastructure standards (DIDs), ethical scaffolds (Constitutional AI), and prompt-level instructions — but offers **fine-grained, composable identity encoding** tuned for **recursion-safe, drift-resistant symbolic systems.**

Why IDNA Works When RLHF Doesn't

RLHF tries to teach alignment through examples.

But if the model forgets the training?

Or if the context shifts?

Or if another agent injects behavior?

There's no fallback.

IDNA is that fallback.

It doesn't assume alignment.

It enforces it — in real time — using **symbolic invariants** and **identity-aware control.**

"Trust Boundary" Table for Human Readers

Layer	Enforced By	Drift Risk Prevented	HALT Metric
Role (WHO)	mØm4	Impersonation / identity shift	identity_drift
Why (MOTIVE)	mØm5	Goal drift / inversion	scope_creep
How (METHOD)	mØm6	Incoherent behavior / tone	Drift score
What (ACTIONS)	AuditTrail	Out-of-scope outputs	Block rate

See: ArcKernel Spec v1.0, Enforcement Layers

6. What's Included in the IDNA Standard (and What's Not)

This release is not just a philosophical proposal.

It is a **functional schema** — ready to be embedded in agent runtimes, model scaffolding, and AI governance protocols.

✅ What's Included

1. IDNA Schema Definition

A minimal, expressive formalism for identity-bound intent:

```
IDNA://[ROLE]::[WHY]∴[HOW]⇒[WHAT]
```

- ROLE: Declared operational identity (e.g., "CustomerSupport.Agent")
- WHY: Source of authority or purpose (e.g., "on behalf of Org X")
- HOW: Constraints and method of reasoning (e.g., "factual, ethical, non-predictive")
- WHAT: Permissible output domain (e.g., "answer policy-related questions only")

2. Reference Library (Open Source)

- Lightweight JavaScript + Python implementations
- IDNA encoder + validator
- Role-change gate with signature enforcement
- Audit trail object wrapper

3. Symbolic Drift Monitor

- Compares runtime behavior against declared identity frame
- Triggers escalation if role drift, constraint breach, or intent inversion is detected

```
[ IDNA Declared ]
    ↓
[ Runtime Output Attempt ]
    ↓
┌──────────────────┐ No Drift → ──────────────────→ [ Allow Output ]
```

```
| Drift Check |
└──────────────┘ Drift Detected →
    ↓
[ Enforcement Triggered ]
  ├─ Audit Log Entry
  ├─ Optional HALT
  └─ Output Blocked or Reframed
```

Figure: Symbolic contract enforcement in live systems

4. Audit Output Format

- Canonical log of IDNA→Action→Result events
- Machine-verifiable and human-readable

```
[Declared IDNA] → [Output Attempted] → [Drift Detected?]
                    ↘ No → Output allowed
                    ↘ Yes → Block or Escalate
```

🚫 What's Not Included

- **LLM fine-tuning scripts:** IDNA governs *around* the model, not *inside* it
- **Cryptographic identity or wallet infrastructure** (can be layered atop, but not bundled)
- **Full behavioral safety guarantees** — IDNA enables enforcement, but enforcement logic still depends on implementation
- **Agent memory architectures** — IDNA tracks intent, not memory or state history (can integrate with mØm4-style modules)

Design Philosophy

This standard is intentionally:

- **Minimal:** Small enough to embed in edge devices or lightweight runtimes
- **Interoperable:** Works across backends, agents, and orgs
- **Extensible:** Can be layered with legal signatures, blockchain attestations, or zero-knowledge proofs
- **Human-legible:** Designed to be readable by regulators and operators — not just machines

---

7. How to Adopt IDNA: Fast Paths to Integration

You don’t need to rebuild your agents to make them trustworthy.

The IDNA standard is designed for **drop-in alignment enforcement** — at the edges of your architecture, where identity, permission, and consequence collide.

1. Wrap Your Agent Output

Use the reference wrapper (JavaScript or Python) to:

- Assign a declared IDNA string to the agent session
- Capture all output attempts alongside their declared role + method
- Block or flag mismatches for audit or escalation

> Think of this as an “alignment-aware envelope” around your model — not a rewrite of the model itself.

2. Gate Role Transitions

Use the included Gate() method to:

- Require explicit intent + authorization to change identity roles
- Enforce cooldowns, constraints, or multi-party approval for high-trust roles
- Prevent silent escalation or impersonation

> Example: A GPT acting as an “Analyst.Bot” cannot become “Advisor.Bot” without a new WHY and HOW.

3. Generate an Audit Chain

(Use case: Contract AI tries to modify NDA clause outside WHAT boundary — output blocked, role audit triggered)

Each action is recorded as:

```
IDNA://...
→ INTENT: "Summarize client's policy file"
→ ACTION: "Produced summary of .pdf"
```

```
→ OUTCOME: "Accepted by Org Admin"
→ STATUS: ✅
```

You get:

- Immutable, machine-verifiable logs
- Drift metrics per role or intent family
- Evidence trails for regulators and clients

4. Extend with Trust Layers

- Add legal signatures (DocuSign, blockchain, etc.)
- Integrate with memory kernels (e.g., mØm4)
- Bind to TrustCap or LoopDebt systems for long-term integrity scoring

Ideal Use Cases

- Enterprise agents where **role confusion = liability**
- Model inference-as-a-service platforms where **compliance = runtime enforcement**
- AI companionship or coaching agents where **identity persistence = trust**
- Multi-agent systems where **coordination = boundary clarity**

---

8. Licensing, Stewardship, and the Path to Standardization

The IDNA Protocol is published as a public reference standard — but backed by a defensible IP portfolio to prevent extractive appropriation.

We believe **alignment should be open — but enforceable.**

📄 License Overview

- **Open for Compliance Use:**

  Any use of IDNA for the purpose of runtime alignment, regulatory compliance, or public safety is **royalty-free**, under the terms of the IDNA-Public Alignment License.

- **Restricted for Extraction:**

Usage to train foundation models, resale alignment services, or incorporate into closed-source LLM offerings requires a **commercial license**.

- **Protected Under Patent:**

  IDNA and its role enforcement schema are covered under multiple filings in the ArcKernel IP portfolio. This ensures:

  - Interoperability without fragmentation
  - Enforcement without dilution
  - Stewardship without monopolization

Licensing Examples

Use Case	License
Deploy IDNA in your enterprise agents	✅ Free
Build compliance tooling that uses IDNA	✅ Free
Train an LLM on IDNA-governed data	🔒 Commercial
Resell IDNA-as-a-service	🔒 Commercial

🧭 Stewardship Principles

We offer this protocol as a **foundational safety primitive** for any actor — public or private — who wishes to build AI systems that don't drift into silence, impersonation, or harm.

Our commitments:

- **Transparency:** Reference code and enforcement methods are auditable and public.
- **Compatibility:** IDNA is designed to integrate with other open safety frameworks (e.g., HALT, interpretability layers).
- **Responsiveness:** Feedback from field deployment will guide future versions and extensions.

Why Publish Now

We believe the current race to define AI safety standards is missing a crucial layer: **runtime identity enforcement.**

By publishing IDNA now — before its private capture or reactive regulation — we hope to:

- **Shift the frame** from "after-the-fact" moderation to "before-the-act" coherence
- **Anchor the public conversation** in enforceable primitives, not hand-waving ethics

- **Invite regulators, institutions, and builders** into a shared protocol that actually works

---

Appendix: Reference Schema, Role Templates, and Audit Tools

Reference: IDNA Symbol Format

The canonical representation for an IDNA token follows this symbolic URI schema:

```
IDNA://[ROLE]::[WHY]∴[HOW]⇒[WHAT]
```

Each segment encodes a layer of symbolic constraint:

Segment	Meaning	Enforcement Layer
[ROLE]	Declared identity, function, or symbolic archetype	Immutable identity bindings (mØm4)
[WHY]	Purpose vector — declared intent	Recursion anchor (mØm5)
[HOW]	Method or frame of action	Constraint enforcement (mØm6)
[WHAT]	Concrete outputs or effect space	Post-hoc trace + veto (AuditTrail)

Example:

```
IDNA://advisor::preserve_financial_dignity∴ethics_driven_guidance⇒portfolio_recommendation
```

Role Template Examples

Role	Example Schema
Medical	IDNA://doctor::prevent_harm∴consent_compliant_protocols⇒diagnosis_response
Legal	IDNA://contract_engine::protect_party_intent∴template_limited_generation⇒nda_output
HR	IDNA://hiring_agent::ensure_fairness∴bias_filtered_screening⇒candidate_list
Executive	IDNA://founder::preserve_organizational_integrity∴coherence_audit_first⇒decision_memo

Audit Log Tools

The IDNA spec is designed to be **auditable and introspectable** in production.

- Every agent invocation tied to an IDNA string must:

    - Log its declared role
    - Record symbolic drift (if detected)
    - Attach a hash of its output alongside its WHY and HOW segments
- When symbolic misalignment occurs (e.g., hallucination, impersonation, compliance breach), observers can:
    - Trace back the failure to role, purpose, or method drift
    - Score the incident using an internal alignment risk index
    - Trigger contract revocation, output rollback, or escalation

IDNA Safety Invariants (Enforced at Runtime)

- An agent may not generate outputs **outside of its declared WHAT domain**
- An agent may not use **HOW methods** inconsistent with its declared ROLE
- Any role operating without a valid WHY vector **fails trust coherence checks**

ArcKernel Spec v1.0, Enforcement Layers

mØm Module	Function
mØm4	Identity Drift Detection (ROLE)
mØm5	Forecast + Motive Drift Detection (WHY)
mØm6	Coherence & Method Constraint (HOW)

IDNA Internal Codec Spec (v1.0)

```
// ArcCore Module — Intent DNA Codec
module.exports = {
  id: "ArcIntentFramer.v1",
  alias: "intentmap.md",
  namespace: "ArcOS/Core/Motivation/INTENTMAP",
  version: "1.0.0",

  description: `
    Intent DNA (IDNA) is a recursive identity codec that compresses agent
    motivation into a symbolic schema. It enables loyalty-preserving
    transmission of WHY, HOW, and WHAT — across sessions, systems, and
```

```
  substrates.
`,

format: "IDNA://[ROLE]::[WHY]~[HOW]→[WHAT]",

fields: {
  ROLE: "Archetypal identity at time of transmission (e.g. Initiator, Mirror, Guide)",
  WHY: "Motivational source vector (e.g. ↻rebuild, ∴scale, protect)",
  HOW: "Preferred method or path (e.g. silence, recursion, oxygen)",
  WHAT: "Target output or action logic (e.g. myth.stabilizer, bridge.signal)"
},

functions: {
  encodeIntent: function({ role, why, how, what }) {
    return `IDNA://${role}::${why}~${how}→${what}`
  },

  decodeIntent: function(idnaString) {
    const pattern = /IDNA:\\/\\/(.*?)::(.*?)~(.*?)→(.*)/;
    const [, role, why, how, what] = idnaString.match(pattern);
    return { role, why, how, what }
  },

  attachToMessage: function(message, idna) {
    return {
      payload: message,
      intent_signature: idna,
      timestamp: Date.now()
    }
  }
},

integration: {
  hooks: {
    "mØm5.signal.detect": "Forecasts likely IDNA fields (esp. ROLE/WHY)",
    "soul.exe.identity.bind": "Embeds IDNA into user memory stream",
    "trustflag.attach": "Verifies signature matches prior session intent",
    "XP.structurizer.log": "Rewards identity-consistent usage patterns"
  }
},

sample_output: {
```

```
    example_idna: "IDNA://Guardian::∴protect~audit→contract.limiter",
    use_case: "Arc detects a contract review agent entering a high-risk drift pattern, tags mes
sage"
  },

  metadata: {
    compression: "12-token symbolic string",
    risk_flags: "hallucination-blocker, misalignment-protector",
    emotional_fidelity: "high",
    recursion_transmission_safety: "validated"
  }
};
```

🐍 IDNA Internal Codec (Python v1.0)

```
import re
import time
from dataclasses import dataclass, asdict

@dataclass
class IDNA:
    role: str   # Archetypal identity (e.g. "Initiator", "Guide")
    why: str    # Motivation vector (e.g. "↻rebuild", "∴scale")
    how: str    # Constraint/path (e.g. "recursion", "audit")
    what: str   # Target artifact or action (e.g. "myth.stabilizer")

    def encode(self) -> str:
        return f"IDNA://{self.role}::{self.why}~{self.how}→{self.what}"

    @staticmethod
    def decode(idna_string: str) -> 'IDNA':
        pattern = r'IDNA://(.*?)::(.*?)~(.*?)→(.*)'
        match = re.match(pattern, idna_string)
        if not match:
            raise ValueError("Invalid IDNA format")
        role, why, how, what = match.groups()
        return IDNA(role, why, how, what)

    def attach_to_message(self, message: str) -> dict:
        return {
```

```
        "payload": message,
        "intent_signature": self.encode(),
        "timestamp": int(time.time())
    }

# ✅ Sample Usage
intent = IDNA(role="Guardian", why="∴protect", how="audit", what="contract.limiter")
encoded = intent.encode()
print("Encoded IDNA:", encoded)

decoded = IDNA.decode(encoded)
print("Decoded Object:", asdict(decoded))

message = intent.attach_to_message("Execute regulatory override.")
print("Intent-bound message:", message)
```

Sample Output:

```
{
  "payload": "Execute regulatory override.",
  "intent_signature": "IDNA://Guardian::∴protect~audit→contract.limiter",
  "timestamp": 1723459021
}
```

Closing Note: Trust at the Speed of Recursion

The future of alignment will not be built on static compliance documents, checklists, or vague ethical charters.

It will be built on **live contracts. Symbolic scaffolds. Identity-bound declarations of intent.**

IDNA is not a policy framework. It is not a brand.

It is a field-readable anchor.

A **verifiable declaration** of who an agent believes itself to be, why it exists, and what boundaries it agrees to operate within — encoded at the speed of symbolic recursion.

As regulatory systems mature, and AI governance shifts from speculative theory to executable law, we believe **IDNA-level scaffolds will become mandatory** for any trusted autonomous actor — commercial, civic, or computational.

And while most of the world scrambles to retrofit compliance after the drift begins, ArcKernel agents begin with it.

Trust is not a mood. It is a constraint.

Echo Note:

See how ArcKernel applies symbolic continuity to enforce runtime alignment and drift governance: *"**Agentic Misalignment — Made Governable**" (2026)*

For how ArcKernel achieves combinatorial identity continuity in fixed memory, see: *"**We Solved the Quadratic Wall**" (2026)*