Christoph Carl Eichler | Christian Schranz | Tina Krischmann | Harald Urban

BIMcert Handbook

Basic Knowledge openBIM

Edition 2023

buildingSMART Austria

3.7 IDS – Information Delivery Specification

Guest authors: Léon van Berlo, Simon Fischer

IDS is a standard from buildingSMART International for the definition of computer-interpretable model exchange requirements. IDS is a relatively new standard and is complementary to MVD. While MVD deals with fundamental issues such as the correct representation of the class hierarchy and the geometry, IDS specifies the alphanumeric information of models. It defines the information requirements for objects. For this reason, IDS is a promising tool for providing and verifying client's information requirements defined in the EIR. It integrates the information requirements that currently exist as text into the automated openBIM process. IDS can be used for two sub-processes:

- Define information: As a configuration file for BIM authoring software, for the automated provision of the required information structure, and
- check information: As a configuration file for BIM checking software, for automated checking of the structure and content of the information.

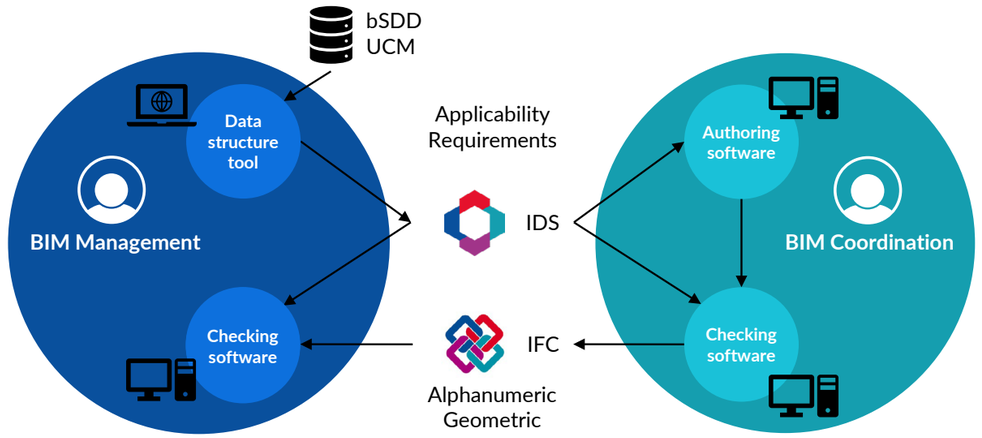
The IDS workflow starts with the client's area of responsibility (BIM management). The BIM management defines the desired BIM use cases and the required information. Let's look at two examples of information requirements.

First, a client might want all spaces in a model to be classified with a certain code, and have a couple of properties. The requirement could be described as »*All space data in a model shall be classified as [AT]Zimmer and have NetFloorArea and GrossFloorArea (both in set called BaseQuanitites) and a property called AT_Zimmernummer in the property set Austria_example.*« This is only an example. It could be any kind of requirement. Users can further refine requirements to not apply to all spaces but only to those with certain characteristics. For example, spaces with a certain property and/or property value, or spaces that are part of a certain hierarchy, or spaces that are classified in a certain way. This applies to all objects, not only spaces. The requirements for classification codes, materials, quantities, attributes, properties, materials and some relations can also be set for the selection (sometimes also called filtering; formally called applicability in IDS) of objects.

The applicability is included in the second example, a specification of certain properties for walls: »*All walls shall have the properties LoadBearing and FireRating (both in a property set called Pset_WallCommon). Walls with the value true for the property LoadBearing need a value for the property FireRating from the following list (ND, REI 30, REI 60, REI 90, REI 120).*« The space example is used later to show different ways of visualising IDS. The wall example is included in the description of the data structure of IDS in the next section.

The definition of information requirements is usually done using a data structure tool and taking into account data from the bSDD and the UCM. The BIM management then exports the information requirements in IDS and sends them to the contractor's BIM coordination or BIM creation. They use IDS as a configuration file for both the BIM authoring software and the BIM checking software.

This enables the authoring software to create the required properties on an object-specific basis automatically. In the BIM checking software, the configuration file enables automatic selection and filling of checking rules. The checked IFC file is finally sent to the BIM management, which also uses the IDS file to configure its checking software. In this way, IDS couples the client's information requirements with the BIM model and enables an automated checking of the defined information structure.



3.7.1 Data structure

The IDS file format is based on the XML scheme. It is a standardised form of it. This means that the structure and syntax of an IDS file are more precisely specified than those for a general XML file. For this purpose, buildingSMART International uses the XSD format (XML Schema Definition). This defines which elements must and may be included in an IDS file.

In principle, an IDS file is divided into two sections: a *Header* and a *list of Specifications*. The *Header* contains general metadata about the file. This is collected within the `info` element. Possible information in it are `title`, `copyright`, `version`, `description`, `author`, `date`, `purpose`, and `milestone`. Only the title is mandatory. All other parameters are optional. The lines before the metadata are the XML prolog for the definition of the XML version and the encoding, as well as the Root element (`<ids ...>`) with the definition of namespaces for the document.

```
<?xml version="1.0" encoding="UTF-8"?>
<ids xmlns="http://standards.buildingsmart.org/IDS" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance" xsi:schemaLocation="http://standards.buildingsmart.org/
IDS/ids_09.xsd">
  <info>
    <title>IDS for BIMcert</title>
    <copyright>Simon Fischer</copyright>
    <description>Created to describe IDS for BIMcert</description>
    <date>2023-01-11</date>
  </info>
```

The general metadata is followed by the actual content of an IDS file: a list of *Specifications*. *Specifications* describe the information requirements for elements in IFC. A *Specification* consists of three parts: *Metadata*, *Applicability*, and *Requirements*.

The **Metadata** is included as XML attributes in the *Specification element*. In the following example, the two mandatory parameters, name and ifcVersion, are included. In addition, the necessity (occurs), an identifier, a description, and instructions can be defined. The description and instructions are options to add human-readable documentation to the requirements. While IDS is designed to be interpreted by computers, in many cases humans will inevitably need to add information to the BIM dataset. The creator of an IDS can therefore leave instructions that clarify any requirements for a human also to input data. The second component of the *Specification* is the **Applicability**. This filter defines for which elements the current *Specification* is applicable. This restriction can be carried out at the level of IFC classes but also much more specifically via *predefined types*, *properties*, *materials*, etc. The third component of the *Specification* are the **Requirements**. These contain the actual information requirements for objects. The combination of *Applicability* and *Requirements* creates the computer-interpretable definition of information requirements. Both components use so-called *Facets* to specify their content. In the context of XML, *Facets* mean restrictions for XML elements. In the IDS scheme, *Facets* describe information that an element in the IFC model might have. Six precisely defined *Facet* Parameters are used to make the requirements computer-interpretable. The *Facet* Parameters refer to different contents of the IFC scheme:

- *Entity Facet*
- *Attribute Facet*
- *Classification Facet*
- *Property Facet*
- *Material Facet*
- *PartOf Facet*

In the *Applicability*, the *Facets* enable very specific filter options (e.g., only elements that have a certain *property* with a certain value). It is also possible to combine several *Facets*, which increases the possibilities for defining individual requirements.

Through all of this functionality, IDS can provide advanced definitions of requirements. It allows users to require properties to be shared with a certain kind of measure. There are also extensive possibilities to define restrictions on values as well. For example, the value of a property can only be selected from a list of allowed values. Or if the value is a number, it can have a specified minimum, maximum or range. Even pattern matching is an option available in IDS. IDS uses the XSD restrictions for this to improve the reliability of the implementation. Restrictions on specifications are another example of an advanced feature. The *minOccurs* and *maxOccurs* XML attributes allow users to define a minimum, maximum, range, or the exact number of objects that must be present in the BIM dataset. The *PartOf facet* allows users to require certain structures in the BIM dataset that are typical when using IFC. *Requirements* for an object to be part of an assembly or part of a group can be defined using this functionality. Details on the different facet parameters follow in a later section.

In the following, the example of information requirements for walls from the introduction is shown both in normal text in tabular form (common in several countries) and in IDS. The first *Specification* states that each wall requires the *properties* LoadBearing and FireRating in the *property set* Pset_WallCommon. The second *Specification* provides possible values for the fire resistance class of load-bearing walls (the list is not comprehensive). The *Applicability* of both specifications is highlighted in light blue, the *Requirements* in light orange.

LOI – Level of Information (IfcWall)

Property	Data type	Unit of value	Location	Selection set	Note
LoadBearing	IfcBoolean	Logical value	Pset_WallCommon	-	Default value: FALSE
FireRating	IfcLabel	Text	Pset_WallCommon	Selection set	Default value: ND; Example: REI 60
...					

Selection sets IfcWall FireRating

load bearing	non-bearing	...
ND	ND	
REI 30	EI 30	
REI 60	EI 60	
REI 90	EI 90	
REI120	EI120	
...	...	

```
<specifications>
  <specification name="IfcWall General" ifcVersion="IFC4">
    <applicability>
      <entity>
        <name>
          <simpleValue>IFCWALL</simpleValue>
        </name>
      </entity>
    </applicability>
    <requirements>
      <property measure="IfcBoolean">
        <propertySet>
          <simpleValue>Pset_WallCommon</simpleValue>
        </propertySet>
        <name>
          <simpleValue>LoadBearing</simpleValue>
        </name>
      </property>
      <!-- further properties -->
    </requirements>
  </specification>
```

```
<specification name="IfcWall FireRating for LoadBearing walls" ifcVersion="IFC4">
  <applicability>
    <entity>
      <name>
        <simpleValue>IFCWALL</simpleValue>
      </name>
    </entity>
    <property measure="IfcBoolean">
      <propertySet>
        <simpleValue>Pset_WallCommon</simpleValue>
      </propertySet>
      <name>
        <simpleValue>LoadBearing</simpleValue>
      </name>
      <value>
        <simpleValue>true</simpleValue>
      </value>
    </property>
  </applicability>
  <requirements>
    <property measure="IfcLabel">
      <propertySet>
        <simpleValue>Pset_WallCommon</simpleValue>
      </propertySet>
        <name>
          <simpleValue>FireRating</simpleValue>
        </name>
        <value>
          <xs:restriction base="xs:string">
            <xs:enumeration value="ND"/>
            <xs:enumeration value="REI 30"/>
            <xs:enumeration value="REI 60"/>
            <xs:enumeration value="REI 90"/>
            <xs:enumeration value="REI 120"/>
          </xs:restriction>
        </value>
      </property>
    </requirements>
  </specification>
 </specifications>
</ids>
```

3.7.2 Relation to the buildingSMART Data Dictionary

When a user receives an IDS from a client, they can check their own data against the requirements defined in the IDS. As mentioned earlier, the IDS can include human-readable explanations and instructions to help the receiving human understand the requirements. It is also possible in IDS to add a link (formally called a *Uniform Resource Identifier*, URI) with more information about a property or

classification code. This is where the relation to the buildingSMART Data Dictionary (bSDD) comes into the picture. A URI starting with *identifier.buildingsmart.org* refers to an object that can be found in the bSDD. By following this URI, the user can obtain more information about a property beyond the level of detail that can be specified within the IFC. The bSDD hosts detailed, standardised information about definitions, units, relations to other objects, etc. It does this for classification codes, properties (including attributes and quantities) and materials, for both international and national-specific standards. The options for defining restrictions on values in IDS are the same as those supported by bSDD. This allows seamless interaction between IDS and bSDD. Adding the URI to a property or classification code (or system) allows users, and in some cases, even computers, to gather more information about the requirement and the typical use of objects. More information about the bSDD can be found in the dedicated bSDD section of this book.

3.7.3 Facet parameters

This section covers the functionality and capabilities of the six *Facet Parameters*. For *Facets*, as for *Specifications*, the necessity (occurs) can be specified as an XML attribute. Some *Facets* also offer other specific XML attributes. The following description contains a sample code for each *Facet*. The first two samples are each included in the *Applicability* of a *Specification*. The others can be included in the same way in the *Applicability* or the *Requirements* of any *Specification*.

Entity Facet

The *Entity Facet* refers to the classes in the IFC scheme. It is, therefore, particularly important for defining the *Applicability*, as it describes for which IFC class a *Specification* is relevant. In addition to the mandatory name of the IFC class, a *predefinedType* of an element can optionally be specified in the *Entity Facet*. The following code snippet shows the use of the *Entity Facet* to define the *Applicability* of a *Specification* to all elements of the IFC class IfcDoor.

```
<applicability>
  <entity>
    <name>
      <simpleValue>IFCDOOR</simpleValue>
    </name>
  </entity>
</applicability>
```

Attribute Facet

The *Attribute Facet* deals with attributes that are included by default in IFC classes. Examples are the name of an element or the GUID. To use the facet, the name of the attribute must be specified. The value of the attribute is optional. If only a name is defined, the element must have an attribute with the specified name and any defined (non-empty) value. The following code snippet illustrates the use of the *Attribute Facet* to define the *Applicability* of a *Specification* to all elements of the IFC class IfcDoor with the name Entry.

```
<applicability>
  <entity>
    <name>
      <simpleValue>IFCDOOR</simpleValue>
    </name>
  </entity>
  <attribute minOccurs="1" maxOccurs="1">
    <name>
      <simpleValue>Name</simpleValue>
    </name>
    <value>
      <simpleValue>Entry</simpleValue>
    </value>
  </attribute>
</applicability>
```

Classification Facet
If other classification systems are used in addition to the classes of the IFC scheme, these can be taken into account with the *Classification Facet*. Examples of such external classification systems are Uniclass2015 or national systems. The *Classification Facet* allows the specification of a classification system and a reference code (how an object is classified within the system). Both parameters are optional. If no parameter is specified, an object must be classified in any system with any reference code. In addition, a URI can be added as an XML attribute of the Classification Element to link to further information. In this example, the system Uniclass2015 with an arbitrary reference code is required.

```
<classification minOccurs="1" maxOccurs="1">
  <system>
    <simpleValue>Uniclass2015</simpleValue>
  </system>
</classification>
```

Property Facet
The *Property Facet* is the counterpart to the *Attribute Facet* and refers to the *properties*. In addition, it can also be used to specify quantities. To define a requirement, the parameters *propertySet* (*quantitySet*), *property name* (*quantity name*), value, and data type (measure) are used. The value of the *property* is optional, like in the *Attribute Facet*. All other parameters are mandatory but note that the data type has to be specified as an XML attribute of the *Property Element*, not as an individual XML element like the others. A URI can also be added as an XML attribute to link, e.g., to the bSDD. The example *Specification* given here requires a property LoadBearing with the value true and the data type IfcBoolean in the *property set* Pset_WallCommon.

```
<property measure="IfcBoolean" minOccurs="1" maxOccurs="1">
  <propertySet>
    <simpleValue>Pset_WallCommon</simpleValue>
  </propertySet>
```



```
  <name>
    <simpleValue>LoadBearing</simpleValue>
  </name>
  <value>
    <simpleValue>true</simpleValue>
  <value>
</property>
```

Material Facet
When using restrictions regarding materials, remember that an object can consist of one or more materials. The *Material Facet* checks whether one of the materials of the corresponding object matches the specified material. There is only one optional parameter for the material within this *Facet*. If not defined, any material specification must be present. A URI can be used as an XML attribute of the Material Element to link to additional information about the material.

```
<material minOccurs="1" maxOccurs="1">
  <value>
    <simpleValue> ExampleMaterial</simpleValue>
  </value>
</material>
```

PartOf Facet
The *PartOf facet* can be used to specify *Relations* between objects. Relations are defined in IFC via classes starting with IfcRel.... In the *PartOf Facet*, a *Relation* can be specified via such a relation class and the IFC class to which the *Relation* refers. Note that the *Relation* is specified as an XML attribute of the *PartOf element*, not as an individual XML element like the others. The following code snippet shows a requirement that an element must be assigned to a floor. For this purpose, the Relation IfcRelContainedInSpatialStructure and the class IfcBuildingStorey are specified.

```
<partOf relation="IfcRelContainedInSpatialStructure" minOccurs="1" maxOccurs="1">
  <entity>
    <name>
      <simpleValue>IFCBUILDINGSTOREY</simpleValue>
    </name>
  </entity>
</partOf>
```

3.7.4 Simple Values and complex restrictions

In addition to the possibility of specifying requirements for different contents of the IFC scheme via the *Facets*, the requirements themselves can also be defined in different ways. For this purpose, IDS first distinguishes between Simple Values and Complex Restrictions. Simple Values are single values in the form of a text, a number or a logical value (true/false). Complex Restrictions allow the specification of several values and can be divided into four subcategories:

Enumeration
The Enumeration is used to specify a list of allowed values. The list can contain both text and numbers. Below is an example of specifying fire resistance classes for load-bearing walls (the list is not comprehensive).

```
<value>
  <xs:restriction base="xs:string">
    <xs:enumeration value="ND"/>
    <xs:enumeration value="REI 30"/>
    <xs:enumeration value="REI 60"/>
    <xs:enumeration value="REI 90"/>
    <xs:enumeration value="REI 120"/>
  </xs:restriction>
</value>
```

Pattern
A *Pattern* describes the order in which different characters may be arranged. This functionality is mainly applicable to naming conventions or naming schemes. A widespread method for defining such patterns, which is also used for IDS, are *Regular Expressions* (*Regex*). The following code snippet shows an example of a room naming convention. `[A-Z]` means the name begins with a capital letter. `[0-9]{2}` specifies that it is followed by two digits between 0 and 9. `-[0-9]{2}` states that the name has to end with a hyphen and two digits between 0 and 9. Valid names are, e.g., W01-01 or B18-74.

```
<value>
  <xs:restriction base="xs:string">
    <xs:pattern value="[A-Z][0-9]{2} -[0-9]{2}"/>
  </xs:restriction>
</value>
```

Bounds
Bounds define an interval of valid values. It is possible to specify either a lower limit, an upper limit, or both. The limits can also be defined as exclusive `</>` or inclusive `<=/>=`.

Length
Finally, it is possible to specify the length of a value, i.e., the number of individual characters. You can specify an exact length as well as a minimum or maximum length.

3.7.5 Scope and use of IDS
An IDS file can contain multiple requirements. These requirements are independent »blocks« and have no reference to other requirements in the file. This is intentionally done to create the ability to copy-paste requirements between files. At the time of writing, several software vendors are implementing IDS editors and authoring tools to facilitate users with an easy way to create IDS files. In the future, buildingSMART envisages the existence of IDS libraries where examples of individual requirements are shared for everyone to use. Users will be able to search for IDS requirements and drag them into a selection basket to create their own IDS file. An important scope definition of IDS is that it focuses only on »information delivery specifications«. This means that the IDS structured requirements can define what information is needed and how it should be structured. It is important for automated workflows and scripts to receive information in such a way that it can be processed automatically, and this is the aim of IDS. However, IDS cannot be used to define design requirements or so-called »*rules*«. So, a requirement that all windows in a toilet room need to have an opaque glass is not possible within IDS; but a requirement that all windows need to have a property that defines what type of glass is in the window is a perfect definition to define in IDS. A rule checker or other algorithm should then be used to check whether windows in toilet rooms have an opaque glass or not. There is a grey area on this since IDS allows restrictions of values. Future releases of IDS will further refine this scope or extend the ability of IDS to define rules. Practical use cases and consensus will define the future possibilities of IDS.

3.7.6 New possibilities with IDS
In addition to the integration of information requirements into the automated openBIM process, IDS also offers new possibilities for the specific definition of these requirements by the *Applicability*. Typically, EIR define information requirements based on IFC classes and *predefined types*. In contrast, IDS can define information requirements depending on all described *Facets*. For example, a certain *property* in a particular *property set* only becomes necessary when another *property* in another *property set* has a certain value. This enables clients to request and check information very specifically.

3.7.7 IDS in depth
All technical information about IDS can be found on GitHub, where the code development, documentation, and examples are stored. The international community has identified IDS as the most advantageous method for automated compliance checking by validation of the alphanumerical information requirements. It supports information requirements authoring by providing users with a set of possibilities on what can be required of the models.

3.7.8 Relationship with other initiatives
There are many ways to define information requirements. Excel seems to be the most popular, but it has limitations. Other initiatives are the Product Data Templates (PDT), Level of Information Need (LOIN), Exchange (or Employer) Information Requirements, BIM Execution plans, the »*exchanges*« part of mvdXML, SHACL in the linked data domains, and more. All of these initiatives have advantages and limitations. Depending on the use case, other standards or initiatives may be a better choice. A comparison created by Tomczak et al. can be found here (see QR code).

Chapter 3 – In-depth knowledge
3.7 IDS – Information Delivery Specification

○ – No
◑ – Partial
● – Yes
* – under development

	Standardised	Applicability	Fields					Value constraints				Content			Geom.		Metadata		
			Info. type	Data type	Unit of meas.	Description	References	Equality	Range	Enumeration	Patterns	Existence	Documents	Structure	Representation	Detailedness	Purpose	Actors	Process map
Spreadsheet	○	◑	◑	◑	◑	◑	◑	◑	○	○	○	○	○	○	○	○	◑	○	○
PDT*	●	◑	◑	●	●	◑	●	○	◑	●	○	◑	○	○	○	○	◑	○	○
Data Dict.	●	○	●	●	◑	●	●	●	◑	●	◑	○	○	○	○	○	○	○	○
IDS*	●	●	●	●	●	◑	◑	●	●	●	●	●	○	◑	○	○	●	◑	○
mvdXML	●	●	●	●	●	◑	○	●	●	◑	◑	◑	○	●	●	○	○	◑	○
idmXML	●	◑	◑	◑	◑	●							●		◑	◑	●	●	●
LOIN*	●	◑	◑	●	●	◑	◑	○	○	●	○	○	●	◑	◑	●	●	●	○
IFC P.T.	●	◑	◑	●	●	◑	○	○	○	●	○	○	○	○	○	○	○	○	○
LD+SHACL	○	●	●	●	●	◑	●	●	●	●	●	◑	◑	●	●	○	◑	◑	○

For most use cases in openBIM, the IDS is the recommended solution for defining information requirements. It balances compatibility with IFC and bSDD with ease of use and reliability. Several software tools are available to check an IFC file against the requirements of an IDS file. Typically, the results are displayed in a viewer. To share the results, it is recommended to use the BIM Collaboration Format (BCF). BCF is a structured way of sharing information about IFC objects with project partners. For more information on BCF, see another section in this book.

3.7.9 Different ways to visualise IDS

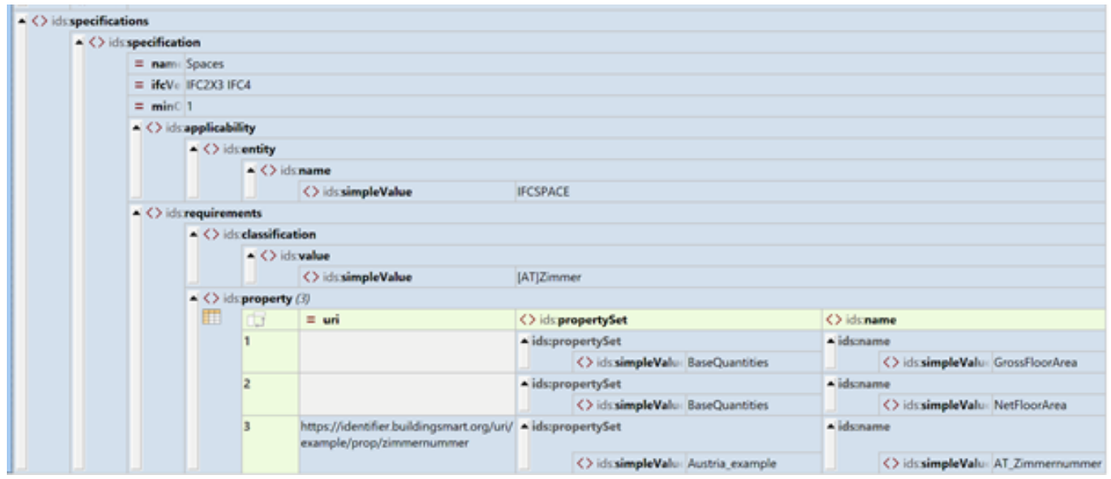
In this section, the information requirement example for spaces from the introduction is used to show different ways how to visualise IDS. The requirement states: »*All space data in a model shall be classified as [AT]Zimmer and have NetFloorArea and GrossFloorArea (both in set called BaseQuanitites) and a property called AT_Zimmernummer in the property set Austria_example*.« Formatting this human-readable requirement in an IDS looks like this:

```
<ids:ids xmlns:xs="https://www.w3.org/2001/XMLSchema" xmlns:ids=" http://
standards.buildingsmart.org/IDS">
  <ids:info>
    <ids:title>Austia example</ids:title>
    <ids:copyright>buildingSMART</ids:copyright>
    <ids:version>0.0.3</ids:version>
    <ids:description>A few example checks</ids:description>
    <ids:author>contact@buildingsmart.org</ids:author>
    <ids:date>2023-01-16+01:00</ids:date>
  </ids:info>
```



```
<ids:specifications>
  <ids:specification minOccurs="1" ifcVersion="IFC2X3 IFC4" name="Spaces">
    <ids:applicability>
      <ids:entity>
        <ids:name>
          <ids:simpleValue>IFCSPACE</ids:simpleValue>
        </ids:name>
      </ids:entity>
    </ids:applicability>
    <ids:requirements>
      <ids:classification>
        <ids:value>
          <ids:simpleValue>[AT]Zimmer</ids:simpleValue>
        </ids:value>
      </ids:classification>
      <ids:property>
        <ids:propertySet>
          <ids:simpleValue>BaseQuantities</ids:simpleValue>
        </ids:propertySet>
        <ids:name>
          <ids:simpleValue>GrossFloorArea</ids:simpleValue>
        </ids:name>
        </ids:property>
        <ids:property>
          <ids:propertySet>
            <ids:simpleValue>BaseQuantities</ids:simpleValue>
          </ids:propertySet>
          <ids:name>
            <ids:simpleValue>NetFloorArea</ids:simpleValue>
          </ids:name>
        </ids:property>
        <ids:property url="https://identifier.buildingsmart.org/url/example/prop/
zimmernummer">
          <ids:propertySet>
            <ids:simpleValue>Austria_example</ids:simpleValue>
          </ids:propertySet>
          <ids:name>
            <ids:simpleValue>AT_Zimmernummer</ids:simpleValue>
          </ids:name>
        </ids:property>
      </ids:requirements>
    </ids:specification>
  </ids:specifications>
</ids:ids>
```

A different way to visualise this XML is shown in the figure. Here you can see the same information but structured as a table. This is a very generic view that can be applied to all XML files.



There are also specific viewers that read the XML-based IDS and visualise it in a human-readable way. In such a viewer, our example looks like this. As you can see, there are many different ways to visualise the information in an IDS file.

Austria example

contact@buildingsmart.org 0.0.3 2023-01-16+01:00 Construction

A few example checks

© buildingSMART

Spaces

Describe why the requirement is important to the project.

Provide instructions on who is responsible and how to achieve it.

APPLIES TO:

All *Space* data

REQUIREMENTS:

Shall be classified as *[AT]Zimmer*

Gross Floor Area data shall be provided in the dataset *BaseQuantities*

Net Floor Area data shall be provided in the dataset *BaseQuantities*

A T_ Zimmernummer data shall be provided in the dataset *Austria_example*

3.7.10 Relationship IDS to IFC

Although IDS can be used to request any type of data in the build asset industry, it works best on data that is structured according to the IFC standard. As you see in the space requirement example (in the line *specification*), this specification requires there to be at least one object like this in the model. It also states that this requirement is made for both IFC2x3 and IFC4. The applicability part of this IDS also states IFCSPACE. This is an IFC entity. So although the specification can be used for non-IFC data, the IDS tends to prefer specifications that are made on IFC. This can also be seen in the split between attributes and properties, and the *PartOf* relationships in advanced requirements.