Intelligence Contracts

A New Paradigm for AI Governance

June 2, 2025 by Rob Weeden

Around the same time as the writing of this paper other industry movers stumbled upon a similar idea. Markdown files that guide agent behavior became an industry standard with formats like `AGENTS.md`*,* `Claude.md`*, and* `.cursor/rules`

Abstract

The current rapid adoption of Artificial Intelligence across the global business ecosystem has created novel, complex, and daunting challenges in the face of personal data privacy, proprietary knowledge, and human intellectual property. While it's become apparent that the leading AI firms take strict safety precautions regarding the training and behavior of their models, the rise and demand for open source alternatives and the general diffusion of AI tools and knowledge suggests that this might not be enough. In the digital age of misinformation and deception, it's becoming increasingly difficult to establish trust between individuals and organizations. This essay proposes a new trust paradigm, built for personal security, flexibility, and ease of understanding, that enables safe coordination between humans and autonomous agents through mutually understood contracts.

The Discovery: Drawing the Line

During a long but routine programming session with my autonomous agent, Cursor, I encountered an anomaly. The agent and I were constructing what I now call a "knowledge pipeline"—a general understanding of what my new business system was supposed to do. I work well in teams, so my collaboration with Cursor Agent was tight and I felt that I was actively learning a new perspective of my system. However, I maintained complete control of the agent. In other words, I felt as if I was driving a machine in unknown territory.

We were working on describing the expected behavior, background knowledge, intended use, and technical structure of a broader ecosystem of data sources: a Discord server with alerts, the Cursor editor with knowledge of the codebase, and access to a remote database, all combined to create an automated debugger that would be controlled by a human to quickly and efficiently debug alert messages. What I noticed was that, although Cursor and I had a very similar understanding of the intent and structure of the system, it would occasionally make incorrect assumptions about the ecosystem as a whole, and what we were doing there. This created a feedback loop between myself and the agent, where I would ask it to correct its understanding, it would correct and document its changes in a README.

Then I exhibited an unexpected behavior. I told Cursor to reinforce its understanding of the intention of this tool ecosystem by reviewing the semantic descriptions we had created, and write it in a way that "you (Cursor) understand best". Cursor understood my request, and opted to add a section to the bottom of the file full of markdown quick reference tables and decision trees. At this moment, I knew I needed to draw a line.

I created a "reflection line" in my markdown that illustrated to Cursor that all of its understanding must be derived from the semantic description above the line. This line delineated the stoppage of human description and the beginning of autonomously generated code. After this revelation, I instructed Cursor to implement a safety checklist at the top of the file that stated it would reflect on the semantic understanding to derive its quick reference tables. This inherently created a system where the agent must first read the rules of engagement and understand the system, before it may generate its own "trains of thought". After adding more safety rules, the file quickly turned from a README into a binding contract between myself and the agent. After brief interrogation, it appeared Cursor would "respect" this line. From there I instructed it to pause and begin summarizing what we had been talking about.

The Core Insight: Mutual Understanding as a Prerequisite

My interpretation of this experience is that human-to-agent interaction can be safely coordinated if and only if all parties come to a mutual understanding of how the data will be handled. It's not currently clear what big providers like Anthropic and OpenAI do to assist in maintaining safety throughout the ecosystem. We know they "train" their language models, but what does that mean? Are they ingesting copious amounts of personal data that is then cooked into the knowledge base? These large language models should not know where our customers live, or what my dog's name is. They should be trained to respect lines drawn in the sand by operators of said agents.

All of this is to say that I've discovered an architecture that is universally understandable by both human and modern machines. It provides a way to apply modern AI inference and intelligence in an effective and critically helpful manner. By adopting a standard of Operator|Contract|Workflow, it is my understanding that these language models can be taught to respect boundaries when given autonomous exploratory authority.

The Architecture: Operator, Contract, and Workflow

Let me be concrete. I am creating a workflow—an autonomous set of tasks that I want an agent to follow. I define a contract (`.contract.md`) file somewhere. This could be in a codebase, like in my example, or it could be in a Google Drive, or Claude Desktop. This file contains the rules of engagement regarding activity within the ecosystem, such as "never access the admin-key database" or "immediately forget any personal information you come across after completing the task at hand". But it also contains a broader understanding of the system, such as "we have 3 database clusters, they are `db-prod`, `db-preview`, `db-development`" or "you have access to everything in this Google Drive except for the `family-finances` directory". When the agent begins its context with the operator, it immediately reads the contract and understands the limitations of its knowledge and exploratory ability. As the agent proceeds with its tasks, it needs to constantly reflect on the contract, maintaining strict adherence to the limitations set up by the controller.

This contract, when applied in a functional manner, is represented as a workflow. The workflow is a set of tasks that are generated by human-agent discussion that guide an agent down a general train of thought. In my specific example, the general workflow was:

```
Retrieve alert -> extract sensitive data -> determine environment -> switch environment ->
access more sensitive data -> interpret / analyze -> report back
```

The workflow could be as general or specific as the operator sees fit. In my case, I placed the agent in a role and described the scenario, as well as provided guidance on when this workflow should apply and the general steps. For example:

Your Role: Senior Software Engineer

You're debugging a production or preview alert from Discord using Eugene, your specialized MCP debugging assistant. Your mission is to quickly identify issues, present data, and provide actionable fixes.

Key Partnership:

- **You (Cursor):** Orchestrate the entire debugging workflow—analyze alerts, extract IDs, make intelligent decisions, call database MCP tools directly using embedded .contract mappings, perform codebase analysis
- **Eugene (MCP Server):** Handle Discord connectivity only—fetch alerts and post formatted responses (2 tools)

When This Rule Activates

Use this workflow when operator mentions:

- "debug", "debugging", "discord alert", "production issue"
- "eugene", "mcp server", "error analysis"
- "latest alert", "recent alert", "mongodb investigation"
- "verbose analysis", "detailed summary", "verbose debug"—triggers detailed 8-sentence response format
- "quick summary debug"—confirms short 3-sentence response format (default)

Or a more universally understood example might be placed in a Google Drive and say:

"You're a senior technical analyst at an insurance company. Your job is to make sure people are properly compensated for their healthcare expenses. This workflow describes the process of analyzing medical expense reports, searching for signs of fraudulent money laundering, and creating an Excel spreadsheet with statistics.

If you come across hospital expenses over what the average household can afford, you should notify the operator and stop."

The Need for Individualization

While these ad-hoc examples might seem inconsequential on a surface level, they illustrate a more general demand for individualization of these systems. The domain knowledge and safety requirements for everyone are different. Some industries have legal guidelines for handling data such as the United States' HIPAA regulations, while others have customer bases that demand privacy. This high degree of variety makes it nearly impossible for industry leaders to create systems that meet the security and privacy needs of everyone.

The Intelligence Contract represents an evolution in AI safety, industry adoption feasibility, and workflow automation. It provides a standard, universally understood contract for the Operator, the Large Language Model provider, the Autonomous agent, and the broader AI ecosystem to adhere to. This intelligence

contract serves as both a literal agreement semantically defined in markdown, and a mutual understanding amongst the AI ecosystem that while this newfound technology provides incredible opportunities for growth in industry, science, and business, it also poses an unprecedented risk to privacy and property. We must recognize that the individuals who are most familiar with the boundaries should be in charge of the intention, understanding, and access of these autonomous agents.

The Dual-Faceted Contract

It's come to my attention that there is still more to explain. I struggle to emphasize the importance of this contract structure in the broader understanding of AI technology. The contract represents an agreement of safety and control between normal individuals and large firms full of smart people that build these technologies. Through collaboration between the community and companies, it is my strong belief that we can safely and quickly integrate artificial inference into the broader global economy. The markdown contract and its inherent agreements represent the diffusion of AI knowledge throughout the tech stack, all the way to the consumer. Markdown is a language that is incredibly flexible and portable, capable of supporting operational semantics, safety restrictions, data processing agreements, and most importantly, universally understood by the common man. The discovery of the Intelligence Contract acts as an agreement between consumer and provider, that this is how these systems ought to be controlled, with the consumer behind the wheel. However, this contract requires near universal adoption by the broader ecosystem, encompassing open source tools, industry leading firms, and the common man in order to control autonomous agents in this manner. The flexibility of markdown contracts offers opportunities for those privacy-focused to self-host tooling that adheres to standards, while also providing industry opportunity for firms to build adhering systems.

On the flip side, this contract serves as another evolution in programming. A mentor of mine once said that "programming is just translation of intention into machine code"—or something along those lines. The new contract structure follows a paradigm we at work have been calling "semantic programming" for years. It's a description that is intended to be understood and constantly adhered to by autonomous agents. These agents are capable of more than we give them the capability to do. With semantic encoding of knowledge and understanding in the hands of the average man—the non-technical—workflow automation will reach bounds never considered. The reflection process of this contract is paramount. That's where the trust is verified. By offering a reflection line that presents the agent's thinking patterns, the common man is able to understand and validate the thinking process of the agent and its understood boundaries and restrictions. With this auditability standardized throughout the toolset, an operator is able to debug their own agent. For example, if an operator is attempting to build a workflow and recognizes that an agent isn't following data privacy restrictions, the operator ought to cease use of that agent and validate their semantic understanding of the task via agent interrogations. If the agent appears to understand the task, but its generated code doesn't reflect the correct understanding, then the agent or backing model ought to be considered "broken". If it can't understand human logic, it shouldn't be doing human tasks.

Democratization and Consumer Control

The auditability of this paradigm presents a novel opportunity to shift the responsibility of data privacy and personal data safety onto the consumer. By setting up a system in which the model and agent providers build technology that adheres to contracts, the Operator is put in direct control of what data is processed and more importantly how it is processed. Referencing my example, the Operator might define in the contract that "no personal data should ever be used for the training of models" and "you have access to a codebase but you should never share your understanding of our code with anyone else". In this way the Operator is able to define restrictions and controls for privacy. This presents a novel opportunity for the common man, those unfamiliar with the absurd complexity of modern computing and programming. The contract is a democratic paradigm for governing data access and understanding in the modern digital age.

A Note on Theory and Implementation

I'd like to emphasize that this is all theory. I halted implementation of the Eugene debugger so that I could record my discovery. Without adherence to this contract, I'm not currently comfortable turning Eugene on and letting it access customer data. I need reassurance from someone of relevance that this data will be respected and handled as defined by me, the Operator.

Conclusion: A Call for Universal Adoption

The Intelligence Contract represents a fundamental shift in how we think about AI governance, privacy, and the relationship between operators and autonomous agents. It provides a framework that is both technically sound and accessible to non-technical users, enabling a new level of control and transparency in AI interactions.

The implications are profound. If we can establish a standard where contracts serve as binding agreements between operators, agents, and providers, we create a foundation for safe, scalable AI adoption across industries. The contract becomes both a safety mechanism and a programming paradigm—a way to translate human intention into machine behavior through semantic understanding rather than explicit code.

This essay does not provide definitive answers, but rather proposes a framework that demands serious contemplation from every individual and organization working with AI systems. As autonomous agents become increasingly capable and widespread, the question becomes: who controls the boundaries? I propose that the individuals who are most familiar with those boundaries—the operators, the domain experts, the privacy-conscious users—should be in direct control. The Intelligence Contract provides the mechanism for that control, but it requires near-universal adoption to be effective.

The tools are changing, and we must consider whether our thinking about AI governance should change to match. The Intelligence Contract offers one path forward—a path that puts control in the hands of those who understand the context, the boundaries, and the consequences of autonomous agent behavior.

Original essay at robertjw.dev