Semantic IFC Validation Using IDS in the Opensource BIMserver

Sebastian Schilling[1], Zaqi Fathis[1] and Christoph Frey[1]
[1]Hochschule für Technik und Wirtschaft Dresden - University of Applied Sciences, Dresden, Germany
E-mail(s): sebastian.schilling@htw-dresden.de, zaqi.fathis@htw-dresden.de, christoph.frey@htw-dresden.de

Abstract: Information Delivery Specification (IDS) provides a standardized mechanism for validating IFC models against explicitly defined information requirements. Compared to Model View Definitions (MVD), IDS is more lightweight and less complex, supporting semantic data quality and consistency in Building Information Modeling (BIM) workflows. However, practical implementations often lack an integrated pipeline connecting requirement authoring with server-side model validation.
This paper introduces a reusable, server-side workflow linking data dictionary–based requirement definition and semi-automated IFC model validation. Its core contribution is the systematic derivation of structured information requirements as IDS rules from domain-specific semantics and their execution within a decoupled validation architecture.
The workflow is implemented using open-source components. The property server datacat supports authoring and publishing data dictionaries, while the Opensource BIMserver is used to store and process IFC models. The architecture enforces a clear separation of responsibilities, with datacat generating IDS rules and the Opensource BIMserver executing IDS-based validation. The approach is evaluated on an IFC model to validate classification-based requirements, demonstrating consistent and machine-interpretable checks across system boundaries. The results indicate that the workflow supports reproducible and extensible validation processes in distributed BIM environments.

Keywords: Opensource BIMserver, Information Delivery Specification (IDS), Industry Foundation Classes (IFC), semantic validation, data dictionary



Published in the conference proceedings of the 37. Forum Bauinformatik 2026, Rostock, Germany, DOI: https://doi.org/10.18453/rosdok_id00005846
© 2026 The copyright for this article lies with the authors. Use is permitted under a Creative Commons license Attribution 4.0 International.

1 Introduction

The growing digitalization in the architecture, engineering, and construction (AEC) industry is enabling the development of automated inspection processes to optimize workflows in construction projects and minimize errors. Most BIM processes use Industry Foundation Classes (IFC) for the file-based exchange of building models. However, IFC models can also be made available online. One option for doing so is the Opensource Building Information Model server (BIMserver)[1].

[1]https://github.com/opensourceBIM/BIMserver

The Opensource BIMserver can be used to store and manage the information of a construction project using the IFC schema. However, it does not operate on a file-based system. Instead, the IFC data is stored as objects using the key-value database BerkeleyDB, with one record per IFC entity instance. Through queries, they can also be retrieved as submodels and can be serialized into different outputs [1]. Additionally, models can be checked, versioned, and merged to support IFC models throughout the construction lifecycle. Checking project-specific semantic requirements, however, is not yet possible. BuildingSMART International introduced the Information Delivery Specification (IDS) for such checks. Existing IDS workflows are predominantly local and file-based, lacking integration with server-side model management platforms such as the Opensource BIMserver.
[2] argue that rule-checking should support a lifecycle-oriented workflow in which requirements are structured, validated, reused, and handed over in a standardized and traceable manner, rather than being limited to single projects or isolated project phases. This motivates the use of automated validation mechanisms such as IDS in the Opensource BIMserver. IDS is machine-readable and allows for automatic compliance checking of IFC models, which increases quality control and data reliability [3].
To enable standardized validation rules in IDS, consistent semantics must be used in both the validation rules and the IFC models. In the construction industry, data dictionaries increasingly serve as external classification systems to uniformly describe classes and attributes in IFC models. In this process, concepts are made centrally and openly accessible via data dictionaries, which BIM authoring software can access through interfaces. Therefore, IDS files should also be defined based on the concepts provided in the data dictionaries.
To this end, we have developed an open-source property server called "datacat"[2]. It can not only provide data dictionaries, such as the buildingSMART Data Dictionary (bSDD), but can also be used to create, edit, and manage concepts. Beyond this, datacat is intended to generate and provide IDS files directly from these concepts. The Opensource BIMserver will retrieve them for checking IFC models through a semi-automated workflow.
The following research questions will be addressed:

- Can IDS rules be created directly in a property server based on standardized concepts?
- How can property servers and open-source BIM servers be linked so that the semantics in IFC models can be validated automatically?

2 Literature Review

Semantic validation or rule checking has already been the subject of some research. [4] used semantic web technologies to validate project requirements, and [5] used ontologies to check the quality of BIM models against standards. However, semantic web technologies are expressive, but remain a largely academic approach that is not yet widely used in practice. buildingSMART International therefore developed the Information Delivery Specification (IDS), a lightweight schema for verifying requirements.

[2]https://github.com/dd-bim/datacat-stack

This lightweight character sets IDS apart from its predecessor, the Model View Definition (MVD). An MVD defines a subset of the IFC schema, as well as the concepts to be exchanged. Its serialization, mvdXML, can also validate the 3D models. As it defines concepts as IFC instance graphs and resolves the IFC inheritance hierarchy, a rule defined on a supertype also applies to its derived entities [6]. However, this expressiveness was intended for software certification rather than requirement authoring and is one reason for the complexity of mvdXML. buildingSMART reports that vendors considered mvdXML as unimplementable, that the published XSD is invalid, and that its documentation is ambiguous. For these reasons, IDS is the recommended method to define information requirements for IFC datasets [7]. IDS trades expressiveness for implementability. It limits requirements to entities, properties, attributes, classifications, materials, and relations [6]. IDS does not apply automatic class inheritance, so applicable entities are listed explicitly [8]. Unlike MVD, IDS cannot check the geometric representation of an element. Because IDS is bound to the IFC schema, it returns the same results across checking software [3], which fits the reproducible and server-side validation in this work.
IDS has been investigated by several authors [9], [10], [11], [12]. [10] use the bSDD to create Exchange Information Requirements (EIR), but do not use external classification systems to create IDS. [9], [11], [12] use bSDD content to create IDS files with different IDS editors. They all use the BonsaiBIM add-on for Blender to check IFC models against IDS files. These approaches share a local, file-based checking process with a low degree of automation. The bSDD itself does not author IDS files. It serves standardized classes, properties, and materials that IDS editors reference by URI [11], [13].
A community script automates IDS creation from a dictionary by converting a bSDD object-type dictionary directly into an IDS file [14], yet it runs locally as a one-shot conversion, produces classification requirements only, and supports neither property-set validation nor class inheritance.

3 Methodology and Implementation

Our goal is to verify the use of an external classification system's semantics within an IFC model. IFC entities should be classified according to project-specific requirements and possess certain properties and values. In this context, datacat acts as an authoring service that formalizes (project-specific) class and property definitions as external classification systems that can be used to enrich the IFC model semantically with information during its creation. To verify that the enrichment was implemented according to the information requirements, we generate standard-compliant IDS files from them. To avoid typing errors, we extended the property server with a module that generates IDS based on the classification systems' concepts, ensuring they utilize the correct semantics directly.
An IDS specification expresses its applicability and requirements through six facets [15]. The *entity* facet identifies the IFC class of an element and, optionally, its predefined type. The *attribute* facet constrains the direct attributes of an IFC entity, such as `Name` or `Description`, by name and value. The *classification* facet checks the classification system and code assigned to an element, which may reference an external source such as datacat or the bSDD via a URI. The *property* facet checks properties and quantities, identified by their property set, name, and value. The *material* facet checks the names of associated materials, and the *partOf* facet checks structural dependencies between elements, for example membership in an assembly or group.

A model-checking plugin based on an IDS was developed and implemented in the Opensource BIMserver. BIMserver supports such extensions through a plugin mechanism that operates independently of its core IFC storage and query services. This mechanism has previously been used to extend the server for other purposes, such as data format conversion [16]. The plugin itself utilizes BonsaiBIM as a reference for the testing and report structures. Among the three automatic execution triggers supported by BIMserver, the plugin utilizes the new-revision trigger. The trigger is activated automatically whenever a new IFC model revision is uploaded, and the plugin subsequently runs a validation process against the IDS specification shown in Figure 1.

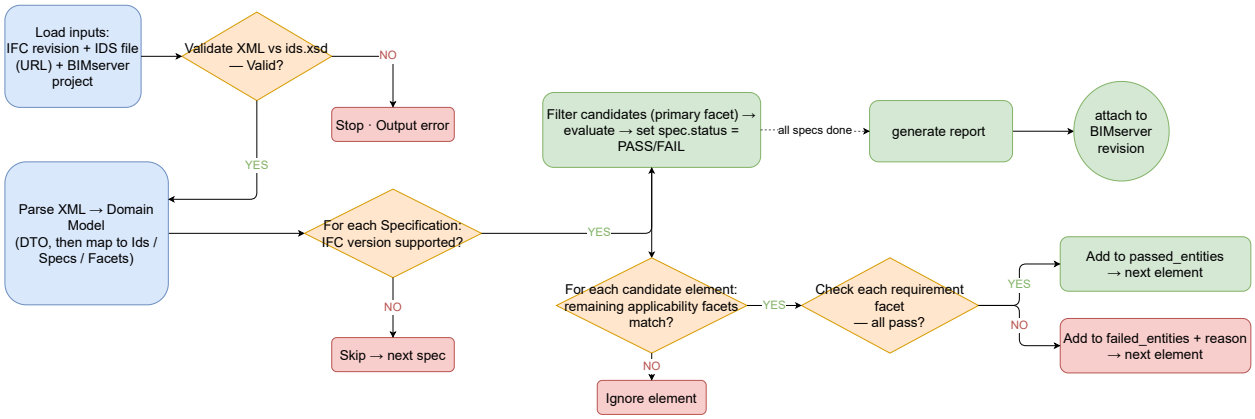


Figure 1: IDS checking plugin workflow in BIMserver

The plugin retrieves the referenced IDS file, validates it against the IDS XML schema, and parses it into an internal model of specifications and facets. For each specification whose IFC version is supported, it identifies applicable elements via the applicability facets, then checks each against the requirement facets to record a pass or fail with reason. The results are aggregated into a report that can be downloaded as a text file.

Because BIMserver's plugin setting only accepts text or a link rather than a file, we additionally implemented an architecture that uses object storage as a transfer and storage location for IDS files authored in datacat. Figure 2 shows the components and the data flow between them. This architecture enables semi-automated verification of IFC models in the BIMserver. The IDS module in the datacat property server has a new button that allows users to upload their IDS files to external, persistent storage. In the BIMserver, the link to the IDS file in the object storage is stored only once.

4 Validation and Discussion

Validation is performed on an IFC model of a garage. The model was classified and attributed using a Computer-Aided Facility Management (CAFM) catalog and a project-specific OpenFabTwin catalog. Figure 3 shows the garage door in BIMserver with its added properties from the CAFM catalog. For the windows and the interior space, an IDS file was created in datacat to verify the classification and attribution. All IfcWindows, for example, should be annotated with the "Fenster" class from the CAFM catalog and the associated property set "OFT_Pset_Fenster". In addition, the "Schalldämmung"

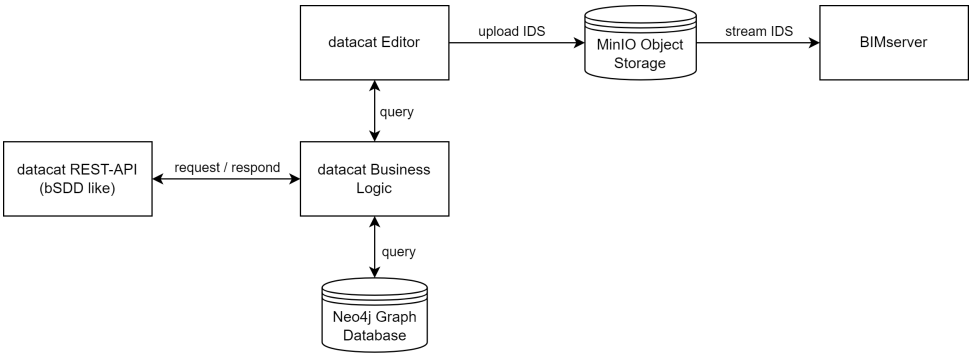


Figure 2: Data flow between datacat and BIMserver

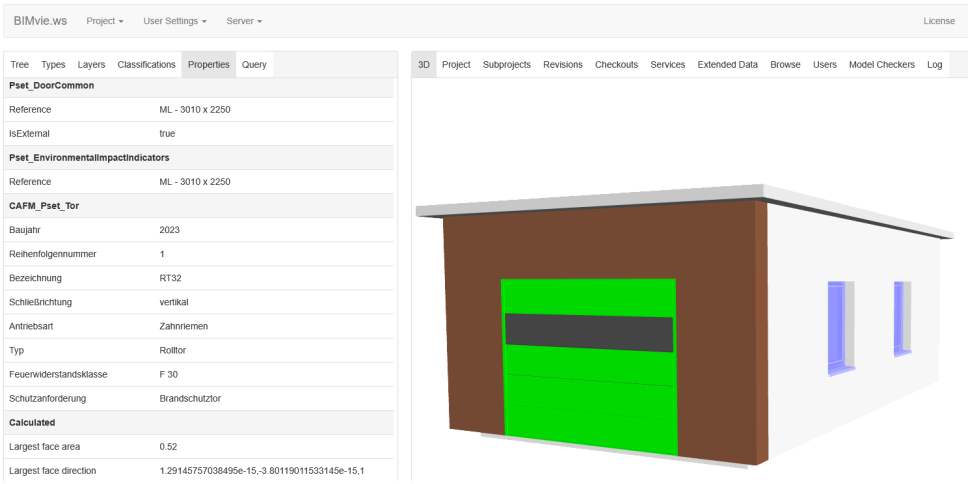


Figure 3: IFC model with entity properties in the BIMserver

property should have the value “true.” Figure 4 shows the IDS specifications in datacat, and Figure 5 shows an excerpt from the IDS file in XML syntax.

IfcWindows are classified as "Fenster"

Applicability: Type (IFC Class)

IFC Versions (Multiple Selection): IFC4

IFC Classes (Multiple Selection): IFCWINDOW

Requirements: [Classification] CAFM, [Property] OFT_Pset_Fenster

IfcSpace is classified as "Raum"

Applicability: Type (IFC Class)

IFC Versions (Multiple Selection): IFC4

IFC Classes (Multiple Selection): IFCSPACE

Requirements: [Property] OFT_Pset_Raum, [Classification] OpenFab

Figure 4: IDS specifications created in datacat

```
<ids:specification name="IfcWindows are classified as 'Fenster'" identifier="S3"
ifcVersion="IFC4">
 <ids:applicability maxOccurs="unbounded">
  <ids:entity>
   <ids:name><ids:simpleValue>IFCWINDOW</ids:simpleValue></ids:name>
  </ids:entity>
 </ids:applicability>
 <ids:requirements>
  <ids:property cardinality="required">
   <ids:propertySet><ids:simpleValue>OFT_Pset_Fenster</ids:simpleValue></ids:propertySet>
   <ids:baseName><ids:simpleValue>Schalldämmung</ids:simpleValue></ids:baseName>
   <ids:value>
    <xs:restriction base="xs:string"><xs:enumeration value="true" /></xs:restriction>
   </ids:value>
  </ids:property>
  <ids:classification>
   <ids:value>
    <xs:restriction base="xs:string"><xs:enumeration value="Fenster" /></xs:restriction>
   </ids:value>
   <ids:system>
    <xs:restriction base="xs:string"><xs:pattern value="CAFM" /></xs:restriction>
   </ids:system>
  </ids:classification>
 </ids:requirements>
</ids:specification>
```

Figure 5: Snippet of an IDS specification

The IDS validation plugin in BIMserver receives the URL of the IDS file from MinIO Object Storage. A check is performed automatically each time a new revision of the BIM model is uploaded. The plugin then generates an IDS report. Figure 6 shows an excerpt displaying a result for each specification

```
[FAIL] Check passed: 3/4 | Total elements: 2
IfcWindows are classified as "Fenster"
    Applies to:
        1.  All IFCWINDOW data
    Requirements:
        1.  Shall have a pattern: /CAFM/ reference of enums: [Fenster]
                [P] Class: IfcWindow | Name: FE 1 tlg - DrehKipp:1100 x 1400:476357 | GUID: 1s2y3MJjT8AuHONdaM2zhh
                [P] Class: IfcWindow | Name: FE 1 tlg - DrehKipp:1100 x 1400:476358 | GUID: 1s2y3MJjT8AuHONdaM2zhe
        2.  Schalldämmung data shall be provided in the dataset OFT_Pset_Fenster
                [P] Class: IfcWindow | Name: FE 1 tlg - DrehKipp:1100 x 1400:476358 | GUID: 1s2y3MJjT8AuHONdaM2zhe
                [F] The required property set does not exist | Class: IfcWindow | Name: FE 1 tlg - DrehKipp:1100 x 1400:476357 | GUID: 1s2y3MJjT8AuHONdaM2zhh

[PASS] Check passed: 2/2 | Total elements: 1
IfcSpace is classified as "Raum"
    Applies to:
        1.  All IFCSPACE data
    Requirements:
        1.  Shall have a pattern: /OpenFabTwin/ reference of enums: [Raum]
                [P] Class: IfcSpace | Name: 6 | GUID: 1s2y3MJjT8AuHONdaM2zji
        2.  Raumnummer data shall be provided in the dataset OFT_Pset_Raum
                [P] Class: IfcSpace | Name: 6 | GUID: 1s2y3MJjT8AuHONdaM2zji
```

Figure 6: Snippet of the IDS validation report in the BIMserver

in the IDS file. The room requirements passed. However, the report indicates that the property set "OFT_Pset_Fenster" is missing for one window. The user can use the report to identify which changes to the IFC model are still required to meet the project requirements.
This example illustrates a possible workflow for utilizing a single source of semantics—the property server—and making project information available via a BIM server. The workflow is applied throughout the project period to

1. define project requirements with consistent semantics encoded as IDS,
2. use the same semantics when creating the BIM model, and
3. semi-automatically check the current status of the BIM model against the IDS requirements in a BIM server.

There are also some limitations by the used software. The MinIO Object Storage works as a handoff point between the property server and the BIMserver. Using object storage or other third-party applications has the disadvantage that, if the IDS file is updated in datacat and uploaded again to MinIO, the shared link is not stable, and the new link has to be updated in the BIMserver. In the future, the IDS file could be stored in the BIMserver via a direct connection to property servers to expand the BIMserver into a Common Data Environment (CDE).
Further development of the BIMserver is necessary. A notable limitation of BIMserver is its absence of project-specific settings. Consequently, the IDS-model checking plugin implements a single IDS file across all projects. Moreover, the triggers option in BIMserver, which is designated for automatic checking, has the following specifications: it is only available when a new revision, project, or extended data is created. As a result, the update of the IDS files did not result in the automatic triggering of the checking process.
The datacat IDS module currently implements the entity, attribute, classification, and property facets. These cover the classification- and property-based requirements evaluated in this work. The material and partOf facets are not yet implemented, because no use case has required them so far. Both can be represented by the property server and are planned for future updates.
A general limitation of IDS is that geometry and schema validation are not possible with IDS.

5 Conclusion

The IDS report in BIMserver serves as proof for project stakeholders that all requirements have been fulfilled. It is centrally available alongside the current version of the BIM model and runs automatically whenever the model is modified. This ensures consistent quality assurance. However, further development of automatically generating a BCF (BIM Collaboration Format) from the report would be a useful addition, as it would allow for directly linking issues to model elements and integrating findings into existing BIM workflow. Since project requirements generally should not change during the project, it is sufficient for the project administrator to set the link to the IDS file at the beginning. Due to rights and role management, other employees do not influence the requirements in IDS.
In general, the workflow shows how property servers and open-source BIM servers could be used together in project management. However, the workflow has also shown that using a third-party application as an intermediary for the IDS files leads to certain problems, additional work, and an increased risk of errors. For this reason, it makes sense to establish a direct connection between the property server and the BIM server and store the IDS files on the latter.

Acknowledgements

This Project is supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK) on the basis of a decision by the German Bundestag (Funding-No.: 16KN106936). Supervised by HTW Berlin (Prof. Dr.-Ing. Helga Tauscher) and HTW Dresden (Prof. Dr.-Ing. Christian Clemen).

References

[1] Z. Fathis and H. Tauscher, “Contributor support for maintaining opensource software. A case study of BIMserver”, *Electronic Communications of the EASST*, vol. 85, 2025, ISSN: 1863-2122. DOI: 10.14279/eceasst.v85.2702

[2] M. Wogan, N. Mohan, S. Schilling, S. Jaud, K. Menzel, and R. Gross, “Management of model requirements across projects : An ontology and IDM / MVD-based approach across lifecycles”, in *EG-ICE 2025: AI-Driven Collaboration for Sustainable and Resilient Built Environments Conference Proceedings*, Glasgow, 2025. DOI: 10.17868/strath.00093233

[3] buildingSMART International, *Information Delivery Specification - buildingSMART International*, 2026. Accessed: May 15, 2026. [Online]. Available: https://www.buildingsmart.org/standards/bsi-standards/information-delivery-specification-ids/

[4] P. Pauwels, E. van den Bersselaar, and L. Verhelst, “Validation of technical requirements for a BIM model using semantic web technologies”, *Advanced Engineering Informatics*, vol. 60, 2024, ISSN: 1474-0346. DOI: 10.1016/j.aei.2024.102426

[5] Z. Ma, H. Zhu, X. Xiang, Ž. Turk, and R. Klinc, “Automatic compliance checking of BIM models against quality standards based on ontology technology”, *Automation in Construction*, vol. 166, p. 105 656, 2024, ISSN: 09265805. DOI: 10.1016/j.autcon.2024.105656

[6] A. Tomczak, L. van Berlo, T. Krijnen, A. Borrmann, and M. Bolpagni, “A review of methods to specify information requirements in digital construction projects”, *IOP Conference Series: Earth and Environmental Science*, vol. 1101, no. 9, 2022. DOI: 10.1088/1755-1315/1101/9/092024

[7] buildingSMART International. “Mvdxml”, Accessed: Jun. 23, 2026. [Online]. Available: https://technical.buildingsmart.org/standards/ifc/mvd/mvdxml/

[8] buildingSMART International. “IDS user manual: Entity facet”, Accessed: Jun. 23, 2026. [Online]. Available: https://github.com/buildingSMART/IDS/blob/development/Documentation/UserManual/entity-facet.md

[9] A. Fonsati, A. Dervishaj, and K. Gudmundsson, “Leveraging OpenBIM standards and information delivery specification (IDS) for digital validation in circular construction: Reusing hollow core slabs”, *Smart and Sustainable Built Environment*, 2026. DOI: 10.1108/SASBE-10-2025-0683

[10] E. A. de Mendonça and S. L. Ferreira, “Use case for Automated Code Compliance Checking of Accessibility Rules in BIM models using the IDS standard”, in *Proceedings of the 41st International Conference of CIB W78*, Marrakech, 2024. Accessed: Jun. 23, 2026. [Online]. Available: http://itc.scix.net/paper/w78-2024-21

[11] M. Kładź and A. S. Borkowski, “IDS Standard and bSDD Service as Tools for Automating Information Exchange and Verification in Projects Implemented in the BIM Methodology”, *Buildings*, vol. 15, no. 3, 2025, ISSN: 2075-5309. DOI: 10.3390/buildings15030378

[12] L. Guntermann, P. Hagedorn, and M. König, “Enhancing IFC Models for Circular Design: Validation of Required Information through an IDS Framework”, in *42nd International Symposium on Automation and Robotics in Construction*, Montreal, 2025. DOI: 10.22260/ISARC2025/0093

[13] buildingSMART International. “Referencing bSDD in IDS and IFC”, Accessed: Jun. 23, 2026. [Online]. Available: https://technical.buildingsmart.org/services/bsdd/referencing-bsdd-in-ids-and-ifc/

[14] J. Brouwer, A. Tomczak, and L. Casteleyn, *bSDD_dictionary_to_IDS: Convert a bSDD dictionary into an IDS*, GitHub repository, BIM-Tools, MIT License. Accessed: Jun. 23, 2026. [Online]. Available: https://github.com/BIM-Tools/bSDD_dictionary_to_IDS

[15] buildingSMART International. “IDS documentation: User manual”, Accessed: Jun. 23, 2026. [Online]. Available: https://github.com/buildingSMART/IDS/tree/master/Documentation

[16] R. de Laat and L. van Berlo, “Integration of BIM and GIS: The development of the CityGML GeoBIM extension”, in *Advances in 3D Geo-Information Sciences*, Berlin, Heidelberg, 2011, ISBN: 978-3-642-12669-7. DOI: 10.1007/978-3-642-12670-3_13