Atlas Protocol Whitepaper v1.0

Atlas Protocol: Foundation Documents

Glossary (v1.0)

Atlas Protocol – A modular, intent-driven digital design system where parts, processes, and systems are generated, versioned, and orchestrated using logic, not geometry. Atlas is built with API modularity in mind, allowing it to scale across industries, systems, and collaborative environments with seamless integration potential. It is CAD program agnostic, enabling complete freedom of tool choice while ensuring consistency and interoperability across the design process.

Intent Design / Logic – The process of defining the purpose, constraints, and behavioral rules of a part or system, enabling Atlas to generate the actual design from those inputs. This is the core of Atlas' generative capability and enables modular adaptation and system-wide coherence.

Ripple Cognition – The core principle that any change to a part or constraint propagates through the system, alerting or modifying downstream components accordingly.

Enigma – The version control engine of Atlas tracks every change, commit, and branch in a Git-style system for physical and digital designs.

BOMBE – The logic engine that analyzes changes and resolves ripple effects, handles conflict resolution and auto-adapts where allowed.

Titan – The master orchestrator layer; sets global constraints and relationships between subsystems, ensuring that modular parts cohere in massive systems (e.g., aircraft, smart cities). Especially useful for larger, more complex projects or where separate teams work on distinct components in parallel.

ACE (Atlas Community Edition) – The open-source implementation of Atlas, licensed for public use, development, and community innovation.

Atlas Liberation License (ALL) – A dual-license designed to allow open innovation, prevent misuse by oppressive entities, and ensure commercial users contribute back or purchase a license.

Intent Constructor – A new design role focused on programming intent logic, rule-based generation, and constraint behavior, rather than drawing geometry.

Intent Metadata – Structured data embedded within each design element that describes its origin, purpose, constraints, material properties (e.g., steel grade, durability, weight), and allowable ripple behavior. This metadata allows Atlas to trace causal relationships and predict downstream impacts across parts and systems. It also enables Gauntlet testing,

where multiple twin configurations can be compared based on performance, efficiency, material stress, and environmental conditions.

Digital Twin (Intent-first) – A digital representation generated from logic and constraints, not reverse-engineered from reality. It is the source of truth, not a reflection.

Fork/Merge Design – The ability to create variations of a part or system, test them independently, then merge changes back into the master twin (Titan-regulated). This mechanism also operates at the individual part level within a single Atlas instance, allowing for localized iterations, modular experimentation, and controlled ripple resolution without needing full system-wide orchestration.

Atlas Protocol Interface – An agnostic viewport into the twin model, supporting exploded views, transparency, and selectable geometry. Users can launch the selected drawing directly into their preferred editor. (Future expansions may include showing the 'shadow' of affected drawings to visualize ripple changes in real time — enabling predictive understanding of design impact.)A UI/UX approach that lets users hide or explode parts, observe constraints, and interactively select nodes to edit or regenerate parts.

Ripple Bi-Directionality – When a downstream part's limitation can block or reverse an upstream design change, enforcing co-dependency rules. When coordinated through Titan, the ripple can, if allowed, propagate through multiple Atlases based on hard constraint boundaries, ensuring coherence across distributed systems.

Gauntlet – A comparative testing framework within Atlas that uses intent metadata and material properties to evaluate multiple digital twin variants. Gauntlet enables performance benchmarking across criteria such as durability, weight, structural stress, cost efficiency, and environmental resilience. This allows users to simulate and select optimal designs in a ripple-aware, constraint-bound environment.

Caladan – A global material availability intelligence layer that integrates with Atlas to track and forecast disruptions across the material chain. Caladan analyzes EPDs (Environmental Product Declarations), regional raw material output, shipping manifests, production data, and export trends to identify constraints and ripple threats. It attacks the material chain from both ends — tracing resource origin and projecting availability into design decision-making. By piercing the fog of logistics, Caladan enables proactive adaptation of designs, material substitutions, and strategic stock recommendations.

Relay – The ERP integration and real-world inventory interface within Atlas. Relay connects with existing ERP systems (e.g., Monitor G5) via APIs — starting with read-only access for real-time BOM verification and material matching. It enables part autocompletion by scanning available inventory and proposing compatible parts, or generating new parts if none are suitable. Relay ensures BOM accuracy, shortens lead times, and turns the digital twin into an ERP-informed, stock-aware entity. Additionally, Relay supports exporting BOMs as CSV files for manual import into ERP systems, making it compatible even with limited or offline environments.

Whitepaper (v1.0)

Title: *Atlas Protocol: A Ripple-Aware, Intent-Driven Design Framework for the Modular Age*

Author:

Tom Erik Harnes

Founder & System Architect, Atlas Protocol

Norway, 1. May, 2025

email: teharnes@gmail.com

Linkedin: https://www.linkedin.com/in/tom-erik-harnes/

Executive Summary

Atlas is a modular, intent-driven digital design framework that allows physical and digital systems to be constructed from purpose, not just geometry. Using a ripple-aware logic engine, Atlas ensures that changes propagate intelligently throughout connected systems. It redefines how we manage CAD, ERP, versioning, and real-world manufacturing — with minimal waste, full traceability, and a future proof protocol.

1. Introduction

- Have you ever stood there with a part in your hand that didn't fit and wondered why this wasn't caught at an earlier stage in production, hopefully during final design?
- Have you ever thought about why the part couldn't just change to reflect the changes of other parts when either they or something else changed, maybe the space around it also changed?
- How about BOMs (Bill Of Materials), how often has part either been missing, the wrong number of parts or an entire wrong part?
- How much time do you think has been lost, how many deadlines pushed, overtime worked because of mismatched drawings, outdated assemblies?
- Digital Twins today are a product of the parts available, most of the time there is no twin at all. It's labor intensive to add all parts, bolts and nuts and keep it updated.

2. Atlas Protocol Overview

- Atlas works on a completely new design philosophy, instead it flips the process, twin first, draws from intent and generates parts as needed based upon intent logic, it doesn't design the part but understands the reason it needs to exist, and it's up to you to define that intent and at what level.

  My journey started from a single piece, could I make Python to automate the STEP file of the part? Yes I could, all I needed was the measurements and geometry rules and limits to how it should look.

  But why should "I" put in the measurements when they all had to "fit" around an object and inside a different object, so my idea expanded to generate the parts that set the geometry for that part.

  But what decides those sizes? In my case it was a technical design report stating the measurements. This entire report was the baseline for the entire item, in my situation it was a transformer. The case had to be this big on the inside, this amount of cooling fins, their size. Then the core has to be this big, windings like this. And from all of this you extrapolate the parts needed to secure the other parts, those have to be around the core, fit the case and must have bolts going through here. Simple logic and a finished Twin can be generated, all drawings generated, all parts like bolts and washers could be added based upon logic in that the holes in the lid should include bolts, washers and nuts, and we just search them up in the ERP database and add them. Now we can generate the BOM automatically based upon the complete model.

- Now we take it one step further, the engineer who designed the transformer, he has guidelines that he has to follow. Transformer must be no bigger than this to fit the room, it must have less than 1000 liter oil, the power loss must be less than this. Then he uses a design program to run different calculations until hopefully all specifications are met. But how about automating this too? And not only basing it upon those specs, how about we also include material price, availability, emissions, etc. Since Atlas generates a complete twin all the time, it always knows how much material it needs, the weight, EPD data. You could run multiple tests against several scenarios and select the one that suited the current situation.

- Atlas as a system is made to be a CAD program agnostic, it removes itself from the old PDM style version control and embraces Git with intent metadata. You not only specify the material type, but you could add stress limits or durability. With Git-like version control you could branch or fork your own version, allowing multiple users editing the same drawing at the same time with a conflict resolver.

- All changes done to one piece would generate an Enigma, the diff value from the commit that "Bombe" would try to resolve. It will check face to face alignment, see if a bolt hole moved and if there are collisions. If collisions or mismatched holes are detected enter the affected part and modify it. Then check if that design change made another conflict. This propregate all the way down until there are no more conflicts. If there by any chance was an unresolvable conflict based upon intent logic and constraints, it could be bi-directional, reporting back that the intent has reached a constraint limit and negotiates pieces to adjust to comply. If it's still not resolvable, the file is flagged for human modification.

  This is what we call ripple logic, it not only catches change - but understands consequence.

- Atlas Protocols basic idea is to piece the modules out, to make it into an API-driven architecture, scalable and able to integrate into any future modules. Where today's Digital Twins are copies, Atlas is the origin.

- Atlas Protocol as a whole, is more than the sum of its parts, it's a Gestalt. And if you can see the potential vision of it as I do, you too will understand that this is not something that should be locked away under closed code, the ecosystem this can be is beyond my current understanding.

  And it's the reason why I as the founder and system architect of Atlas Protocol want to release it to the world as open source encouraging open development, but with the ability for those who want to make closed modules able under licensing fees.

  This will ensure that everyone has the same tool available to explore the ideas and vision about what true scale is, it's so enormous it is hard to grasp. From generating a piece to global material flow.

  As an example, not enough power in the grids. First-order thought is to make more power, better grids. But what would happen if we instead stopped wasting power on parts, forged or produced in China, shipped half around the world to only be recycled or scrapped because an outdated design drawing was used.

  How much time, money and resources has been wasted only because of this? This is all resources we have available, but wasted. And if Atlas even at its early start, generating a complete digital twin model from input data could eliminate the headaches of incompatible parts that are worth something, and it could be worth so much more. The second and third-order effects could be immense.

  We don't fix the problem. We fix the reason the problem exists in the first place.

  And here's one of the most basic and fundamental truths about the Atlas Protocol. This technology is possible today, even without AI, and has for a long time. All you have to do is explain your intent.

3. Core Components

- **Enigma**: Git-style version control of parts, assemblies, and logic
- **BOMBE**: Ripple analyzer and logic resolver for local/global change
- **Titan**: Constraint-based global orchestrator (macro system design)
- **Relay**: ERP integration layer; autocompletes BOMs, exports CSVs
- **Atlas Protocol Interface**: Viewport for twin model interaction (explode, transparent, ripple-shadow preview)

4. Strategic Impact

Domain	Current Standard	Atlas Advantage
CAD	Geometry-defined	Intent-defined
PDM	Static/versioned	Branchable + reactive
ERP	Manual BOM sync	Live ripple-aware autocompletion
Energy	Wasted power from inefficiency	Optimized workflows via intelligent intent propagation

- Enhances ERP without replacing it
- Eliminates need for traditional PDM
- Introduces ripple-aware scheduling and lead-time prediction

5. Licensing & Ethics

- Atlas Liberation License (ALL): dual-license model
- Free for community, commercial requires contribution/license
- Ethically guarded: bans oppressive, surveillance, and weaponized use
- Open innovation with defense against misuse
- Encouraging community-led modules and logic libraries

6. Future Ecosystem (Under Development)

- **Caladan**: Global material and disruption monitoring via EPDs, shipping, sourcing
- **Gauntlet**: Twin model testing across materials, cost, durability, and performance
- AI-enhanced co-creation based on structured intent logic
- UI frameworks for visual design manipulation and ripple tracking
- Modular, drag-and-drop intent logic editor for non-coders
- Community editions, commercial extensions

7. Call to Action

- Join the movement
- Fork the repo, contribute logic modules, or connect ERP systems
- Rethink how we design, build, and adapt
- Help bring humanity into the post-PDM design paradigm

Atlas Liberation License (Draft v0.1)

License Summary:

- Open for non-commercial use, education, research, and prototyping
- Commercial use requires registration or licensing
- Contributions must stay open source
- Prohibited for use in systems of oppression, mass surveillance, or intentional weaponization targeting civilian populations. Ordinary military R&D or dual-use applications are not automatically excluded but must comply with ethical transparency and non-oppressive intent.
- Redistribution must preserve license, authorship, and core principles

Full license text: Coming soon in LICENSE.txt