Designing the Rule Layer: A File Type for Interoperable & Productized Construction

How standardizing product constraints will unlock configurators, automation, and cross-vendor building assembly.

Revised December 11, 2025

NEW YORK TECH

Jason Van Nest
Executive Director
Center for Offsite Construction

School of Architecture & Design
1855 Broadway
New York, NY 10023

Steve DeWitt
CfOC Senior Research Fellow

Custom Building Configurators

52 Cordone Drive
San Anselmo, CA 94960

Samuel Williams
CfOC Senior Research Fellow

Matechi

6221 Shattuck Avenue
Berkeley, CA 94609

AI Usage Disclosure: Earlier versions of this document were drafted with assistance from AI tools – especially meeting transcription, outlines, spelling and grammar. The content has been reviewed and heavily edited by several humans.

Acknowledgements

The authors gratefully acknowledge a huge community that has made this work possible.

As we complete this draft, we'll make sure to complete this section.

Executive Summary

The U.S. construction industry is undergoing a structural shift from bespoke, Engineer-to-Order (ETO) delivery toward Configure-to-Order (CTO) building systems shaped by industrialized offsite manufacturing. Pods, panels, utility walls, and modular assemblies are increasingly designed, certified, and produced as products—not drawings. Yet the digital infrastructure that governs these products remains fragmented across incompatible BIM tools, manufacturing CAD platforms, proprietary configurators, and ad-hoc metadata. Each vendor encodes its rules differently; each configurator rebuilds its logic from scratch; and no shared digital substrate exists to describe how offsite products may be configured, connected, or composed. This fragmentation prevents automated configuration, undermines interoperability, and stalls the emergence of a national CTO marketplace.

This whitepaper proposes a solution: the development of a Configurator File Type.[1] The file type is an opt-in, rule-native, vendor-neutral standard that captures the geometry, interfaces, constraints, allowable variations, parametric logic, and version history of offsite products in a machine-readable format. The file type does not replace BIM or MCAD. It sits above them, providing the missing semantics those tools cannot express. It enables configurators (either human-driven or AI-driven) to assemble products safely, predictably, and in compliance with both manufacturer specifications and code requirements. Once standardized, the file type allows products from different manufacturers to interoperate for the first time, forming the substrate of a decentralized, multi-vendor CTO ecosystem.

It is critical to distinguish these two activities: offsite products undergo a dedicated *product design stage*, in which engineers define the product platform, interface geometry, allowable variations, and certifications; only after these definitions are complete can a *building configuration stage* occur, in which designers and configurators assemble buildings from pre-engineered products.

The benefits of such a standard are profound. Manufacturers can publish authoritative, versioned catalogs hosted on their own servers. Configurators can fetch those definitions in real time and compose valid building systems without relying on proprietary rule engines. Developers and general contractors can explore thousands of feasible configurations in hours rather than weeks. Code officials can review configurations through embedded metadata rather than through ambiguous drawings. And AI agents—operating inside constraint-bounded configuration spaces—can produce building layouts, unit plans, and assemblies that are valid by construction, not heuristically plausible. The result is a building delivery system that is more scalable, more transparent, and dramatically faster.

To realize this future, the standard must be developed through ANSI-compliant governance, ensuring balanced stakeholder representation, transparent drafting, structured comment resolution, and rigorous public review. As the first ANSI-accredited standards developer dedicated to offsite construction, the Center for Offsite Construction (CfOC) is uniquely positioned to steward this work. The implementation roadmap includes domain modeling, technical specification, integration with existing toolchains, pilot deployments, and continuous maintenance through recurring consensus cycles.

Building such a standard requires a coalition. We call on technical experts—product engineers, parametric modelers, computational designers, and software architects—to help define the rule structures that will

[1] This file type builds upon prior CfOC work on the Configure-to-Order methods needed to scale offsite methods in North America. This work, cataloged at the *Future of Design & Delivery* "Research Roadmap," offers a continuity of efforts to bring open-source, opt-in IP to the AEC industry.

govern next-generation building systems. We invite funding partners—federal agencies, philanthropic foundations, housing accelerators, and impact investors—to support convenings, prototypes, and pilot implementations. We welcome software liaisons—from configurator firms, CAD/BIM platforms, and open-source communities—to shape integrations across AEC toolchains. And we encourage policy leaders, code officials, and housing authorities to align regulatory frameworks with the capabilities this standard unlocks.

A rule-native Configurator File Type is not just a technical artifact—it is the foundation of a modern building economy. By standardizing how products describe themselves, the industry can finally transition from interpretation to configuration, from drawings to products, and from isolated solutions to a vibrant, interoperable CTO marketplace. The CfOC is prepared to lead this effort. We invite stakeholders across the building ecosystem to join us in constructing the digital infrastructure that will shape the next century of construction.

1. The Problem: Fragmented Tools, Siloed Data, and the Limits of ETO

For more than a century, the AEC industry has operated inside an **Engineer-to-Order** (ETO) paradigm. Every project is bespoke; every building is a one-off. Contemporary tools created to support this paradigm (BIM for architects, MCAD for manufacturers, and a patchwork of proprietary fabrication software) assume that a building will be designed once, constructed once, and not repeated in precisely the same engineered form again. As a result, the digital infrastructure is fragmented by design. Each domain maintains its own conventions, file formats, metadata structures, and workflows.

Today's offsite product manufacturers are forced to operate inside this ETO logic even when their business models depend on repetition, configurability, and standardization – which means they *depend* on their collaborators' focus on repetition, configurability, and standardization.

Offsite manufacturers' product definitions live inside environments never intended to fully support their use. An exterior wall panel exists as a Revit family on one workstation, a SolidWorks assembly on another, a PDF cut sheet on a server, and a spreadsheet of production tolerances on a local drive. Each fragment carries part of the truth (its ingredients, context, and performance) but none carry the whole. When that panel appears in a BIM model, it loses critical manufacturing constraints; when it appears in a fabrication tool, it loses the architectural context that determined its role within the building. At no point is the building element (and its application) represented by a single, authoritative, machine-interpretable definition.

In an ETO world, the *product design stage* is invisible because products are rarely treated as products; in a CTO world, this upstream stage becomes decisive. Engineers author the product's geometry, interfaces, constraints, and certification conditions once, and those definitions anchor every downstream building configuration.

This fractured data environment is a root condition stalling the transition to scalable offsite construction. It prevents manufacturers from publishing their products in a consistent way. It prevents building designers from rapidly applying the products as intended. It prevents contractors and developers from comparing equivalent products across vendors. It prevents code officials from validating a product's use without requiring bespoke documentation. And it prevents automated tools (configurators, generative engines, and AI agents) from reliably assembling products into coherent buildings.

The result: every existing offsite configurator (no matter how capable) is a *silo*. Each firm builds its own proprietary ruleset, its own modeling conventions, its own interpretation of interfaces, and its own logic for allowable or disallowed configurations. None of these systems speak to one another. None expose their constraints in a standardized format. None share their intended dependencies in a standardized format. Each must be rebuilt for each firm; perpetuating an inherently unscalable structure.

In this environment, even manufacturers with excellent product platforms are trapped inside closed ecosystems. A vendor's catalog cannot be easily imported into another firm's configurator; a designer cannot move from one vendor to another without re-authoring the logic; and multi-vendor solutions (essential for multifamily housing) are nearly impossible to automate. Because the underlying file types cannot store rules, decision-spaces, or product interfaces, every workflow must manually rebuild that logic from scratch.

And critically, emerging AI tools cannot operate safely in this environment. They lack the constraint-aware substrate needed to generate valid building solutions. Without a standard describing what a product is, how it connects, how it varies, and what configurations are permissible, agentic design tools can only hallucinate plausible geometry, and never guaranteed, interoperable building systems.

In sum: ETO-era file types cannot support **Configure-to-Order** (CTO)-era workflows. The industry's digital foundations have reached their limit. A new substrate is required.

2. The Administrative Stack Behind ETO Construction

The **Engineer-to-Order** (ETO) paradigm is not only an economic or legal system. It is also a digital system. The contracts, scopes, inspection practices, and labor jurisdictions that define contemporary construction created (over decades) the BIM, MCAD, and exchange formats that now structure the industry's entire digital landscape. These file types inherited the assumptions of their administrative environment: that every project would be bespoke; that rules would be interpreted by licensed professionals; and that building systems would be engineered once, documented once, and inspected once.

Because this administrative worldview assumed *bespoke building design and engineering*, today's dominant file types never evolved to carry the rule structures required for productized, interoperable offsite construction. They document what was drawn, not what is allowed. They express geometry, not constraints. They record intent, not composability.

This chapter outlines the ETO legal-administrative stack, and shows how its logic shaped the digital tools now limiting the transition to **Configure-to-Order** (CTO) workflows.

2.1 The ETO Legal Worldview: Services, Not Products

ETO construction is rooted in a service-based legal model.

At its center sit the Standard Forms of Agreement (AIA Contract Docs, ConsensusDocs, and their peers). Documents such as the AIA's A101 (Owner–Contractor), A201 (General Conditions), and B101 (Owner–Architect) define how parties relate, who designs, who builds, how changes are priced, how disputes are resolved, and how payment flows.[2]

These standard forms of agreement carry a constellation of fundamental assumptions. They are based in a Common Law, service-based model of construction. They are the foundation of a mature design-based administrative stack, requiring contingent specification systems, rigid drawing conventions. They also are the foundation of an established construction-based logic: service-based payment mechanisms, diverse inspection domains, and a constellation of labor jurisdictions all propping up to an ETO world. Long before a specific project exists, this infrastructure runs ahead of the project, shapes how scope is divided, defines how risk is allocated, and structures how (on-site) work is performed.

All these documents forge relationships built on the presumption that value is created as professional services and labor are performed over time, in the future. The AIA Schedule of Values (AIA G703) operationalizes this logic: it allocates the contract sum across site-based activities, and progress payments are certified against "work in place," not goods in inventory. Risk, cash flow, and performance are all indexed to *service categories*.

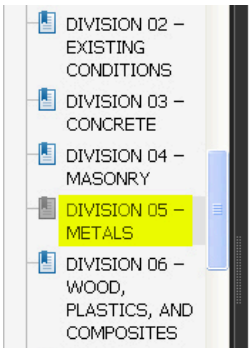
ETO contracts are anchored in an architect's project manual: the construction drawings, specifications, bidding instructions, etc.[3] that the AIA framework treats as the architect's "Instruments of Service." Geometry, details, and specifications become the legally enforceable description of services, called "the

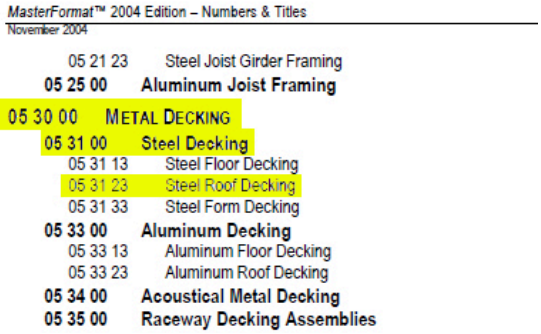
[2] Of interesting note, IPD generally, and AIA G202 introduce non-design-bid-build legal arrangements. Such IPD workflows, might be restudied here.

[3] For more on the bespoke nature of the architect's Project Manual, see the AIA's *The Architecture Student's Handbook of Professional Practice*, 15th Edition (2017) page 390.

Work."[4] When a contractor applies for payment, or an owner asserts a defect, the question is whether the Work conform to these documents and their categories. The system assumes that what matters legally is conformance to bespoke design intent, not conformance to a catalogued product.

The CSI MasterFormat ties this legal and financial structure into shared technical categories.[5] Specifications are organized into divisions and sections;[6] the AIA Schedule of Values[7] commonly mirrors those divisions; bids and subcontracts are sliced along the same lines.[8] This alignment allows every dollar, drawing, and dispute to be traced back to a numbered category of "the Work." MasterFormat is not just a filing system — it is the categorical spine along which ETO risk, scope, and responsibility are pre-sorted before construction begins.





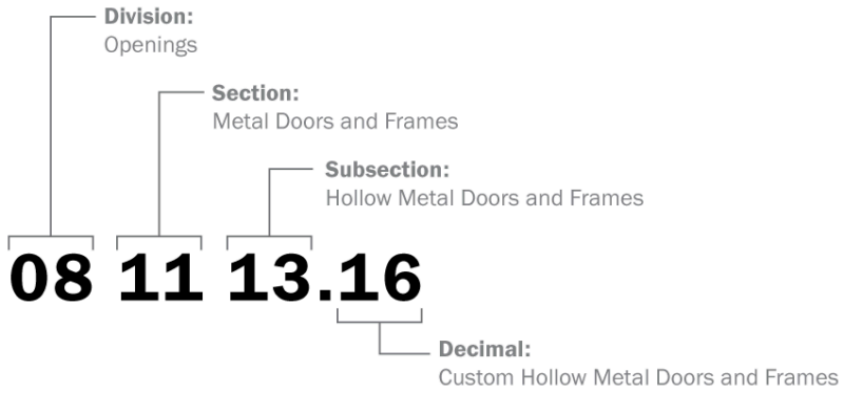


Figure 2a: *A guide to understanding MasterFormat's three-part number (sometimes four-parts) for classifying construction products into categories, composed by ArchToolbox.*

Organized in parallel, is the layer of union and trade jurisdiction, which organizes who may perform which scopes of work. While MasterFormat classifies by *system*, unions classify by corresponding *trade*. General

---

[4] AIA A201-2017 § 1.1.3 The Work : The term "Work" means the construction and services required by the Contract Documents, whether completed or partially completed, and includes all other labor, materials, equipment, and services provided or to be provided by the Contractor to fulfill the Contractor's obligations. The Work may constitute the whole or a part of the Project.

[5] From CSI's CEO: "The textbook definition of specifications would say it's a document that outlines the materials, methods, and practices that are used in a construction project, but that leaves out a lot of critical nuance provided by the specifiers themselves." from CSI's blog article *CSI CEO Mark Dorsey … Shares How CSI's Rich Past Will Lead to Future Opportunities* (April 26, 2023)

[6] See CRM Service's guide to divisions.

[7] See more at *What is a Schedule of Values and Why is it Required on Construction Projects?* by Sara M. (Bour) Betancourth, March 2, 2018.

[8] See more at *Understanding the Schedule Of Values (SOV) and Payment Applications: An entry-level perspective.* by Gisel Marisol Hernandez, March 22, 2022.

contractors and construction managers reconcile these two maps using the AIA-CSI contract hierarchy and dispute mechanisms. In practice, this combination of contracts, specifications, and labor rules functions as a *self-reinforcing operating system*: it pre-distributes work, pre-defines boundaries, and pre-negotiates many of the conflicts that might otherwise arise on every job.

Within this ETO structure, the boundaries of "the Work" are not neutral; they are contested terrain. Each CSI division, specification section, and pay-application line item represents both economic opportunity and jurisdictional claim. When new technologies or assemblies appear (such as prefabricated bathroom pods) they threaten to redraw those lines. *Should the pod be classified under Division 22 (Plumbing), Division 09 (Finishes), or treated as a specialty under Division 13 (Special Construction) or Division 11 (Equipment)?* Each choice determines which trade or union gains authority, man-hours, and dues. Competing locals may argue that their members should install the pod because it contains their traditional work — carpenters for framing, plumbers for piping, electricians for wiring, tile setters for finishes. In an ETO system, where labor jurisdiction and payment are linked to divisions of "the Work," expanding one's scope to capture these new assemblies becomes a defensive act of economic survival. The system rewards territorial control, not integration, making innovation appear as encroachment rather than progress.

In the same way, offsite products are inspected not as standardized goods but as bespoke assemblies; they are evaluated case-by-case by each jurisdiction having authority (JHA). Identical bathroom pods can therefore face entirely different inspection criteria, documentation requirements, and approval pathways depending on where they land, reinforcing the same territorial fragmentation that slows adoption and prevents true productization.

These recurring categorical and jurisdictional contests are evidence of a highly tuned, self-reinforcing system that keeps the ETO ecosystem coherent, even as it resists change.

Taken together, these elements form a coherent ETO administrative ecosystem. The ecosystem *runs ahead* of individual projects, standardizing expectations, workflows, and relationships, so that bespoke buildings can still be delivered through familiar patterns of documents and roles. Crucially, it is all quasi-optimized for *a world in which scope is defined as services performed on site*, not as products manufactured to fit together (in advance).

In summary, within the ETO system:

- **The Architect's Instruments of Service** (drawings, specifications, addenda) describe a unique building, authored for a specific client, on a specific site.
- **General Contractors and Subcontractors** relate to each other through the Architect's instruments (construction documents and specifications), overlaid with a hierarchy defined by trades, not by product responsibilities.
- **CSI MasterFormat** serves as the categorical backbone for this hierarchy, mapping scopes of work to divisions and sections that align with labor jurisdictions.
- **The Schedule of Values (AIA G703)** ties payment to "percent complete" work executed on-site—not to the procurement of pre-engineered components.
- **Jurisdictional inspectors** evaluate assemblies case-by-case, reinforcing the idea that each installation is uniquely produced and uniquely judged.

The entire ETO legal apparatus presumes that the industry's fundamental unit is *work performed by professionals*, not *products manufactured in advance*. Rules governing assemblies (i.e. their required relationships, allowable contexts, and performance conditions) are distributed across contracts, specifications, addenda, submittals, RFIs, shop drawings, field directives, commissioning reports, and the tacit judgment of licensed practitioners and civil servants.

In a world where rules live in documents and people, not in components, no digital file type is asked to carry them.

2.2 How ETO Legal Logic Became the Blueprint for Digital Tools

Because the legal system treated buildings as bespoke artifacts, digital tools evolved to support *interpretation*, not *pre-certified configurations*. BIM and MCAD environments were designed for authorship, annotation, and documentation, and not for expressing interoperable product logic.

Digital tools inherited ETO legal assumptions in several ways:

- **BIM platforms** (Revit, ArchiCad, etc.) mirror the architect's legal role: they may start with a building model, but that is not the legal deliverable. Instead, the underlying model is background art to produce drawings whose meaning is interpreted through specifications and professional judgment.
    - The core organizing element in BIM is levels (or reference planes) that simply describe planes that host (extruded) system families.
    - Then, system families are hosted to these levels. System families are defined on a project-by-project basis. They are site-fabricated assemblies: the walls, floors, ceiling, and roofs of a building. Most of these site-build assemblies are required to host loadable families.
    - Only then, can loadable families be deployed in a project. Loadable families are the products of manufacturing: windows, sconces, toilets, outlets, and kitchen cabinets.
    - It's almost impossible to place a product of manufacturing into a BIM model without defining site-built assemblies. Windows and doors cannot be deployed without a site-defined wall. Hatches cannot be deployed without a roof. Cabinets cannot be defined without a partition.
    - Even if one "tricks" BIM software to do this, the product's constraints and dependencies are unresolved. For example if one placed a 3'x5' window on a 3'x5' wall, the window can appear to float in space. Still, the building designer would have no computational access the the window's required rough opening (which is dependant on wall composition), minimum sill height (which is dependent on level and use case), or size requirements (dependent on manufacturer's catalog).
    - System families (not loadable families) carry the functions of the architecture discipline. To preserve "enclosure" site-built walls are "attached" to roofs, and auto-extend to meet them (creating unstudied assembly conditions). To preserve "integration," the architect's wall finish is "joined" to structural cavity walls, allowing hosted windows to punch through the total wall assembly, but creating unstudied jamb and sill conditions.

  Contemporary BIM's hierarchy reflects the ETO legal landscape: the architect is contractually responsible for defining the *spatial datums*, *primary assemblies*, and *coordination boundaries* that all other parties rely upon. Levels, grids, and system families exist because the architect must legally

describe the host conditions (i.e. enclosure, structure, fire separations, accessibility clearances) into which manufactured products will eventually be inserted. These site-built assemblies become the juridical "canvas" against which compliance is measured: inspectors, contractors, and subcontractors all evaluate products relative to the architect's documented assemblies, not relative to the manufacturer's intended usage context. In effect, BIM embeds the architect's service obligations directly into the model's organizational spine, ensuring that industrialized components can only be deployed *after* the architect defines the legally authoritative surfaces, cavities, and interfaces they must occupy. This structure makes contemporary BIM software an instrument of the ETO legal regime… an environment optimized to express architectural responsibility, not manufacturable product rules.

- **Project Documentation Platforms** (Procore, Newforma, etc.) create a parallel layer of the ETO digital environment. These platforms do not author geometry, but they operationalize the legal architecture of the AIA contract suite. Their entire structure—roles, workflows, timestamps, approvals, and document hierarchies—is built on the assumption that buildings are bespoke, assemblies are site-built, and compliance must be demonstrated through a traceable narrative of review and interpretation. They are, in a very deep sense, *the digital manifestation of the AIA contract system.*

  Every workflow in these environments mirrors a corresponding legal instrument.

  - **RFIs** replicate the contractual requirement that ambiguities in the Instruments of Service be resolved through written clarification.
  - **Submittals** recreate the gatekeeping role of the architect and consultants to confirm conformance—not to a product spec, but to a project-specific interpretation of design intent.
  - **Shop drawings** represent the contractor's documented transformation of design intent into constructible means and methods, requiring approval on a project-by-project basis because no pre-certified product logic exists.
  - **Punch lists and daily reports** record the continuous verification that site-executed work aligns with the contract documents, not with a product rule set.
  - **Change orders** formalize the economic implications of deviations from bespoke intent, reinforcing the idea that every adjustment implicates services, not interchangeable products.

  This administrative layer exists because the ETO system assumes that rules are *external* to building components and must therefore be rediscovered, debated, approved, and archived for each project. These platforms capture professional judgment as data. The digital tools do not require it, but the legal model demands it. For manufactured elements, this means that every window, panel, fixture, or pod is reinterpreted within the project's documentation ecosystem, rather than understood through authoritative, pre-certified, machine-readable logic authored by the manufacturer.

  The outcome is a system where digital platforms excel at recording how the industry resolves ambiguity, but have *no mechanism for eliminating ambiguity in the first place*. They create enormous administrative structures, because they inherit a world where products have no inherent, pre-certified rule sets… and where configuration cannot occur without an accompanying narrative of professional interpretation.

- **(optional) MCAD tools** (SolidWorks, Inventor, CATIA, etc.) are digital environments used to design, engineer, analyze, and document manufactured physical components, from machine parts to assemblies to full mechanical systems. They exist outside of building ETO, and mirror the manufacturing engineer's role: they define precise geometry and assemblies, but their proprietary formats assume variation will be manually controlled by trained engineers.

- **(optional) Inspection workflows** shaped the way compliance is represented: BIM expresses the *result* of judgment, not the *rules* that would allow automated validation. Most of these rule (outputs) aren't exposed by default, many go unused, and nearly all must be manually searched for..

- **(optional) Progress-based payment systems** shaped the granularity and content of digital documentation: models and drawings describe what will be built, not the rule-space of what *may* be built.

These tools were created to satisfy the administrative needs of a Common-Law, service-centered industry. They do not—and cannot—encode constraints, interfaces, allowable compositions, or certified limits.

They are the digital expression of a legal regime that expects *professionals*, not *products*, to carry the rule layer.

We will see how, under CTO, the majority of 'design' migrates upstream into the product design stage, leaving downstream teams to perform building configuration, not authorship.

2.3 File Types as Instruments of Professional Interpretation, Not Configuration

Each major file family—BIM, MCAD, IFC, PDFs, spreadsheets—stores geometry, parameters, and annotations about the product (or product platform), **but not usage rules or acceptable product contexts**, because rules lived in contracts, specifications, and professional judgment.

- BIM files store shapes, not context-based constraints. BIM and IFC files describe building components only as they appear in a specific project context, not the full range of contexts in which a product may legally or functionally operate.

- MCAD files store part relationships, material definitions, EBOM information, but not allowable deployment configurations.

- PDFs (construction drawings, shop drawings, and other vector-based 2D representations) freeze the architect's or fabricator's interpretation of intent, and it's editorial frame, but not the manufacturer's logic for a product, or the variations of its use.

These file types succeed at retroactively documenting what was authored. They cannot proactively express what is permissible.

2.4 Why Today's Tools Cannot Host Product Logic

When offsite manufacturers attempt to publish their products using existing formats, they discover that the underlying data structures simply have no place to store the essential rule layer of a productized system.

ETO file types cannot express:[9]

- Required or prohibited adjacencies
- Interface definitions that must align with other products
- Parametric ranges (min/max lengths, widths, offsets)
- Conditional configuration *constraint* logic ("instance X allowed only if geometry is…")
- Conditional configuration *dependency* logic ("variant Y allowed only when host level is…")
- Performance constraints (fire, acoustic, structural) tied to usage contexts
- Inspection logic or compliance metadata
- Legal matelines where product liability transitions to site-built responsibility
- Certified limits or disallowed compositions

Because **product-external rules** are implicit, tacit, or proprietary, digital tools strip these rules away, leaving only geometry. A bathroom pod becomes a 3D box. A façade cassette becomes a surface. A utility wall becomes a drawing.

The absence of a rule-native file type forces the industry to treat products as drawings with attributes—not as systems with legally and technically bounded configuration spaces.

2.5 The Resulting Silo Problem

ETO software cannot carry product deployment rules, so every stakeholder recreates them manually:

- Manufacturers encode constraints inside proprietary configurators.
- Architects approximate constraints in BIM families.
- Contractors translate constraints into spreadsheets, RFIs, and installation notes.
- Inspectors interpret constraints through bespoke documentation.
- Configurator developers re-encode constraints for each vendor, each time.

Each reconstruction is incomplete. All are incompatible. None are authoritative.

This fragmentation guarantees that:

- No multi-vendor configuration is possible.
- No shared interface logic can emerge.
- No automated validation can occur.
- No AI tool can generate safe solutions.
- No product ecosystem can scale.

Rules live everywhere except in the file types—therefore every digital environment becomes a silo, and every silo becomes a barrier to the CTO marketplace.

[9] Expert users may reproduce *some* of this functionality with current BIM tools, but not without advanced experience, bespoke coding, significant trial-and-error, and producing results format for understanding across the AEC industry.

2.6 The Coordination Trap: Why ETO Prevents Rule-Exchange Across Companies

A deeper structural problem under ETO is that no entity can publish authoritative rules for how its products must connect to others, because the ETO administrative stack never created a place for such rules to live. In a service-based system, every building interface condition is trade-based (i.e. slab edges, fire separation planes, plumbing rough-ins, electrical stubs, etc.), and re-authored on every project. These conditions are expressed through representations: drawings, details, RFIs, shop drawings, and field directives. They are therefore project-specific interpretations, not stable product definitions. Because rule-definition is downstream, product logic cannot be exchanged upstream, and no two companies can rely on each other's constraints.

This traps the entire offsite construction industry.

- A pod manufacturer cannot digitally express its required service zones; a façade manufacturer cannot express its anchoring dependencies;
- a railing supplier cannot digitally express its attachment geometry relative to a slab edge; and
- a curtain-wall manufacturer cannot digitally share the configuration space that its mullions, brackets, and anchors must operate within.

Every rule that matters must be rediscovered, translated, negotiated, and re-approved on every project. Nothing is portable. Nothing is reusable. Nothing is authoritative outside the project that spawned it.

In technical terms, the administrative stack suppresses *cross-vendor interoperability*, because there is no mechanism to globally express usage rules. Manufacturers cannot publish machine-readable usage contexts; architects cannot receive them; contractors cannot validate them; inspectors cannot trust them. These rule sets sit outside any digital file type and instead dissolve into the narrative record of ETO documentation environments. The industry therefore lacks the preconditions for a functioning marketplace: shared definitions, shared interfaces, and shared constraints.

This is why previous attempts at offsite construction marketplaces failed. They may one day offer catalogs of products but with no system for exchanging each product's rule layers, delivery methods are still mired in ETO interpretation services. Without machine-readable interfaces, no marketplace could compose multi-vendor systems, validate compatibility, or create trustable combinations. The result continues to be superficial choices without true interoperability.

A CTO marketplace requires the opposite logic: manufacturers must define rules upfront; those rules must be published in a common format; and those rule sets must travel with the product, not with the project team. Only a rule-native file type can break the coordination trap and enable products from different companies to interoperate with confidence.

2.7 Chapter Conclusion

What offsite construction currently lacks is an equivalent ecosystem for a **Configure-to-Order** (CTO) world: a parallel stack of contracts, classifications, interfaces, and roles designed for repeatable products, interstate movement, and UCC-style treatment of goods. The project-by-project workarounds now used for modular deliveries are not yet that system. The work described in this paper (hybrid contracts, legal matelines, standardized roles, and data-driven documentation) is aimed at seeding that missing stack so

that statutory review, when it comes, can see a complete and testable alternative, not just isolated experiments.

3. From Engineer-to-Order to Configure-to-Order: The Structural Shift Underway

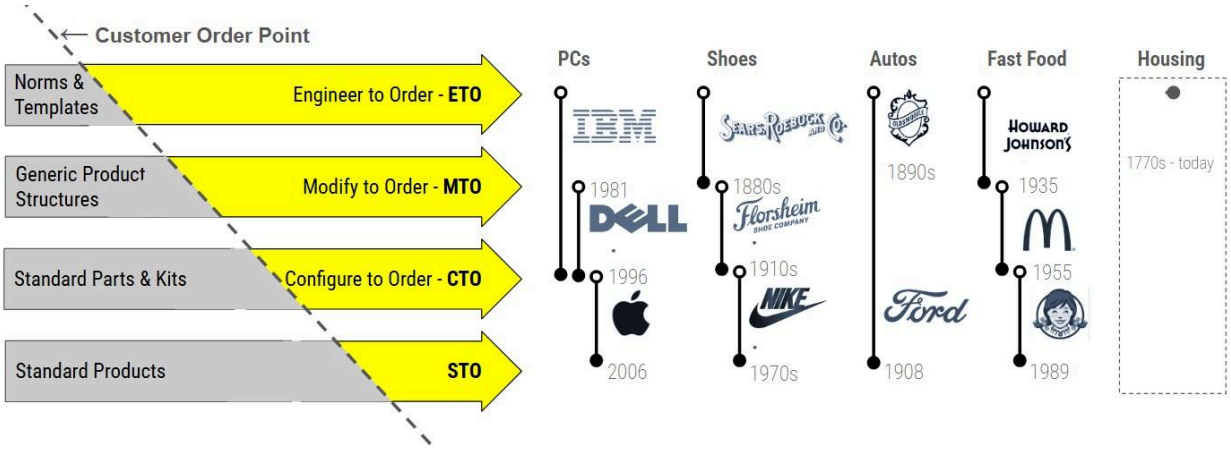


Figure 3a. From Engineer-to-Order (ETO) to Configure-to-Order (CTO). *Image adjusted from Jensen, Patrik.* Configuration of Platform Architectures in Construction*. Doctoral Thesis (2014). This diagram illustrates the structural transformation at the core of the CfOC's Vision: moving from bespoke, one-off project delivery to a platform-based ecosystem of standardized, interoperable components.*

- *In ETO, clients are at maximal financial risk, because each building is engineered from scratch, requiring sequential workflows and ad-hoc supply chains.*
- *In CTO, such risk is dramatically reduced, since buildings are made within a pre-engineered product platform, enabling parallel production, and predictable performance.*

CTO separates two distinct forms of design: (1) ***product design****, performed by engineers authoring pre-certified components, and (2)* ***building configuration****, performed by designers, configurators, or agents assembling buildings from those pre-engineered components.*

A mature **Configure-to-Order (CTO)** marketplace operates on a fundamentally different logic than today's **Engineer-to-Order (ETO)** environment: it is built around pre-designed, pre-certified products that can be specified from catalogs and installed without professional oversight. Hallmarks of CTO include.

- Manufacturers placing complete, code-compliant sub-products into larger compositions with defined scopes, tolerances, and warranty terms, *instead of relying on bespoke drawings authored by licensed design professionals for every assembly,*
- Individual products sold with credentials already attached, *instead of depending on code inspectors to perform quality-control review services.*
- Individual products designed to connect instantly through shared dimensional interfaces so that building products produced by different manufacturing firms interoperate on the same project, *instead of requiring a collection of project-specific designers to administer all scales of coordination with planning and negotiation.*
- Customers (key: not clients) configuring their larger project solutions from digital catalogs, *instead of beginning with a unique design, and inventing a bespoke project delivery plan from scratch*.

The CTO shift turns construction from a one-off engineering-and-logistics exercise into a repeatable act of configuration, whose interoperable products are the precondition for scale, reliability, and market liquidity.

Every other mature industrial sector has already solved these interoperability problems through external certifications and standardized interfaces, demonstrating the path construction must follow:

- Electronic components circulate globally because products carry UL listings, ETL marks, CE conformity declarations, or CSA certifications, which guarantee safety and performance without requiring each project team to re-engineer or re-inspect them.
- Digital devices interoperate because of universal interfaces such as USB-C, HDMI, Bluetooth, and Wi-Fi, all of which allow components from competing manufacturers to plug into shared ecosystems instantly.
- Defense and transportation sectors depend on standard interfaces (NATO STANAG mountings, ISO shipping container geometries, SAE automotive connectors) that allow equipment, containers, and vehicles to intermix seamlessly.

These precedents show that once a market agrees on certifications and interfaces, innovation accelerates: firms invest in better products rather than reinventing basic compatibility on every project. Offsite construction remains an outlier only because these shared rules do not yet exist.

Here are examples of CTO-enabled workflows applied to distinctive offsite construction-like circumstances:

- A homeowner installed her unique IKEA™ kitchen, after ordering from an online configurator of pre-designed elements, and assembling the cabinets from flat-packed boxes.
- A child assembled his unique LEGO™ playscape, after ordering several pre-designed sets, and assembling a larger diorama of individually-sold box sets on a stud-scape plate.

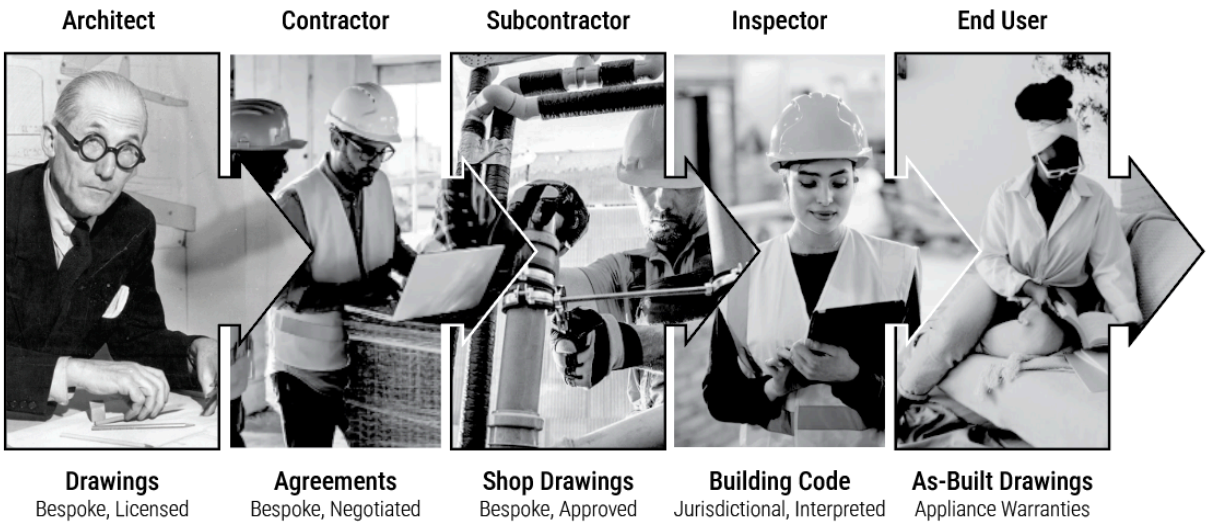
Figure 3b: Installing IKEA kitchen cabinets on a wall-mounted steel cleat.
Screen-shot from IKEA METOD Kitchen Installation 3/7 - Installing the cabinets.

Figure 3c: Child installing LEGO sets on a large LEGO-stud slab.
Screen-shot from blog "Life with My Little," post titled "How to Save and Protect Your LEGO Building Instructions".

The legal and economic implications of this shift are profound.

- Under an ETO regime, the central question is, *"How well is the service performed?"* — whether the design met the standard of care, or the workmanship met specifications.
- Under a CTO regime, the central question becomes, *"How well do the products perform?"* — whether it conforms to the agreed arrangement, warranty, or certification.

ETO Roles for the AEC industry



CTO Roles for the AEC industry

Figure 3d: The roles in a CTO workflow are reduced, less complex, and easy to adopt.
Reliance on pre-engineered products that work interoperably with industry-standard interfaces allow for simplicity and speed.

The US offsite construction space is already undergoing a disorganized shift to CTO tools and methods.

- Private firms are making proprietary interfaces, like the Z-Block connector (Z-modular) and the Cloud Connect (Cloud Apartments).
- The Center for Offsite Construction is working to organize open-source industry-standard interfaces[10] as the only ANSI Accredited Standards Developer dedicated to the North American offsite industry[11].
- The International Code Council (ICC) has developed *Appendix N* to provide individual jurisdictions with a means of incorporating guidelines for replicable buildings into their building code adoption process. Individual state adoption has been uneven and slow.[12]
- The ICC and Modular Building Institute (MBI) have formed the ICC/MBI-1200 to provide standard requirements for planning, designing, fabrication, transportation, and assembly of off-site construction, and ICC/MBI-1205 to provide requirements for inspection, approval, and compliance practices of off-site construction. Both require individual state adoption, which has been uneven.[13]

[10] See "Modular interface Standards" here and here.
[11] See "Vision for Offsite Construction Standardization" in the CfOC's .edu site.
[12] See this adoption table, compiled by MiTek.
[13] ibid.

Moving from the ETO to CTO requires more than these individual steps, as a fundamental reorganization of how vast portions of the Architecture, Engineering, and Construction (AEC) industry conceives, designs, contracts, and delivers the built environment.[14]

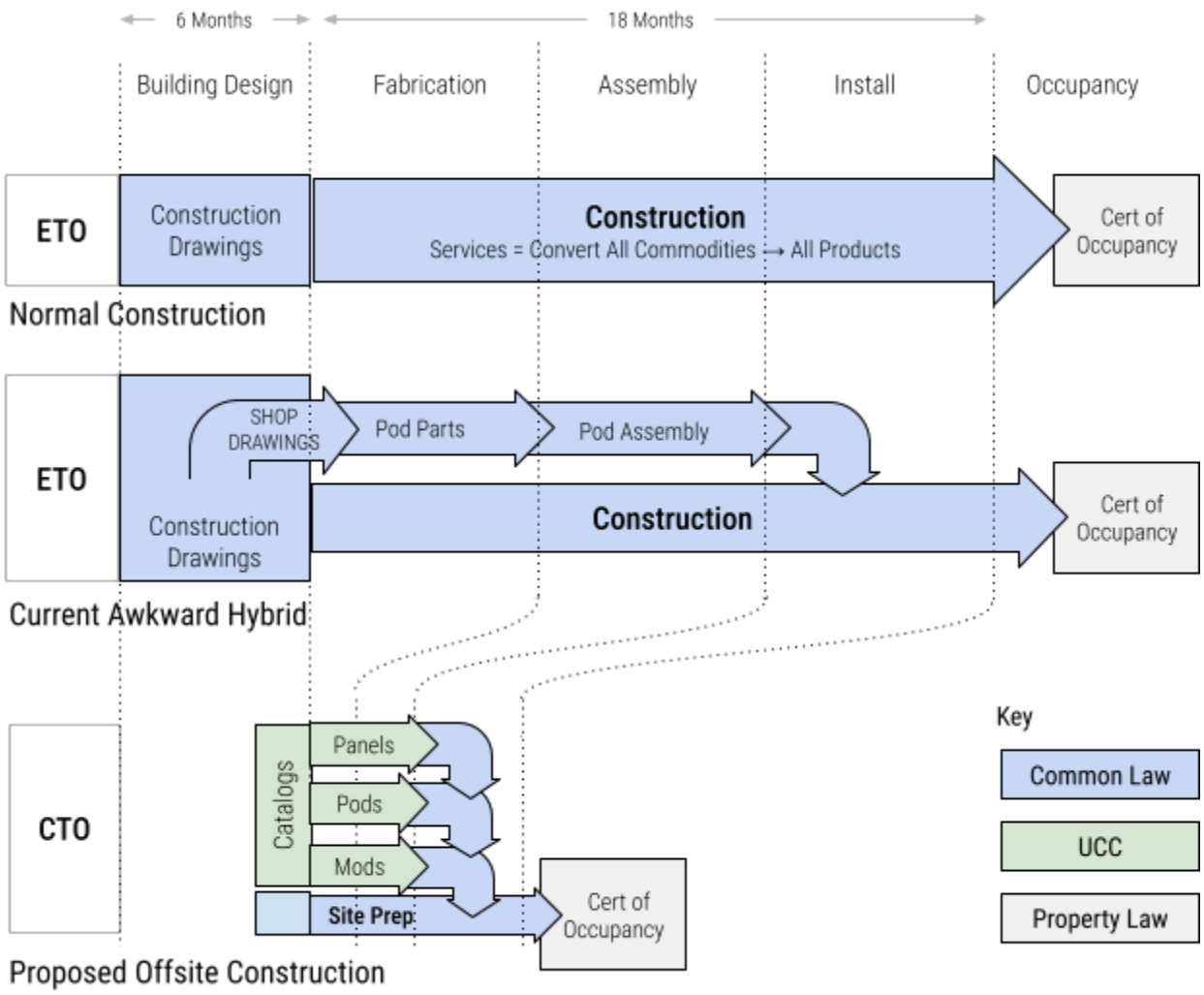


Figure 3e: Promotion of commodities into housing Products, with the on site and offsite labor scope of ETO vs CTO.
Note that all Services are trade-based in Construction, delivered on-site at a maximum expense.
In contrast, Products convert Services into an MSRP, offsite.

In commercial terms, this move marks a legal shift from contract-for-service (Common Law) to contract-for-goods (UCC). In software terms, this move marks a digital shift from static product descriptions to dynamic, rule-based product platform interactions.

In other industries, CTO frameworks have been supported by uniform commercial law and digital interoperability standards. In automotive, aerospace, and electronics manufacturing, for example, buyers configure products within predefined technical limits, and sellers guarantee conformity through certification, not interpretation. Contract law in those fields evolved accordingly, supported by data standards that enable transparency and verification.

[14] For a complete map, visit the Future of Design & Delivery's *Modes of Delivery* section.

Why the ETO-to-CTO shift is Urgent: Housing, Sustainability, and Labor

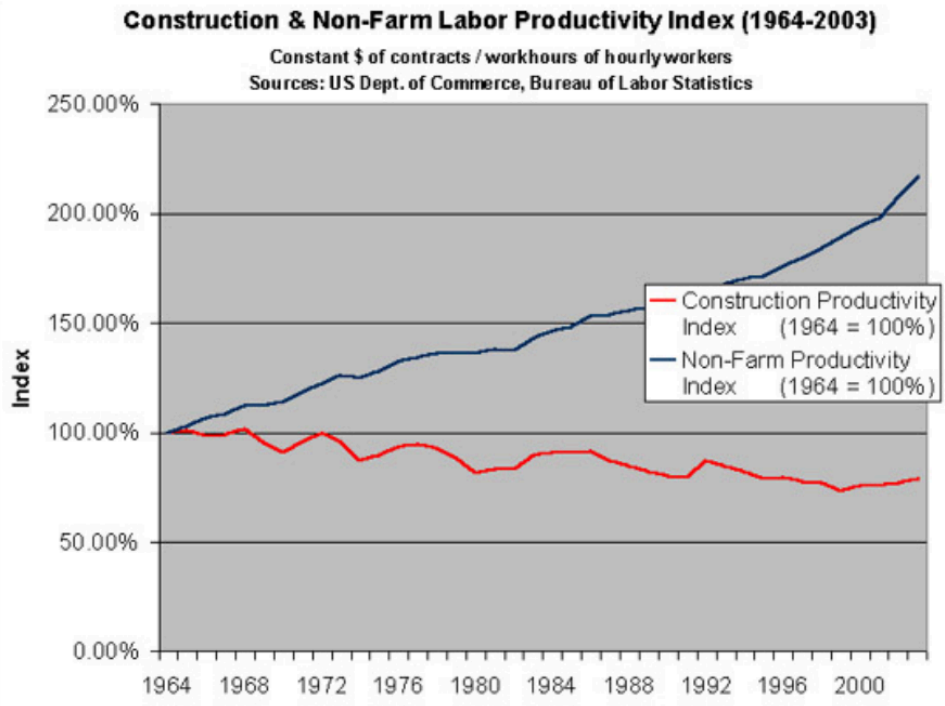


Figure 3e: Teicholz Construction Productivity Index Graph*. (Paul Teicholz 2013)*
Indexes of labor productivity from construction and non-farm industries, 1964-2004, indicates no productivity gain in the construction industry.

The economic consequence of remaining in an ETO paradigm is visible in national productivity data. The Teicholz Productivity Index (above)[15] shows that while productivity in most non-farm industries has more than doubled since the 1960s, construction productivity has remained flat for half a century — a signature symptom of the ETO model's limits. In contrast, industries that adopted CTO production (on a pathway of further advancement), converted design-and-delivery repetition into exponential efficiency gains.

These are gains that construction has yet to realize, and that stand at the heart of today's nationwide affordable housing challenges.[16] This efficiency explains why manufactured homes are still found to be much lower-cost, averaging $72 per ft^2, or just over half the site-built home average of $144 per ft^2.[17]

Below is an expanded list of benefits attainable, when converting design repetition into exponential product-based efficiency gains — precisely the kind of repeatable reliability that configurators and standardization is meant to protect.

---

[15] Teicholz, P. M., Goodrum, P., & Haas, C. (2001). *U.S. Construction Labor Productivity Trends, 1970-1998*. Journal of Construction Engineering and Management, 127(5), 427-429. DOI: 10.1061/(ASCE)0733-9364(2001)127:5(427)
[16] For more, see Alexandrov, A., & Goodman, L. (2024, January). *Place the Blame Where It Belongs: Lack of Housing Supply Is Largely Responsible for High Home Prices and Rents*. Washington, DC: Urban Institute, Housing Finance Policy Center. https://www.urban.org
[17] *Comparison of the Costs of Manufactured and Site-Built Housing*, Joint Center for Housing Studies Harvard University (2023), pages 4-5.

1. **Productivity Gains (Economic Competitiveness)**[18] The U.S. construction sector has lagged behind other industries in productivity growth for decades. CTO aligns construction with the manufacturing playbook that drove productivity gains in automotive, aerospace, and electronics: repeatable processes, precision tooling, and supply chain integration.

2. **Environmental Performance (Sustainability)** Industrialized production allows for tighter material control, less waste, and more efficient energy usage. Mass-configurable products can be optimized for lifecycle performance (from embodied carbon to operational efficiency) in ways that bespoke ETO projects rarely achieve.

3. **Risk Reduction (Reliability and Warranty Clarity)** Pre-defined configuration systems reduce the number of unique failure modes. Components are tested in controlled environments, defects are caught before deployment, and maintenance regimes can be standardized across a fleet of buildings.

4. **Scalability (Housing Supply Impact)** CTO enables the parallelization of design and production. While site work proceeds, modules and assemblies are fabricated offsite, collapsing overall delivery schedules. This parallel workflow is almost impossible in an ETO environment.

Having traced the operational and economic rationale for CTO production, the next question is tool-based: can a configurator file type (and CTO-based project definitions) help to provide a coherent framework for this new mode of building delivery?

[18] For decades, these benefits of offsite methods have been extolled at a surface-level by proponents: Labor relief, Time savings, Lower costs, Quality and climate control, and Safety. Some examples: Certainteed, Georgia Pacific, AECOM, the ICC, etc.

4. The Need: Preparing New Tools for the CTO Marketplace

Across manufacturing sectors the transition from bespoke engineering to **Configure-to-Order** (CTO) marketplaces has reshaped how products are designed, sold, and assembled. The same transition is now emerging in construction. Its early signals are visible everywhere: larger pre-certified components, developing interoperable interfaces, whole product-platforms.. all signalling the rise of offsite methods as a scalable alternative to site-built labor.

At the heart of CTO is a simple idea: individuals assemble solutions from catalogs of pre-designed, pre-certified products. Configuration replaces authorship.[19] Selection replaces drafting. Validation replaces interpretation. This logic underpins familiar consumer experiences—from IKEA kitchens hung on a steel rail, to children building custom playscapes from multiple LEGO sets on a studded baseplate. In both cases, the interface limits the universe of valid combinations. The product platforms embed the allowable variations. And the user, empowered by those constraints, can assemble countless configurations without specialized training.

Construction is heading toward the same destination. Modular pods, MEP assemblies, structural panels, façades, and interface-ready subassemblies are already proliferating. Each manufacturer has its own catalog and its own internal logic. But the industry lacks the common substrate that makes a true marketplace possible: a standard way to describe those products, their rules, their interfaces, and their allowable variations.

Without that substrate, the industry defaults to bespoke digital handoffs. Architects still produce drawings and BIM models. Manufacturers still translate those drawings into proprietary geometry. Contractors still reconcile differences between catalog products and architectural intent. The workflow remains ETO in everything but branding.

But once a common product definition exists—machine-readable, rules-aware, interface-aware—the dynamics of the industry change. Design becomes an act of selecting and assembling productized elements. Configurators can safely automate design exploration. Builders can prove feasibility before committing capital. Code officials can validate configurations rather than redlining drawings. Developers can compare manufacturers on function, cost, performance, and speed. And AI agents, operating within defined constraint-spaces, can generate only valid building proposals rather than unconstrained geometry.

The CTO marketplace also depends on version-controlled, vendor-maintained catalogs, not static imports. Manufacturers must be able to host their product definitions on their own servers — updating geometry, rules, tolerances, and certifications without requiring every configurator platform to be manually re-synced. A future configurator should not store local copies of every product; it should simply ask each vendor: “What is the current version of your product, and what are its rules?”

This ecosystem—dynamic, linked, versioned, and rules-driven—is impossible without a standard file type. Without a common substrate, configurators will remain fragmented. Vendors will remain walled off. AI tools

[19] This shift is only possible because the design work happens upstream in CTO: each product undergoes a product design stage, where its geometry, interfaces, and rules are fixed before any project exists.

will be unsafe. And the dream of a multi-vendor, platform-driven construction marketplace will remain out of reach.

4.2 What a CTO Marketplace Requires Instead

A CTO ecosystem needs products that describe themselves—not as geometry, but as **rule-bounded, interface-ready components** with defined configuration spaces.

It requires a new substrate:

- A file type capable of storing constraints, interfaces, parametric ranges, disallowed conditions, dependencies, and provenance.
- Software effective in determining whether a product instance's placement or context violates any non-critical configuration rules.
- A structure enabling configurators (either human or AI) to assemble products safely and predictably.
- A medium in which manufacturers publish authoritative definitions, and every downstream tool consumes them without reinterpretation.

Only with a rule-native file type can the industry shift from bespoke authorship to interoperable composition.

The industry needs a file type that makes CTO possible.

- A file type that makes configuration safe.
- A file type that makes automated design trustworthy.
- A file type that allows products to interoperate—no matter who made them.

The need is clear: construction must standardize the way products describe themselves.
Only then can the CTO marketplace emerge.

5. The Solution: A New Configurator File Type for the CTO Marketplace

The AEC industry requires a new digital substrate… one that finally unifies the geometry, rules, constraints, and interface logic of offsite construction products into a single, machine-readable format. We refer to this as the **Configurator File Type**: an open, vendor-neutral specification capable of supporting both today's transitional workflows and tomorrow's fully automated, agentic design systems.

The Configurator File Type does not describe a building; it describes a product. It codifies the results of the product design stage (i.e. geometry references, interface logic, allowable configuration ranges, dependencies, and certifications,) so that building configuration can occur safely and automatically downstream.

The purpose of the file type is not to replace BIM or MCAD. It is to sit above and across those tools, capturing the essential information they cannot store: the logic that governs how products behave inside a configuration space. In effect, it transforms each product into a self-describing object whose constraints are explicit rather than implicit.

A configurator cannot assemble a building from geometry alone. It must know which edges are connectable, which faces carry utilities, which dimensions can vary and within what ranges, which combinations are allowed or prohibited, and what the adjacency rules are between products. The file type becomes the container for that knowledge.

A product's Configurator File is authored once during the product design stage; it is consumed repeatedly during building configuration.

At minimum, each product's Configurator File must encode:

- **Geometry** at the appropriate level of detail for configuration
- **Interfaces** — locations, types, and tolerances (aligned with ANSI/CfOC 1220 modular interface standards as they develop)
- **Parametric Ranges** — allowable variation and flexion of the product (e.g., panel lengths, pod orientation, finish variants)
- **Composability Constraints** — rules that determine legal and illegal combinations
- **Dependencies** — requirements for supporting products, utilities, or clearances
- **Performance Constraints** — code-relevant properties (fire, acoustic, structural, accessibility)
- **Certifications** — UL, CE, third-party approvals, factory certifications
- **Metadata for AI/AEC Use** — identifiers, provenance, versioning, warranty conditions
- **Mateline Definitions** — the legal/technical seam where product governance transitions into site-built responsibility

These components transform the product from a passive geometric object into an *active participant* in *automated configuration*. Once these rules are expressed in a standard schema, any tool (not just the manufacturer's own configurator) can safely assemble products into valid building systems.

This file type must also support two operational modes:

1. As an Overlay to Existing ETO Tools (Today)

In the near term, the file type should be linked to, not embedded within, existing BIM or MCAD models. This allows design teams and manufacturers to continue using Revit, Inventor, MicroStation, CATIA, Fusion, or proprietary fabrication tools without disruption. The file type becomes a "wrapper" that references geometry stored elsewhere while adding rule-based intelligence not available in ETO environments.

This transitional mode is critical. It allows manufacturers to publish product definitions incrementally, aligns with the realities of current production systems, and avoids requiring firms to rewrite their digital infrastructure. As noted in the transcript, the file type should be able to *link to* vendor-hosted models rather than requiring a central repository. This ensures version control remains with the manufacturers and reduces the maintenance burden on any future marketplace or configurator.

2. As a Kernel for Agentic Workflows (Future)

Once rules, constraints, and interfaces are standardized, the file type becomes the basis for fully automated configuration. AI agents—trained not on architectural geometry but on product constraints—can explore complete configuration spaces safely. Parametric variation becomes bounded and verifiable. Configuration engines can compose entire buildings from product catalogs without human drafting.

This is the missing tool that allows agentic design to become trustworthy. Without explicit limits, AI cannot be relied on to generate valid or buildable outcomes. With explicit limits, genetic algorithms, search methods, rule engines, and agentic reasoning systems can produce solutions that conform to manufacturer specifications, zoning constraints, building codes, and interface requirements.

And critically, the constraints come *from the products themselves*, not from a proprietary engine. Any configurator—whether developed by a manufacturer, a general contractor, a third-party startup, or a research institution—can operate on the same rules.

Design Principles for the File Type

The solution must be shaped by several core principles:

- **Openness** - The format must be open and published, owned by a neutral body (such as the CfOC in its ANSI-accredited capacity), ensuring that no single vendor controls the marketplace.
- **Interoperability -** The file type must work across Revit, Inventor, MicroStation, CATIA, IFC, USD, and emerging formats—recognizing that geometry will continue to originate in diverse environments. The file type governs the *rules*, not the *modeling environment*.
- **Vendor Control of Data -** Manufacturers maintain their own catalogs on their own servers. The file type points to versioned geometry hosted by the vendor—not a centralized repository. This ensures accuracy, reduces liability, and prevents data from becoming stale.
- **Versioned and Traceable -** Every configuration produced from a configurator should reference the exact versions of product definitions used at the time of configuration—critical for warranties, code approvals, and manufacturing reliability.

- **Composable -** Products defined in the file type must be able to discover and align with one another based on rules, interfaces, and allowable adjacency. This composability is the backbone of the CTO marketplace.
- **Scalable -** The file type must support simple products (e.g., flat panels) as well as highly complex assemblies (e.g., multifamily bathroom pods with embedded MEP systems), ensuring broad adoption across the AEC sector.
- **AI-Native -** The file type should be designed from the outset to be readable and operable by agentic systems—using clear rule expressions, constraint structures, and interface definitions.

What an Open File Type Enables

With a shared Configurator File Type in place, the industry will finally have the substrate it needs for:

- Multi-vendor configurators
- Automated building configuration with constraint-aware agents
- Rapid feasibility studies
- Product comparison across vendors
- Configurable, code-ready building platforms
- Linked catalogs updated by manufacturers in real time
- Safe, bounded design exploration
- Data transparency without data centralization

It is the file type that makes the CTO marketplace possible. It transforms design from drafting to configuration. It unlocks automation. It reduces rework. It increases clarity. And it finally allows products—not drawings—to become the atomic units of the built environment.

6. The CTO Marketplace: Linked Catalogs, Configurators, & Multi-Vendor Composition

A shared Configurator File Type does more than standardize product definitions. It unlocks an entirely new ecosystem for the AEC industry—one in which manufacturers, configurators, designers, builders, inspectors, and even AI agents can interact through a common substrate of rules, interfaces, and constraints. This ecosystem is not a single platform, nor a centrally controlled marketplace. It is a network of interoperable catalogs and tools that together support the first true **Configure-to-Order** (CTO) marketplace for buildings.

The core idea is simple: *manufacturers host their product definitions; configurators fetch them; and users configure buildings from dynamically updated catalogs.* This is a decentralized, vendor-controlled, versioned ecosystem—more similar to Git, npm, or package repositories than to any existing AEC software model.

At the center of this ecosystem is a division of responsibilities:

- **Manufacturers** maintain authoritative models and rule files on their own digital storage and display.
- **Configurators** read those definitions, interpret the rules, and compose valid building solutions.
- **Designers and builders** select, configure, and assemble productized systems rather than drafting geometry.
- **Code officials and inspectors** validate configurations against the rules embedded in the file type.
- **AI agents** operate inside clearly defined constraint spaces, generating only permissible solutions.

In this ecosystem, the act traditionally called ‘building design’ becomes the *building configuration stage*: assembling pre-designed products under the constraints authored upstream by manufacturers.

This ecosystem is possible only when product definitions are expressed in a common format. Once the rules are standardized, vendors no longer need to build proprietary configurators: *the rules themselves become the interface.*

1. Manufacturer-Hosted Catalogs: A Linked, Versioned Substrate

In the envisioned system, each manufacturer maintains its own online catalog of Configurator File Type definitions—akin to a continuously updated repository. These catalogs may contain:

- Geometric data
- Interface definitions
- Parametric limits
- Variant logic
- Certifications and code mappings
- Version history

- Disallowed or deprecated configurations

The critical principle is that these definitions are *linked, not uploaded*.

Instead of distributing geometry files, vendors provide a reference—a URL pointing to the authoritative definition. Whenever a configurator queries the catalog, it retrieves the most current version. If a product definition changes or is updated to meet new code requirements, every configurator that uses that product automatically receives the update.

This model avoids the failures of past AEC exchanges, where stale objects circulated endlessly. It prevents outdated or deprecated components from being embedded in new designs. It relieves configurator developers from maintaining massive libraries of geometry and rules. And it gives manufacturers full control over their data, reducing liability, maintenance cost, and version confusion.

2. The Configurator Layer: From Design Tools to Assembly Engines

Configurators—whether built by manufacturers, general contractors, insurers, code officials, developers, or third-party startups—become the engines that assemble products into valid building systems. Unlike today's proprietary configurators, they operate on *shared rules*, not private logic.

In this context, a configurator:

- Reads product definitions in real time
- Discovers connectable interfaces across components
- Validates combinations based on encoded constraints
- Selects among parametric ranges
- Enforces performance and code requirements
- Produces configuration outputs (BIM, fabrication files, pricing, schedules)
- Records the full provenance of the configuration (product versions, metadata, code mappings)

This transforms the configurator from a design tool into a **configuration engine**—one that enables, rather than replaces, cross-vendor interoperability.

Importantly, configurators do not need to be monolithic. Some will optimize cost; some will prioritize energy performance; some will focus on affordable housing typologies; some will specialize in interior fit-outs; some will integrate agentic reasoning systems to generate thousands of valid variations.

The common substrate ensures that **every configurator speaks the same product language**, even if their goals differ.

3. Multi-Vendor Assembly: The Marketplace Emerges

Today's offsite products cannot interoperate because their rules are trapped in proprietary tools. Once rules are standardized, multi-vendor assembly becomes straightforward:

- A pod from Vendor A snaps into the structural grid of Vendor B.
- A façade cassette from Vendor C aligns with the envelope geometry of Vendor D.

- Panels from three manufacturers compose a building core using shared interface logic.
- Utility-bearing components automatically align with mateline definitions.

The marketplace is not a website. It is the *ability of products to be composed safely across vendors*.

Once a manufacturer adopts the file type:

- Their products become discoverable by any configurator.
- Their interface logic becomes interoperable with every compliant product.
- Their catalog becomes part of a larger ecosystem without giving up data control.

This dynamic creates the same network effects that transformed other industries. Manufacturers gain more users; configurators gain more components; builders gain more options; AI agents gain more valid search spaces.

The file type creates a gravitational pull toward standardization—not through enforcement, but through opportunity.

4. Version Control and Provenance: Trustworthy Configuration

Each time a building configuration is generated, the configurator captures the exact versions of every product used:

- Wall Panel Model A — v1.07
- Bathroom Pod B — v2.31
- Façade Module C — v0.98
- Interface Pack D — v1.14

This provenance trail is essential for:

- **Code Review** — ensuring the configuration uses currently certified products
- **Manufacturing** — reducing errors from mismatched versions
- **Warranties** — clarifying the exact product variants installed
- **Maintenance** — identifying compatible replacement components
- **Liability** — demonstrating that products were used according to their rules
- **Finance** — simplifying lender's risk, and how it is limited to products, not services

In effect, the file type becomes the foundation of a **chain-of-custody** for productized construction.

5. Agents in the Loop: Safe Automation

AI tools cannot safely generate detailed, technical building configurations today because they lack explicit industry-adopted interfaces, dependencies, and constraints. Once the file type standardizes those constraints:

- Agents can explore the configuration space safely
- Every generated solution is valid by construction
- The search space is bounded by product logic, not hallucination
- Outcomes become predictable, inspectable, and code-aware
- Human oversight shifts from drafting to selecting among valid options

This transforms AI from a novelty into infrastructure.

The configurator marketplace becomes the domain in which agents operate—not as replacements for human designers, but as collaborators generating only permissible solutions.

6. A Decentralized, Durable Ecosystem

The full system thus consists of:

1. Manufacturer-hosted catalogs
2. A universal Configurator File Type
3. Third-party configurators and agentic tools
4. Linked, versioned product definitions
5. Multi-vendor compositional logic
6. Provenance-rich configuration outputs

This is not a single marketplace—it is a network of interoperable marketplaces.
It is not a single configurator—it is a generation of them.
It is not a single product platform—it is every platform, connected through a shared substrate.

The result is a CTO ecosystem that preforms like other transformed industries (consumer electronics, aerospace, automotive, logistics) by shifting from assemblies to *products*, from interpretation to *selection*, and from bespoke decisions to *rule-based composability*.

The Configurator File Type is the foundation that makes this shift possible.

7. Implementation: From Concept to Standard

Developing a Configurator File Type is not a software project. It is a standards project—an effort to define a common substrate that manufacturers, configurators, researchers, and regulators can adopt without requiring permission from any single platform or vendor. Implementation therefore requires structured governance, deliberate outreach, and a phased plan that reflects the realities of the AEC industry. The CfOC, as an ANSI-accredited standards developer, is uniquely positioned to lead this work.

The implementation strategy rests on four pillars:

1. Governance through ANSI-compliant procedures
2. Iterative, stakeholder-driven standard development
3. Technical integration with existing toolchains
4. Demonstration projects and early-market pilots

Each pillar reinforces the others, ensuring that the file type becomes both technically robust and broadly adoptable.

1. Governance: JDF Stewardship Modeled after ANSI Accreditation

The CfOC was accredited by the American National Standards Institute (ANSI) to write open standards for modular interfaces. The same governance model applies to the Configurator File Type through collaboration with an organization like the Joint Development Foundation. Both ANSI & JDF processes require balanced representation, transparent decision-making, structured comment resolution, and public review. This ensures that no single vendor or platform dominates outcomes—a critical requirement for a file type meant to serve the entire AEC ecosystem.

The standards process includes:

- Formation of a balanced consensus committee
- Public call for participation
- Drafting of the initial specification
- Formal public review
- Structured comment resolution and revisions
- Publication of the first edition
- Ongoing maintenance under ANSI's continuous-review model

This governance framework provides legitimacy, durability, and the political neutrality needed to align competing stakeholders.

2. Stakeholder Engagement: Building a Coalition Around the Standard

Successful standards are not authored in isolation; they are stewarded into existence by communities. The CfOC will therefore convene a broad coalition of participants:

- **Manufacturers** of pods, panels, façades, MEP assemblies, and modular subassemblies
- **Software firms** developing configurators, parametric engines, or agentic design tools
- **General contractors and developers** who will rely on the file type for procurement
- **Code officials** and third-party inspectors who require trustworthy metadata
- **Researchers and academics** contributing parametric, geometric, or workflow expertise
- **Industry organizations** (MBI, ICC, CTOHA) who share governance interests

Stakeholders will contribute use cases, constraints, manufacturer needs, agentic-workflow insights, code-mapping requirements, and domain-specific rules. Their participation will ensure the file type reflects real-world conditions, not an idealized abstraction.

The transcripts correctly anticipate the need for "mission-statement-level clarity" to align these groups. The internal mission—*to standardize the rule-space of offsite products so that CTO workflows become safe, interoperable, and automatable*—becomes the anchor around which stakeholder engagement is organized.

3. Technical Development: A Phased Approach to the Specification

Technical development will proceed in defined phases, each expanding the scope of the standard.

Phase 1 — Scope Definition & Domain Modeling

- Identify the core elements: geometry references, interfaces, constraints, parametric ranges, composability rules, code/performance metadata, product provenance.
- Map how these elements intersect with existing AEC schemas (IFC, STEP, USD, Revit families, Inventor assemblies).
- Produce the *domain model* that serves as the backbone of the specification.

Phase 2 — Drafting the Specification

- Define the file structure (likely a JSON-, XML-, or binary-derived schema).
- Define rule-expression syntax for constraints, interfaces, and allowable configurations.
- Define versioning and provenance requirements.
- Define the linking model (vendor-hosted catalogs; no centralized data storage).
- Release an initial draft for committee review.

Phase 3 — Integration with Existing Toolchains

The file type is designed as a *wrapper* or *overlay* on existing models in the near term. Implementation tasks include:

- Defining how to reference geometry stored in Revit, Inventor, CATIA, or IFC
- Defining lightweight exporters/importers for major toolchains

- Ensuring the wrapper approach does not require rewriting manufacturing CAD workflows
- Allowing the file type to operate independently of any proprietary modeling platform

This ensures low-friction adoption and a clear transitional path into agentic workflows.

Phase 4 — Public Review, Comment Resolution, and Finalization

Following ANSI (and similar JDF) requirements, the CfOC will:

- Release the draft specification for public comment
- Conduct multi-round comment adjudication
- Produce a revised specification
- Prepare the first edition for ANSI publication

This ensures industry legitimacy and stability.

Phase 5 — Pilot Implementations

Early pilots serve three purposes:

1. Validate the technical specification
2. Prove interoperability across vendors
3. Demonstrate workflow improvements for CTO environments

Potential pilots include:

- A multi-vendor bathroom pod + panel assembly
- A small multifamily configuration prototype
- An AI-driven configuration engine testing rulesets
- A code-official review mechanism operating on the file type's metadata

These pilots will generate real-world feedback, refine the specification, and illustrate the transformative potential of the standard.

Phase 6 — Long-Term Maintenance and Expansion

Once launched, the standard will require:

- Continuous updates
- New feature modules
- Interpretation documents
- Industry education
- Ongoing synchronization with evolving modular interface standards (e.g., CfOC–ICC 1220)

The standard is treated as a living document—evolving alongside the CTO marketplace.

4. Why the CfOC Is Positioned to Lead

Several conditions converge to make the CfOC the natural steward of this standard:

- It is the only ANSI-accredited organization focused specifically on offsite construction.
- It is independent of major software vendors, ensuring neutrality.
- It has direct relationships with manufacturers, contractors, and regulators.
- It has already demonstrated industry consensus-building through the Modular Interface Standard with ICC.
- It is academically rooted, allowing it to convene experts across technical, legal, and industrial domains.
- It is building a research continuum—from product-platform theory to interface standards to legal matelines—that provides the conceptual foundation for this work.

The Configurator File Type is the next logical step in the CfOC's mission: establishing the infrastructure layer for a national CTO marketplace.

5. Roadmap: A 24-Month Path to the First Edition

A realistic implementation timeline includes:

- **Months 1–3:** Committee formation, domain model, mission definition
- **Months 3–9:** Draft specification (v0.1 → v0.5)
- **Months 9–12:** Vendor consultations, tooling prototypes, integration tests
- **Months 12–15:** ANSI public review
- **Months 15–18:** Comment resolution; v1.0 preparation
- **Months 18–21:** Pilot implementations; refinement
- **Months 21–24:** Publication of the first edition

This timeline balances urgency with rigor.

6. The Outcome: A Standard That Makes the CTO Marketplace Real

Implementation is not about producing software. It is about establishing the **shared ruleset** that empowers software developers, manufacturers, and builders to innovate. When the file type is published:

- Manufacturers can publish interoperable catalogs.
- Configurator developers gain a common substrate for automation.
- AI tools can safely participate in a buildings' design, through a building configuration process.
- Developers and contractors can assemble multi-vendor buildings with reliability.
- Code officials gain clarity and confidence.

- The industry accelerates toward a CTO ecosystem grounded in shared standards.

The Configurator File Type is the infrastructure layer missing from offsite construction.
Implementing it is the work of standards, governance, and stewardship — not proprietary control.

The CfOC is prepared to lead this work.

8. Conclusion & Call to Action

The transition from Engineer-to-Order to Configure-to-Order represents a structural shift for the building industry—one that parallels the transformations that reshaped aerospace, automotive, electronics, and consumer manufacturing. Those sectors standardized their interfaces, codified their product rules, and created the substrates that enabled configurators, marketplaces, and automated toolchains to flourish. Construction has reached the same inflection point. Offsite products are proliferating; factory capacity is expanding; digital tools are maturing; and policy pressure is mounting for fast, scalable, affordable housing production.

What the industry lacks is the shared foundation—a common, rule-native file type—that allows productized building components to interoperate across vendors, platforms, and configurators. This whitepaper proposes the first step toward that foundation: a standardized Configurator File Type capable of describing products not only by their geometry, but by their rules, constraints, interfaces, and allowable compositions. With such a substrate in place, the next generation of CTO workflows becomes possible: multi-vendor building configuration, safe AI-assisted design, automated feasibility analysis, and transparent code review.

This work cannot be accomplished by a single firm or platform. It must be stewarded by a neutral body, governed through consensus, and shaped by a diverse set of contributors. As an ANSI-accredited standards developer with a national research mission, the Center for Offsite Construction (CfOC) is prepared to lead that process—but only with a coalition of partners who believe in a more interoperable, automatable, and scalable future for construction.

Join the Technical Effort

We invite *product engineers, software architects, computational designers, parametric modelers, AI researchers,* and *CAD/BIM specialists* to contribute to the definition of the file type itself. Your expertise in interfaces, variant logic, constraints, and manufacturing workflows is essential to crafting a standard that works across the full range of offsite products.

Support the Standardization Process

We seek *philanthropic funders, federal and state agencies, impact investors*, and *research sponsors* who recognize the value of shared digital infrastructure. Building a national standard requires convening power, administrative capacity, pilot projects, and iterative tools—none of which emerge without dedicated support.

Collaborate as Software Liaisons

We invite *AEC software firms, configurator developers, platform integrations teams, open-source contributors,* and *digital twin organizations* to participate as liaisons in shaping how the file type connects to existing toolchains. Your input will ensure the standard is practical, exportable, and compatible with both ETO-era environments and future agentic design systems.

Engage as Policy and Regulatory Partners

We welcome *state housing agencies, code officials, UL/CE certification bodies, federal technology accelerators*, and *uniform law organizations* to help align the file type with emerging regulatory frameworks. Your participation will ensure that product definitions carry the metadata, provenance, and code-relevant information needed for trustworthy permitting and inspection.

Participate as Industry Stakeholders

We encourage *manufacturers, general contractors, developers, modular builders, panelizers, MEP assembly firms*, and *affordable housing organizations* to help shape the real-world use cases and pilot projects that will validate and refine the standard.

The transformation to a CTO marketplace will not occur by chance. It will happen because this community—practitioners, manufacturers, technologists, policy leaders, funders, and researchers—builds the shared infrastructure needed to support it. A Configurator File Type is not merely a data standard; it is the foundation for a new way of delivering buildings: faster, safer, more interoperably, and at a scale commensurate with the nation's housing needs.

The CfOC stands ready to convene this work. Join us.

Glossary of Key Terms

Agent An autonomous software actor—often AI-enabled—that performs tasks within a bounded configuration environment defined by product rules, interface logic, and encoded constraints. In CTO, agents navigate a predefined rule space to assemble, evaluate, or optimize valid building configurations, governed entirely by the constraints authored by manufacturers, ensuring that every action the agent takes remains valid-by-construction.

Agentic Safety Envelope The bounded constraint-space within which agentic design tools may safely operate; defined by product rules, interface tolerances, code mappings, and performance constraints.

Agentic Configurator An automated system (typically AI-driven) that explores configuration spaces defined by product constraints, interface logic, and rule structures embedded in the Configurator File Type. Unlike generative AI operating on heuristic geometry, an agentic configurator works in a far more bounded CTO parametric range. It produces only valid-by-construction options because its search is bounded by manufacturer-authored rules.

Allowable Composition A machine-readable rule describing which product combinations are permitted or prohibited, including adjacency conditions, interface alignment, and environmental constraints.

Building Configuration Stage The downstream process—performed by designers, configurators, or agents—in which buildings are assembled from pre-engineered products. Unlike traditional "building design," configuration does not author assemblies from scratch; it selects and arranges products within predefined rule spaces.

Building Configuration Model A (CTO) representation of a specific building, containing geometry, datums, system families, and annotations – that is the product of a building configurator.

Building Design Model An (ETO) BIM-authored representation of a specific building, containing geometry, datums, system families, and annotations. It documents one instance of architectural intent, not the full rule space within which offsite products may operate. It reflects ETO logic: valid only in a particular project context and never authoritative for product deployment.

Building Product Design Model A MCAD- or fabrication-authored model that captures the geometry, sub-assemblies, bills of material, internal logic, and manufacturing details for an offsite product. In ETO, these models precisely express the internal composition of a product but do not express its allowable uses, interface logic, or configuration constraints, necessitating a higher-order rule-native file type.

Commissioning The verification step performed after offsite products are installed but before they are legally accepted. Commissioning validates that manufactured components (originally certified in controlled factory conditions) operate correctly after installation within site-built environments, closing the gap between factory-level compliance and as-installed performance.

Common Law A state-specific legal system rooted in judicial precedent that governs service-based construction contracts. It relies on project-specific interpretation, bespoke scopes, and case-by-case adjudication of performance. In ETO construction, Common Law structures labor jurisdictions, design responsibilities, inspection practices, and payment mechanisms.

Configure-to-Order (CTO) A product-based delivery model in which buildings are assembled from families of pre-engineered, interoperable components. Designers select among predefined, certified options rather than drafting bespoke assemblies. The system relies on standardized product rules, interface logic, and compatibility constraints, making repeatability, parallel production, and automation possible.

Configuration Space The mathematically bounded universe of all permitted combinations of products and variants within a CTO platform. The Configurator File Type encodes this space explicitly (i.e. parametric ranges, adjacency rules, interface tolerances) allowing configurators and agents to generate only valid solutions. Smaller configuration spaces require less compute power to traverse.

Configurator File Type The rule-native, open standard proposed in this whitepaper. It stores the geometry references, interfaces, constraints, allowables, parametric ranges, performance metadata, dependencies, mateline conditions, and provenance records that define how offsite products behave within a configuration environment. It is a wrapper around BIM/MCAD, not a replacement.

Dependency Rule A requirement encoded in the file type specifying what supporting components, utilities, clearances, or site conditions must exist before a product may be placed.

Disallowed Configuration Any combination of parameters, interfaces, or compositions that violate manufacturer rules, performance requirements, or code-based limits, explicitly encoded for safe automation.

Engineer-to-Order (ETO) A service-based project delivery system in which each building is engineered as a bespoke, first-principles prototype. All scopes are negotiated (from scratch or template); designs are interpreted through Common Law contracts, drawings, specifications, and professional judgment. Knowledge is rarely reused across projects, and digital tools (BIM, MCAD, inspection platforms) are optimized for *interpretation*, not for productized repeatability.

Geometry Reference A pointer within the file type to vendor-hosted geometry stored in BIM/MCAD environments, allowing the file type to remain lightweight and authoritative without duplicating source data.

IC Specifier A design professional (or consultant) responsible for selecting industrialized construction (IC) products early design phases. The IC Specifier matches project requirements with available offsite products (pod, panel, module, etc.), ensures selections conform to applicable codes and standards, and coordinates with manufacturers to maintain compatibility with the chosen product platform.

Interface Definition A geometric and semantic description of how products connect, that define surfaces, edges, tolerances, alignment rules, utility ports, and mateline conditions. Interface definitions allow products from different vendors to interoperate through shared rules.

Interface Pack A bundle of interface definitions—dimensional, utility, tolerance-based—that define a coherent connection family (e.g., a wall-pod interface pack). Used by multiple products across catalogs.

Mateline The physical and legal seam between productized and site-built scopes. At the product level, it marks where the manufacturer's obligations (geometry, tolerances, performance) end; at the legal level, it separates UCC-governed goods from Common Law services. Correct mateline definition is essential for warranties, risk allocation, and automated configuration.

Mateline Integrator A role responsible for coordinating all interactions across the product/site boundary. The integrator ensures that manufactured products align with the site-built environment, verifies tolerances, manages interface coordination, and documents legal transitions between Common Law and UCC scopes.

Offsite Construction Product (a.k.a. **Industrialized Construction (IC) Product** A pre-engineered, repeatable product (i.e. pod, panel, module, façade cassette, utility wall) manufactured offsite and governed primarily by commercial rules (UCC) until it becomes affixed to real property. These products depend on explicit interface definitions and allowable configuration ranges.

Parametric Range The permissible variation window for a product's parameters (dimensions, offsets, clearances, orientation options, finish variants). In the whitepaper, parametric ranges form a core part of machine-readable rule encoding. The sum of individual Parametric Ranges are the mathematical ingredients of a Configuration Space.

Platform Provider An entity that curates a product platform, including its catalogs, interface definitions, configuration rules, parametric ranges, and update cycles. Platform providers maintain compatibility, host versioned product definitions, and support configurator integrations.

Phantom CTO Marketplace The transitional stage in which teams behave as though a CTO market exists (i.e. catalog shopping, partial configuration, preselection of products) but without standardized interfaces, rule structures, or matelines. This "almost CTO" condition results in rework, misalignment, and recurring disputes.

Product Design Stage The upstream engineering process in which manufacturers define the geometry, interfaces, allowable variations, constraints, and certifications of an offsite product. This stage produces the authoritative product platform; its outputs are encoded in the Configurator File Type.

Product Platform (Offsite) A coordinated family of products connected through shared interfaces, variant logic, tolerances, and certified assemblies. The platform ensures that components remain interoperable over time, much like automotive or electronics ecosystems.

Provenance Trail An auditable record of the exact product definitions (including its versions, rule sets, interface packs) used in a configuration. Essential for permitting, inspection, manufacturing reliability, and warranty claims.

Rule Engine A computational layer that interprets the constraints in the Configurator File Type to determine allowable configurations, enforce interface rules, and validate assemblies. It may operate inside a configurator, an AI agent, or a code-compliance tool.

Uniform Commercial Code (UCC) The interstate commercial law that governs the sale of goods. In the context of offsite construction, the UCC provides a legal foundation for warranties, certifications, and goods-based risk allocation—serving as the legal backbone for CTO products.

Variant Logic Machine-readable rules governing optional elements, finish sets, orientations, or performance tiers that belong to a single product.

Vendor-Hosted Catalog A manufacturer-controlled repository where authoritative product definitions live. Configurators reference these catalog entries at runtime rather than storing local copies, ensuring that configurations always use the most current, certified product versions.

Versioned Product Definition A specific release of the product's rule file, including geometry pointers, constraints, certifications, and interface definitions—used to maintain historical accuracy in configurations.

Appendix A. Current File-Type Landscape in AEC

Today's digital environments rely on geometry-first file types with limited support for rule expression or inter-product interoperability. The Configurator File Type is not intended to replace these formats but to complement them by carrying the rule layer they cannot.

1. BIM Formats (Architecture & Engineering)

- **Revit (RVT, RFA):** Geometry + parameter storage, but no formal semantics for interfaces, constraints, or allowable compositions.
- **ArchiCAD (PLN):** Mature modeling environment; limited support for rule-based composability.
- **IFC:** Strong for building-wide data exchange; weak for product-level constraints and interface logic, especially for offsite products.

2. MCAD Formats (Manufacturing & Fabrication)

- **Inventor, SolidWorks, CATIA:** Excellent for detailed assemblies; poor for sharing rules or variants outside the native environment.
- **STP/STEP:** Strong interoperability for geometry; no embedded logic for composability, constraints, parametric rules, or interface metadata.
- **USD (Universal Scene Description):** Strong for scene composition; emerging interest in AEC; promising for geometry federation but not rule encoding.

3. Proprietary Configurators

- **Simpson Strong-Tie, Oldcastle, and many manufacturers** have configurators that embed rich rule logic—but rules remain trapped inside proprietary systems.
- **No shared substrate** exists for those rules, making cross-vendor configuration impossible.

4. Analysis Tools

- **Energy modeling, structural analysis, MEP coordination** tools depend on paths to geometry and metadata but cannot ingest product rules or allowable configurations.

5. Operations and Code Compliance Tools

- Fire, acoustic, ADA, and UL/CE certification tools all require data that is either unstructured or manually embedded in proprietary formats.

Gap Analysis:
Across all systems, the missing layer is clear: **none of these file types can express product-level rules, interfaces, constraints, or allowable compositions in a standardized, machine-readable format.**

The Configurator File Type fills this gap by serving as the rule substrate above these environments.

Appendix B. Cross-Industry Precedents: How Other Sectors Standardized Configuration

1. USB and USB-C Standards

The USB specification defines electromechanical interfaces, allowable use, limits, and negotiation rules. Devices from any manufacturer can interoperate because their constraints are standardized.

2. LEGO System of Play

LEGO bricks interoperate across decades because stud geometry and tolerances are standardized. Children configure complex structures without specialized training. The interface limits the space of valid combinations.

3. Automotive Vehicle Platforms

OEMs design vehicle platforms with defined hardpoints, connection geometries, and parametric envelopes. Body variants, engines, interiors, and modules all assemble onto pre-defined interface rules.

4. PC Ecosystem (ATX / PCIe / SATA)

Standardized interfaces allow motherboards, power supplies, GPUs, and peripherals from different manufacturers to interoperate. The hardware market thrives because the “rules of assembly” are public.

5. Consumer Kitchen Systems (IKEA Methods)

A universal mounting rail defines cabinet placement, alignment, and load paths. Users select products from a catalog; the rail ensures composability.

6. Aerospace Digital Twins

Aerospace uses parametric kernel formats enhanced with rules (MATLAB/Simulink, Modelica) to capture allowable variations for complex assemblies—an early ancestor of rule-based configurability.

Implication for AEC:
Construction is the only major fabrication industry lacking a rule-native product file format. The Configurator File Type aligns AEC with the standardization approach that transformed every other product ecosystem.

Appendix C. User Stories (Initial Working Set)

These draft user stories (with placeholders where Steve's future refinements will slot in) illustrate how different stakeholders interact with the file type.

1. Manufacturer User Story

The engineering team at PanelCo exports their wall panel geometry from Inventor. They attach a Configurator File Type definition that describes allowable lengths (8'–16'), stud spacing, interface conditions, fire rating, finish variants, and mateline logic. The files are hosted on PanelCo's server. Configurators always fetch the current version.

2. GC / Developer User Story

A developer uses a multi-vendor configurator to explore dozens of building variants. The configurator snaps together pods, panels, and façade elements from several manufacturers. All combinations are validated by the rules in each product definition. The developer exports pricing, scheduling, and structural data for the top three options.

3. Code Official User Story

A code official reviews a configuration package containing provenance data: every product version used, every constraint invoked, and any rule exceptions. Instead of reviewing drawings, the official verifies that rules encoded in the file type meet local code requirements.

4. AI Agent User Story

A generative agent receives a building envelope and a set of goals (maximize 1BR units; minimize plumbing risers; reduce embodied carbon). It searches thousands of possible configurations of compliant products because the file type defines the allowable combinations. Every proposed solution is valid by construction.

5. Third-Party Configurator Startup User Story

A startup builds a lightweight configurator that can read the file type. Without holding any manufacturer data, the tool can configure multi-vendor assemblies as long as the manufacturers publish compliant rule files.

These stories will expand as Steve and Sam develop more detailed workflow diagrams.

Appendix D. Examples of Constraint Types Encoded by the File Type

Geometric Constraints

- Dimensional limits
- Alignment rules
- Tolerances

Performance Constraints

- Fire ratings
- Sound transmission ratings
- ADA clearances

Adjacency Constraints

- Product A must be adjacent to Product B
- Product C cannot be adjacent to Product D

Interface Constraints

- Required connection type
- Required utilities
- Maximum load transfer

Code Constraints

- Egress requirements
- Fixture counts
- Fire separation rules

Manufacturing Constraints

- Maximum panel length for shipping
- Minimum radii for bending
- Factory platform dependencies

Each constraint becomes a machine-readable rule.

Appendix E. Analogy Matrix for Stakeholder Communications

(To help the CfOC explain the file type to different audiences.)

Audience	Analogy	Explanation
Manufacturers	**PC motherboard standards**	Your product becomes a GPU or RAM module: self-describing, interoperable, discoverable.
Developers	**IKEA + Lego**	Building from catalogs, not drawings; configurations are safe by design.
Funders	**USB standard**	A neutral standard unlocks innovation across thousands of vendors.
Regulators	**UL listing + code mapping**	Products carry their certifications in a structured way that is easy to verify.
AI firms	**Modelica / Simulink for buildings**	A rule-native kernel that makes valid configuration spaces accessible to agents.

See more in Chapters 1–6.