2026 European Conference on Computing in Construction
Corfu, Greece
July 12-15, 2026

PROPOSAL FOR A MODERN FOUNDATION FOR A DATA-DRIVEN BUILT ENVIRONMENT

Léon van Berlo[1], Thomas Krijnen[2], Tom van Diggelen[3], Artur Tomczak[4], Evandro Alfieri[4], David de Koning[5], Greg Schleusner[6], Helga Tauscher[7] and Dennis Shelden[8]
[1]Boundless BIM Council, Breda, Netherlands; [2]AECGeeks, Eindhoven, Netherlands; [3]KUBUS, Eindhoven, Netherlands; [4]buildingSMART International, London, UK; [5]Oasys (Arup Canada), Montreal, Canada; [6]HOK, New York, USA; [7]HTW Berlin, Berlin, Germany; [8]RPI, Troy, US

Abstract

The Industry Foundation Classes (IFC) standard underpins BIM interoperability, but its STEP/EXPRESS stack is ill-suited to cloud-native, transactional workflows. Building on buildingSMART community-driven requirements and proposals, review of alternative solutions, and iterative prototyping, we propose the modernisation of IFC. The work includes a proposal for a data schema, JSON serialisation, and exchange mechanisms, while preserving IFC's role as a semantically rich neutral data model. Innovation includes a compositional, layered federation model allowing incremental updates, multi-author collaboration, versioning, and fine-grained, traceable ownership. We compare the proposal to existing standard and discuss the potential, trade-offs and implications.

Introduction

IFC originated in the mid-1990s as a vendor-neutral product model for exchanging building information, initiated by the International Alliance for Interoperability (IAI) and later stewarded by buildingSMART as an open standard for BIM (Building Information Modelling). Through iterative releases in the 1990s–2000s, IFC gravitated towards its current scope. The common IFC use cases span design and construction coordination, design data transfer, asset management, building permitting and documentation archiving. The IFC in its current form has matured into a widely adopted format, implemented in over 400 software tools (buildingSMART Int., 2026b), enabling structured exchange of building and infrastructure data worldwide.

The IFC specification has been extended with each subsequent release, with addition of more entities and properties to describe the broad landscape of architecture, engineering, construction and operations (AECO) concepts. For example, the latest release – the IFC 4.3.2 – introduced classes and properties to more accurately describe roads and rails (buildingSMART Int., 2024). Some updates unified properties, that before could differ in definition depending on the object described.

Aiming to support plurality of industry use cases, the IFC standard grew in complexity, allowing many alternative ways for modelling some aspects. For example, alphanumeric data can be attached to objects as attributes or properties; objects are classified using IFC entities, but also predefined types and classification references; there are forty ways to assign material to an object (Tomczak et al., 2024) and over 170 kinds of geometric primitives - subtypes of *IfcRepresentationItem* and *IfcProfileDef* (buildingSMART, 2024). Noardo et al. (2019) investigated geometric errors and implementation inconsistency. This kind of implementation inconsistency is incredibly detrimental to the necessary trust in a digital language of the construction sector. The combinatorial nature of the geometry definitions results in implementations supporting different subsets of the standard which further inhibits successful interoperability. Lastly, the complexity associated with the immense breadth of geometric paradigms can be seen as a barrier to adoption.

On a specification level, the IFC follows strict compatibility dogmas, so reductions in scope are rarely realised, which results in a gap between specification content and actual usage in practice. While cloud-based data exchange and incremental updates are not new in contemporary BIM tools, the IFC today is primarily used for file-based exchange, where a full dataset needs to be transferred, even for a simple update of one property value. Furthermore, the IFC preserves information about a single author and authoring application per file, limiting traceability.

Related works

An influential attempt at modernizing IFC has been captured by van Berlo et al. (2021). Core ideas there were a re-evaluation of the amount and lack of consistency between relationship types, and introduction of the concept of 'late binding' to IFC. Over time, the community has come to realise that the taxonomy of building elements does not need to map to distinct classes in computer programming languages, and are therefore better kept out of the main schema in a secondary "late bound" layer, more akin to the data format in which property sets are described in the specification. This same research also indicates the desire to move towards 'object-based incremental updates' but does not yet give concrete technical steps towards realizing this, but does list predictable, normalised tree structures in the graph as an enabler. The Technical Roadmap (buildingSMART, 2020) sets the direction and calls to reconsider the technological stack of openBIM standards.

The STEP Physical File (SPF) remains the primary and most widely used serialisation format for IFC, although it is not the only officially supported option. Alternative serialisations include ifcXML and the compressed ifcZIP

formats (buildingSMART, 2022). JSON-based representations, aligned with contemporary web data exchange practices, have been proposed as alternatives to the traditional IFC stack by Afsari et al. (2017) and Brouwer et al. (2020). Despite these efforts, SPF has retained dominance due to its tight coupling with EXPRESS schemas. It enforces strict schema rigidity and ordered, unnamed attribute sequences, allowing SPF to represent complex graph structures concisely. This effective SPF structure has no real equivalent in modern languages and thus limits the industrial adoption of alternative serialisation formats.

Several efforts have explored the application of semantic web technologies for BIM. As part of the IfcOWL project, the IFC has been mapped to OWL, enabling RDF-based representations that can be queried using SPARQL and linked with other web resources (Pauwels et al. 2017). IfcOWL also introduced alternative geometric encodings based on Well-Known Text (WKT) to reduce verbosity and support GeoSPARQL. However, the direct mapping of EXPRESS to OWL introduces many indirections, which negatively affect query performance and reasoning expressiveness. In response, lighter-weight ontologies such as BOT have been proposed to support higher-level building topology without replicating the full IFC schema (Bonduel et al., 2018).

Indirections are the intermediary constructs and steps that must be traversed in order to relate a value to an entity. IFC is known for its many indirections, for example to get from a material to a colour: "*IfcMaterial←IfcMaterialDefinitionRepresentation→IfcStyledRepresentation→IfcStyledItem→IfcStyleAssignmentSelect→IfcPresentationStyleAssignment→IfcPresentationStyleSelect→IfcSurfaceStyle→IfcSurfaceStyleElementSelect→IfcSurfaceStyleShading→IfcColourRgb".* These indirections come at a cost in terms of: (a) performance; data model size; query joins (b) cognitive overhead in understanding (c) lack of ownership clarity when reusing data. In RDF additional indirection occurs by representing ordered lists as binary trees of <*first, rest*>-pairs.

Property graph approaches have been explored in systems such as IFCWebServer (Ismail et al., 2017) and in multiple Neo4j-based mappings (Zhu et al., 2023). These approaches leverage graph traversal and analytics capabilities but typically require bespoke schema mappings and tooling.

Relational database storage of IFC data has also been extensively studied (Li et al., 2016; Solihin et al., 2017), with roots tracing back to early systems such as BLIS and Sable within the IMSrv platform (Beetz et al., 2010). However, mapping EXPRESS constructs - including deep inheritance hierarchies, complex aggregations, inverse attributes, and SELECT types - to relational schemas introduces significant complexity. This results either in large numbers of tables or substantial application-layer logic, a challenge commonly referred to as the object-relational impedance mismatch.

Key-value stores offer a more flexible alternative, as they impose minimal structural constraints and delegate serialisation entirely to the application. BIMserver.org is a prominent example of this approach (Beetz et al., 2010). While efficient and well-suited to complex object graphs, key-value storage is typically binary, non-portable, and not human-readable, limiting its suitability for archival or interoperable exchange scenarios.

Binary formats have also been explored for handling large explicit datasets, particularly meshes and point clouds, using HDF5 (Krijnen & Beetz, 2020). Although technically effective for specific high-volume data scenarios, these approaches address a narrow set of use cases and introduce dependencies on additional complex standards, despite the existence of an official EXPRESS-to-HDF5 mapping in ISO 10303-26. They also do not align well with requirements for incremental exchange or web-based workflows.

More recently, Universal Scene Description (USD), originally developed by Pixar, has gained attention due to its strong support for large-scale collaborative asset composition. Governed by the Alliance for OpenUSD and undergoing standardisation efforts, USD emphasises explicit data, layered composition, and conflict resolution through ordered 'opinions'. Its schema-agnostic composition mechanisms resemble certain IFC concepts, such as TypeObject–Occurrence relationships but operate at the serialisation level rather than within a formal schema. These characteristics make USD conceptually attractive for AEC collaboration, despite its origins outside the domain.

Finally, lightweight mesh-based formats such as glTF support many contemporary visualisation-centric use cases, including coordination and clash detection, while enabling integration with a broad ecosystem of off-the-shelf tools. However, they largely omit semantic richness.

Borkowski (2026) argues that mesh-based formats such as glTF and USD, combined with structured non-geometric data stored for example in JSON, CSV, or SQL, represent a plausible alternative to kernel-dependent IFC. He characterises traditional file-based BIM workflows as slower and more cumbersome to integrate than granular data systems, and suggests that BIM practice may need to shift from a software-centric toward more data-centric.

The breadth of explored serialisation and storage approaches highlights fundamental limitations of the EXPRESS-based ecosystem in supporting contemporary construction industry use cases. The expressive power of EXPRESS, while beneficial for formal schema definition, complicates mapping to modern, web-oriented data frameworks. The requirements driving these investigations are discussed in the following section.

Method

Requirements

The requirements documented here were shaped over years of involvement in the buildingSMART community, and have been refined through in-person meetings and presentations.

1. Distributed ownership and authorship traceability

The construction sector is a multi-disciplinary field with strict responsibility boundaries and liability. It should be possible to combine datasets while strictly retaining the provenance of where information comes from, without inhibiting establishing rich semantic links between concepts from distinct disciplines. To realise accountability, the provenance of information needs to be traceable. IFC to date has an *IfcOwnerHistory* entity, but that tracks only the meta-data of the latest rendition of the object. There is no way to retain deleted information, or a longer provenance trail.

2. Incremental updates

Building models can realistically amount up to gigabytes of data in contemporary engineering projects. It should be possible to provide only the changes to earlier exchanged model.

3. Moderate complexity

Many of the contemporary use cases in which IFC is employed are not that complex, such as visualisation and coordination. There is also an asymmetry in complexity: IFC is often seen as a compromise between many different options of representing elements. Therefore, for an authoring tool it is sufficient to only support one single option and write that out. Conversely, a downstream application, such as a viewer or manufacturing platform must harmonize all different options coming from different tools. This asymmetry needs to be reduced to open BIM data to a wide variety of neighbouring domains and facilitate reuse of information downstream. This could for example mean moving the responsibility of interpreting complex procedural geometry to the producing side, where it is already rendered to a triangle mesh for display anyway.

4. Plurality of representation

Each physical and spatial object can be represented with geometry, but the accuracy, complexity and kind of geometry might differ, depending on the needs. A building at an urban scale might be simplified to a bounding box cube, while at close-up could be highly detailed. An example wall can be modelled as an extruded set of layers with procedurally cut-through openings, or as a triangulated mesh resulted from computation, to speed-up rendering speed. Same wall could be simplified to a guide curve and thickness, or have a point-cloud representation to compare as-built model with reality.

5. Performant

In order to increase the number of iterative design and engineering cycles the import/export/analysis times need to be reduced compared to contemporary IFC workflows. This also means less indirection (intermediary steps) between data, that are problematic in current querying workflows and may obscure ownership semantics.

6. Cloud-friendly, web-native

Common Data Environments (CDE) have become more popular, but true distributed and collaborative workflows are impeded by the monolithic and idiosyncratic nature of SPF-based exchanges.

7. Granular data access

Ability to expose or load only a subset of information. This could be either for performance or confidentiality reasons. For example, an engineer might want to see architect's walls, but not their finishing.

8. Extensibility

To support rapid innovation and multi-disciplinary challenges, extensibility should be possible on a level beyond adding additional key-value pairs as user-defined property sets. Examples include efficient encoding of compressed point clouds (Krijnen & Beetz 2017) or the impediments in realizing modelling circularity related element connectivity (Tomczak et al. 2024). Currently, in EXPRESS-based schemas, there is no realistic way to provide this kind of extensibility as it would require publishing a novel schema version with a unique schema identifier which affects even the software that would ignore the extension because it is outside of their intended domain.

9. More predictable and atomic data patterns.

Currently unclear whether something is a primitive with an identity or an attribute with a value. For example, *IfcMaterial* does not have a Globally Unique Identifier (GUID), and is therefore only a transitive object. Relationships have GUIDs, but tools cannot typically maintain consistency between export cycles.

10. Bridge BIM with GIS, IoT, Digital Twins etc.

Building information is not limited to design data, but fits into larger geospatial context commonly represented with GIS, and can serve during asset operation phase. In case of continuously updated models, often fed with live sensor data, BIM can serve a role of a Digital Twin. Therefore the data exchange standard of the future should not limit to one, but rather allow multiple schemas and representations.

11. Independent and easy to update modules

Up until now, the IFC standard has functioned as a monolithic data model published as one large standard document with almost 500 pages (ISO 16739-1:2024). While this ensured a comprehensive coverage of the built environment, it also resulted in a massive specification that is difficult to maintain and cumbersome for software vendors to implement fully. As the scope of IFC expanded to include new domains such as Rail, Road, and Tunnel, the time required between release cycles increased, often taking years to ratify updates.

Proposed Solution

We propose a novel data exchange solution, that is aimed to resolve common IFC issues, and address the requirements (1-11). The first building-block module of the proposed modern IFC is the schema, enabling further definition of an API module, domain knowledge/semantics modules and technical extensions.

A dataset called the 'Hello Wall' will be used to demonstrate functionalities of the proposed modern IFC, and is available together with other examples, in the *IFC 5 Development* GitHub repository (buildingSMART, 2026a). The dataset is presented in figures 3 and 4.

Layers and independent data storage

We advocate a fundamental dissociation between the business logic data model and its physical storage. A dataset should not be required to be a single monolithic file; instead, it operates as a compositional, layered federation of data. The *layers* mechanism in our proposal addresses requirements for distributed ownership (1) and incremental updates (2) by dissociating the data from physical storage, allowing a single dataset (or 'stream') to be composed of one or multiple federated chunks resolved at runtime.

This architecture enables non-destructive collaboration, where stakeholders can overlay their own data attributes onto a base model without altering the original data, thereby strictly preserving authorship and liability boundaries. Furthermore, this same technical framework handles incremental updates by treating changes as lightweight *delta* streams containing only the proposed modifications, eliminating the need to re-transmit massive monolithic files when only specific data points have evolved.

Composition and paths

For software to make meaning out of disconnected layers or streams, the composition needs to take place. The composition relies on consistent path identifiers like GUIDs, e.g. *'e3035b71-bd9f-4cdc-86fd-b56e2f4605b6'*, or a tree path, e.g. *'project/site/ building/wall_id'*. Because every piece of data is tied to a specific path, information about a single object does not need to reside in the same physical location.

For example, the base geometry of a wall might exist in a *geometry.ifcx* file, while its material properties reside in *materials.ifcx*, and a fire engineer's specific requirements in a yet separate *fire_safety.ifcx* file. When a software tool loads the data by reading these files, it recognises that they all refer to the same object path. It then overlays or *composes* these attributes to form the complete object.

Attributes

The most trivial data to attach to a path is an attribute, which links some set of data to a specific path. Attributes have a name and value and follow a pre-specified schema. Attributes in the proposed IFC are JSON key-value pair objects. For every key a schema needs to be defined either locally in the file or imported. Attribute keys are also namespaced to provide context. Numeric attribute values derive their unit from the quantity kind associated with the schemas. There is no distinction alike IFC4's Attributes and Properties, both are exchanged and validated in the same fashion. Similar situation is presented in figure 1.

```
(…)"data":[
{"path": "c8b…c48",
"attributes": {"name": "Something"}},
{"path": "c8b…c48",
"attributes": {"my::prop": 3}},
```

```
c8b…c48:
- name = "Something"
- my::prop = 3
```

Figure 1: Raw data nodes (top) and resultant composed object with attributes (bottom)

Incremental updates and distributed ownership

The proposed architecture directly supports modern workflows that were difficult to manage with previous IFC versions. Instead of re-sending a massive file when a single property changes, a user can exchange a small *delta* set containing only the changed attributes for specific object paths.

Different disciplines can maintain their own data – their *opinions* – without altering the original author's data. As demonstrated on figure 2, one user can add a '*FireRating: R60*' property in their own layer. If this conflicts with an existing *'FireRating'* of value *'R30'*, the composition engine warns user about the conflict. All while ensuring that ownership and provenance of data are preserved at fine-grained level, allowing to see exactly who authored each information and when.

```
(…)"data":[
{"path":"c8b…c48",
   "attributes":{"FireRating": "R30"}},
{"path":"c8b…c48",
   "attributes":{"FireRating": "R60"}},
```

```
c8b…c48:
- FireRating = ~~"R30"~~ "R60"
```

Figure 2: Demonstration of conflict during composition – the nodes (top) and resultant compiled object (bottom)

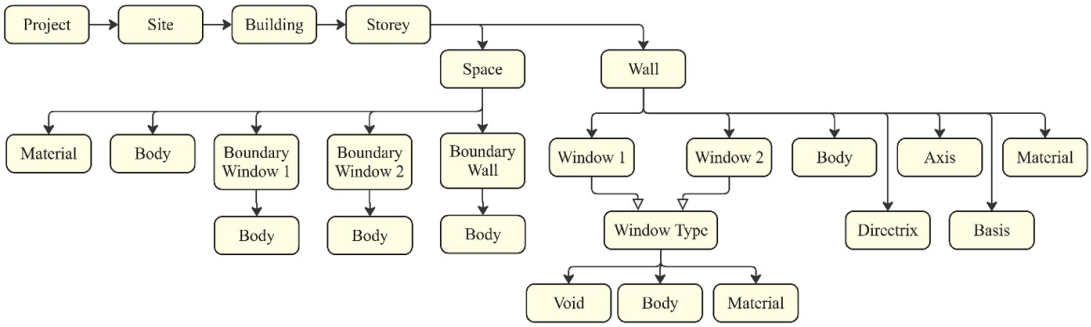

Figure 3: Diagram of the entities in the 'Hello Wall' file

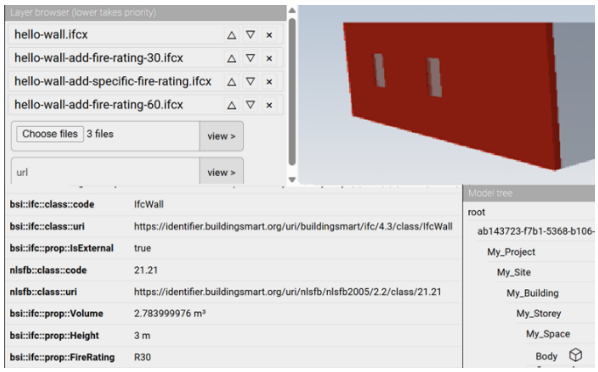

Figure 4: Layered 'Hello Wall' model composed and displayed in a demo IFCX viewer.

Children

Spatial decomposition relationships and local placement known from IFC 4.3.2 are represented in the proposal with *children*. Merging into one concept improves consistency. This, however, only applies to hierarchical IFC relationships like *Decomposes, Nests, AdheresTo, Voids, FillsVoids*. System-like graph relationships will be realised by means of attributes with path references.

Inheritance

We propose a generic, and multi-level information reuse mechanism to avoid complexity of mapped items and textual documentation on how to implement relationship between them. An example application is presented in figure 5, where two windows inherit from the same type defining frame and glazing.

```
(…)"data":[
{"path":"c8b…c48","children":
   {"window1":"e48…834",
   "window2":"caa…15d"}},
{"path":"e48…834","inherits":
   {"windowType":"189…5c3"}},
{"path":"caa…15d","inherits":
   {"windowType":"189…5c3"}},
{"path":"189…5c3","children":
   {"frame":"651…f2b",
   "glazing":"6ed…953"}}]
```

```
c8b…c48
   c8b…c48/window1
      c8b…c48/window1/frame
      c8b…c48/window1/glazing
   c8b…c48/window2
      c8b…c48/window2/frame
      c8b…c48/window2/glazing
```

Figure 5: Demonstration of the children and inheritance – the nodes (top) and compiled hierarchy tree (bottom)

Note that this is not the multiple inheritance that has gotten a negative connotation in certain programming languages with things like the 'diamond problems' but rather a formalised information reusage mechanism.

Modularisation of the schema

To address Requirement 11, our solution introduces a modular setup that streamlines ISO standardisation by decoupling a stable technical core from more dynamic, domain-specific semantics and extensions. Organised into distinct ISO Parts, this strategy replaces the previous monolithic structure to allow for independent versioning and faster industry adoption. This enables software to implement a reliable core quickly while allowing domain experts to expand semantic coverage without disrupting the underlying ecosystem.

Serialisation

The presented solution is serialised entirely in JSON, making it human-readable and compatible with modern web technologies. However, other serialisation forms are not off the table, JSON is just a readable and universally understood logical choice at this phase of the development and facilitates easy dissemination of the ideas around composition and file format mechanisms without being obfuscated by complex serialisation mechanisms. Binary format was discussed and can become a potential solution.

Schema definition

Our solution is defined using TypeSpec, a highly productive API description language that brings a concise, developer-friendly syntax (Microsoft, 2026). By leveraging TypeSpec, the schema benefits from improved readability and modularity, allowing for the automated generation of multi-protocol representations such as JSON Schema, OpenAPI, and GraphQL from a single source of truth. By utilising TypeSpec as the authoritative source for maintaining the standard, the architecture effectively decouples internal schema management from external distribution (JSON Schema, OpenAPI, etc.). This ensures rigor-

ous type safety across disparate data exchanges, ultimately fostering a more agile and interoperable ecosystem for the built environment.

Import flexibility

Our proposed architecture introduces a flexible import mechanism that allows the core schema to be extended with external vocabularies via URIs, preventing the standard from becoming a monolithic 'theory of everything'. This capability enables the composition of data models that mix and match distinct domain semantics while keeping the core lightweight.

In the 'Hello Wall' example, this is demonstrated through the direct import of OpenUSD for geometry, glTF PBR (Physically Based Rendering) for material definitions, and NlSfb and CCI for classification systems, and Open Geospatial Consortium (OGC) standards for geospatial data. By namespacing these attributes, we allow these diverse standards to coexist within a single dataset, facilitating richer interoperability and allowing the industry to leverage existing specialised standards rather than redefining them within IFC.

Similarly, the IFC does not need to reinvent the identifiers like *Global Trade Item Number* (GTIN) or *Global Location Number* (GLN), instead, those can be added to datasets with metadata acknowledging they are originating from GS1 organisation.

Geometry and Representation

Our proposal prioritises the use of explicit geometry to ensure consistent and efficient visualisation across software platforms. While the framework supports computational or procedural geometry definitions as semantic overlays or extensions, a objects should always be accompanied by an explicit geometric representation. This dual approach maximises interoperability by ensuring fast rendering and precise analysis without requiring consuming applications to interpret complex procedural logic. Currently, the alpha examples utilize the USD geometry schema (*usd::usdgeom*) as the rendering core to leverage its robust capabilities in visual composition and incremental updates. However, the architecture remains technology-agnostic regarding specific mesh formats; other explicit geometry definitions, such as those from the OGC and glTF, have also been evaluated to ensure broader compatibility with digital twin and GIS ecosystems.

Discussion and comparative analysis

Schema difference

Differences are visible when comparing equivalent IFC content captured in traditional STEP-based and proposed JSON-based form – see figure 6. The STEP sample has many more first-class objects like Direction, Point or Axis. While the STEP sample lists data in specific and defined entities, like the IfcSpace, the JSON sample defines data through components, inspired by the Entity Component System (ECS) architecture.

```
#1=IFCCARTESIANPOINT((0.,0.,0.));
#2=IFCDIRECTION((0.,0.,1.));
#3=IFCDIRECTION((1.,0.,0.));
#4=IFCAXIS2PLACEMENT3D(#1,#2,#3);
#5=IFCLOCALPLACEMENT($,#4);
#6=IFCCARTESIANPOINT((0.,0.,0.));
#7=IFCDIRECTION((0.,0.,1.));
#8=IFCDIRECTION((1.,0.,0.));
#9=IFCAXIS2PLACEMENT3D(#6,#7,#8);
#10=IFCLOCALPLACEMENT(#5,#9);
#11=IFCCARTESIANPOINT((0.,0.,0.));
#12=IFCDIRECTION((0.,0.,1.));
#13=IFCDIRECTION((1.,0.,0.));
#14=IFCAXIS2PLACEMENT3D(#11,#12,#13);
#15=IFCLOCALPLACEMENT(#10,#14);
#16=IFCCARTESIANPOINT((0.,0.123,0.));
#17=IFCDIRECTION((0.,0.,1.));
#18=IFCDIRECTION((1.,0.,0.));
#19=IFCAXIS2PLACEMENT3D(#16,#17,#18);
#20=IFCLOCALPLACEMENT(#15,#19);
#21=IFCSPACE('x…1',$,$,$,$,#20,#22,$,$,$,$);
```

```
(…)"data":[
{"path": "x...1",
  "attributes": {
    "usd::xformop": {
      "transform": [
        [1,0,0,0],[0,1,0,0],
        [0,0,1,0],[0,0.123,0,1]]}}},
{"path": "x...1",
  "attributes": {
       "bsi::ifc::class": {
         "code": "IfcSpace",
         "uri": "https://.../IfcSpace"}}}]
```

Figure 6: Comparison of a space definition in STEP-based IFC 4.3.2 (top) and proposed JSON syntax (bottom).

Inheritance depth and reification

The traditional IFC is full of reified objects, like a relation or a set of properties. For example, there are five intermediate entities between object and its property value: *"Object←RelDefinesByType→TypeObject→PropertySet→Property→IfcValue (select)→Simple value"*. In contrast, the proposal simplifies the assignment by allowing values to be added directly to objects: *"Object: {Property: Value}"*.

The IFC of today also employs rather deep inheritance of the entities. To read the attributes of an IfcWall shown in figure 7, one needs to traverse six levels up and six down the concept hierarchy. In proposed IFC, the inheritance depth can be three to four times shorter.

```
IfcRoot→…IfcObject→IfcProduct→IfcElement→…
IfcWall
→IfcRelDefinesByType
  →IfcWallType
    →IfcRelAssociatesMaterial
      →IfcMaterialLayerSetUsage
        →IfcMaterialLayerSet
          →IfcMaterialLayer: "mineral wool"
          →IfcMaterialLayer: "concrete"
```

```
Wall
(→WallType)
  →Insulation
```

```
    →Material: "mineral wool"
  →Core
    →Material: "concrete"
```

Figure 7: Comparison of inheritance depth in defining materials of a wall; IFC 4.3.2 (top) our solution (bottom).

On the other hand, the IFC 4.3.2 allows elements to only be typed by one element (type), while proposed IFC allows multiple typical levels to reuse data from. For example, a certain manufacturer's product could inherit from a typical kind of a product, that could also inherit from a more generic type.

Even though using IFC as product catalogues were possible with previous versions, our proposal introduces data ownership that allows to track origins of each property value, ensuring higher trust in the data. For example, a sustainability analyst could spot if Environmental Product Declaration (EPD) values were altered by architect before performing the Life Cycle Assessment (LCA).

Storage size

The mere storage is not a priority these days with fast internet and cheap storage. Still, the smaller files can be exchanged faster, so we performed a comparison of the simple building model represented as both STEP-based IFC and the proposed modernised IFC.

At first glance, the proposed IFC JSON syntax and serialisation does not bring advantages to storage size – the 'Simple scene' model recreated in .ifcx is over a quarter larger (see table 1). However, the JSON format is commonly condensed (minified), which gives our proposed solution a significant advantage over STEP representation – almost 60% reduced size. Additionally, it also compresses twice as efficiently as STEP, which is relevant for archiving.

Table 1: File size comparison on the sample dataset.

Version	Default (prettified)	Minified	Compressed (zipped)
IFC 4.3.2	7.91 Mb	NA (same)	1.04 Mb
Proposed IFC (.ifcx)	10.10 Mb	3.39 Mb	0.52 Mb
Difference	+27.7 %	-57.1 %	- 50 %

The big difference can be observed when we look at a typical exchange process where files are being sent at regular intervals, despite insignificant changes being made. Table 3 shows that whereas in STEP-based IFC each transfer involves export of the full dataset, in the proposed solution only the deltas containing a change are being exchanged. This means significant storage saving throughout project duration.

Next steps

The comprehensive restructuring of the IFC data model into a modular, predictable ecosystem of parts serves as the necessary foundation for the true long-term objective: the creation of a standardised API. As articulated in the development roadmap, the finalisation of the Core data schema is the critical prerequisite that allows the industry to move beyond static file exchanges toward dynamic, cloud-native interactions where online agents or automated services, such as energy analysis tools or (manufacturing, simulation, etc.) bots, can interact directly with data. The current technical shifts toward JSON serialisation and ECS-like components are essentially the predecessor work required to standardize the openAPI interface itself – covering authentication, endpoints, and transactional calls. Binary data streaming is also not ruled out as part of the final solution. This will ultimately enable the real-time, granular data access that modern digital twins and connected ecosystems require.

Table 2: File size comparison on the sample dataset with two iterations of property value changes.

Version	Default (Mb)	1st change (Mb)	2nd change (Mb)	Total growth (%)
IFC 4.3.2	7.91	7.91	7.91	+200
Proposed IFC (.ifcx)	10.10	0.0004	0.0004	+0.008

Conclusions

The proposed modernisation of IFC represents a necessary shift for the construction industry, moving away from the monolithic legacy of previous standards toward a modular, high-performance data architecture. By adopting ECS-like components serialised in JSON, our proposal addresses the long-standing industry demand for files that are not only human-readable and web-friendly but also capable of supporting modern workflows such as incremental updates, version control, and multi-user collaboration. This technological leap is designed to drastically lower the barrier to entry for software vendors, prioritizing implementation speed and predictability to foster a more robust and reliable openBIM ecosystem.

The introduction of a modularisation strategy is central to this vision. By decoupling the stable technical Core from dynamic Domain Semantics and Industry Practices, the standard solves the historical challenge of painfully slow-release cycles. This separation of concerns allows the Core to serve as a reliable, domain-agnostic kernel for identifying, grouping, and positioning objects, while allowing specific disciplines, such as Rail, Tunnel, or Structural, to evolve their semantic definitions independently and rapidly.

While we introduce a breaking change regarding file formats, it ensures semantic continuity by leveraging existing IFC 4.x definitions and integrating with broader industry standards like USD, 3D Tiles, glTF, GS1, OGC or classification systems like ETIM, ECLASS, CCI or UNICLASS. Ultimately, the goal of IFC modernisation is to achieve fully reliable data usage. By reducing the complexity of the core schema and eliminating ambiguous exceptions, our work positions itself as a predictable and reliable foundation for the next generation of automation, digital twins, and connected built environment data.

Acknowledgments

The paper represents the result of the incubation phase of IFC 5 project at buildingSMART during 2021 to 2025. The final version will be further shaped by the buildingSMART and ISO processes. Authors would like to thank all the people involved in the initiative over the years, the members and community of buildingSMART, participants in recurring Implementers Assemblies and independent contributors.

References

Afsari, K., Eastman, C.M. & Castro-Lacouture, D. (2017) JavaScript Object Notation (JSON) data serialization for IFC schema in web-based BIM data exchange. *Automation in Construction*, 77, p.pp.24–51. https://doi.org/10.1016/j.autcon.2017.01.011

Beetz, J., de Laat, R., van Berlo, L. A. H. M., & van den Helm, P. (2010). Towards an open building information model server: report on the progress of an open IFC framework. In *10th International Conference on Design and Decision Support Systems in Architecture and Urban Planning, DDSS 2010*. Eindhoven University of Technology.

van Berlo, L., Krijnen, T., Tauscher, H., Liebich, T., van Kranenburg, A. & Paasiala, P. (2021) Future of the Industry Foundation Classes: towards IFC 5. In: *Proceedings of the 38th International Conference of CIB W78. Luxemburg*, 11 October 2021. p.pp.123–137.

Bonduel, M., Oraskari, J., Pauwels, P., Vergauwen, M., & Klein, R. (2018) The IFC to Linked Building Data Converter-Current Status. *Proceedings of 6th Linked Data in Architecture and Construction Workshop*.

Borkowski, A. S. (2026) A critical analysis of the direction of BIM development: From CAD through file-based approaches and SaaS to granular data. *Journal of Information Technology in Construction*, 31, 380-393. https://doi.org/10.36680/j.itcon.2026.016

Brouwer, J., Pauwels, P., Sparks, D. (2020) ifcJSON Repository. Available at: https://github.com/buildingsmart-community/ifcJSON (Accessed: 23 Jan 2026).

buildingSMART International (2020) Technical Roadmap buildingSMART: Getting ready for the future. Version: 30 April 2020. Available at: https://www.buildingsmart.org/wp-content/uploads/2020/09/20200430_buildingSMART_Technical_Roadmap.pdf (Accessed: 23 January 2026).

buildingSMART International (2022) IFC Formats. Available at: https://technical.buildingsmart.org/standards/ifc/ifc-formats/ (Accessed: 23 Jan 2026).

buildingSMART International (2024) IFC 4.3.2 documentation. Available at: https://ifc43-docs.standards.buildingsmart.org/ (Accessed: 23 Jan 2026).

buildingSMART International (2025) Global openBIM Mandates. Available at: https://www.buildingsmart.org/wp-content/uploads/2025/03/IFC-Mandate_2025.pdf (Accessed: 23 January 2026).

buildingSMART International (2026a) IFC5 Development Repository. Available at: https://github.com/buildingSMART/IFC5-development/ (Accessed: 23 Jan 2026).

buildingSMART International (2026b) Software Implementations. Available at: https://technical.buildingsmart.org/resources/software-implementations/ (Accessed: 23 Jan 2026).

Ismail, A., Nahar, A., & Scherer, R. (2017) Application of graph databases and graph theory concepts for advanced analysing of BIM models based on IFC standard. In: Conference: 24th International Workshop on Intelligent Computing in Engineering (EG-ICE), Nottingham, UK.

ISO 16739-1:2024 (2024) Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries – Part 1: Data schema. Available at: https://www.iso.org/standard/84123.html

Krijnen, T., & Beetz, J. (2020). An efficient binary storage format for IFC building models using HDF5 hierarchical data format. *Automation in Construction*, *113*.

Li, H., Liu, H., Liu, Y., & Wang, Y. (2016). An object-relational IFC storage model based on oracle database. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *41*, 625-631.

Microsoft (2026). What is TypeSpec? Available at: https://learn.microsoft.com/en-us/azure/developer/typespec/overview (Accessed: 23 Jan 2026).

Noardo, F., Arroyo Ohori, K., Biljecki, F., Krijnen, T., Ellul, C., Harrie, L., & Stoter, J. (2019). GeoBIM benchmark 2019: design and initial results. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives* (No. 2/W13, pp. 1339-1346). ISPRS.

Pauwels, P., Krijnen, T., Terkaj, W., Beetz, J. (2017) Enhancing the ifcOWL ontology with an alternative representation for geometric data https://doi.org/10.1016/j.autcon.2017.03.001

Solihin, W., Eastman, C., Lee, Y. C., & Yang, D. H. (2017). A simplified relational database schema for transformation of BIM data into a query-efficient and spatially enabled database. *Automation in Construction*, *84*, 367-383.

Tomczak, A., Benghi, C., van Berlo, L., Hjelseth, E. (2024) Requiring Circularity Data in BIM With Information Delivery Specification. https://doi.org/10.55845/REJY5239

Zhu, J., Wu, P., Lei, X. (2023) IFC-graph for facilitating building information access and query, https://doi.org/10.1016/j.autcon.2023.104778