| Document id | Title | Organisation /Author | Date | Status |
|---|---|---|---|---|
| | ifcXML4 Specification Methodology | Thomas Liebich, Matthias Weise | 2013-07-22 | Final |

# ifcXML4
# Specification Methodology

Specification of the mapping methodology from
EXPRESS to XML schema definition for IFC4

developed by
Model Support Group (MSG) of buildingSMART International Ltd.

authors
Liebich, Thomas; Weise, Matthias

Version 1.1
22. July 2013

# Document history

| | | |
|---|---|---|
| 1.1 | First revised release based on IFC4 Final<br>• Updated namespaces and locations<br>• New configuration option<br>• Improvements on examples | 2013-07-22 |
| 1.0 | First public release based on IFC4 RC4<br>• Manually added configuration setting for specific SELECT types changed from *type-tag* to *double-tag* | 2012-10-04 |
| 0.9 | First release for internal discussion and submission to the project sponsors | 2012-07-09 |

# Table of Content

# Table of Tables

# 1 Overview

With the new development of the IFC4 standard, the XML equivalent to the EXPRESS based specification, called ifcXML, has undergone a complete redevelopment. The goal has been to simplify the corresponding XML schema, the ifcXML XSD, and to obtain smaller and more compact XML data files. The development has been executed under the codeword "simple ifcXML".

## 1.1 buildingSMART objectives

buildingSMART as developer and publisher of the openBIM standard IFC (equivalent to ISO16739) has the desire to publish the data schema behind IFC alternatively as an XML schema. While an XSD has already been made available since the IFC2x release, published in 2001, the previous ifcXML schemas have been rather complex, sometimes cumbersome and leading to large and inefficient XML data files.

Meanwhile the official ISO 10303-28 standard is available that includes a configuration capability that enables schema developers to reduce some of the overhead coming from converting an EXPRESS schema (such as IFC) into an XML schema. While a previous working group version of ISO 10303-28 has already been used for IFC2x2 and IFC2x3 XML schemas, those do not take advantage of the new configuration capabilities.

At the beginning of the "simple ifcXML" development project the goal was to enable the enhanced use of the configuration capabilities only in scope of a Model View Definition. Based on the encouraging intermediate results the decision was made to also apply it to the complete IFC4 schema as "ifcXML4".

> ITM Resolution 49-1: ifcXML4 should already implement all simplifications besides nesting [remark: refers to inverting an inverse relationship]. Purpose specific ifcXML schemas [refers to an ifcXML schema for a specific Model View Definition] implement nesting for specific exchanges.

## 1.2 Development strategy for ifcXML

The IFC datasets are converted into the new ifcXML4 structure according to the rules described in this document. Following these rules would result into conformance to the ifcXML4 schema definition.

The conversion rules are a subset of the configured ISO10303-28 XML language binding of EXPRESS based schemas and data. They are published as a configuration file that conforms to chapter 10 of ISO10303-28. The same configuration file is used to generate the ifcXML XSD and the actual XML data files (if generated by a toolbox based on the EXPRESS schema).

## 1.3 buildingSMART development task

buildingSMART International will publish the official ifcXML XSD for all official releases of the IFC data model standard. The development task is carried out by the buildingSMART Model Support Group responsible for IFC and other data model standards. It is also anticipated that buildingSMART International will publish simple ifcXML XSD's for all official Model View Definitions that require an XML based data exchange.

## 1.4 Software developer tasks

A software developer intending to provide compliant ifcXML solutions has to export IFC data that validates against the ifcXML XSD. Providers of IFC Toolboxes, software development tools that provide basic API functionality to export and import IFC data, may also offer ifcXML capabilities. In this case, the toolbox handles the correct formatting of data according to the ifcXML XSD.

# 2 Conversion rules for ifcXML4 for the entire IFC schema

The following chapter applies to the entire schema for default data exchange, which is generated and published by buildingSMART International. Beside the conversion rules for the entire IFC schema the chapter 3 describes additional configuration settings for Model View Definitions, i.e. sub schemas that support specific data exchange use cases.

## 2.1 Constructs not mapped into ifcXML schema

According to the scope and definitions of ISO10303-28, the selected configuration file, and the XML Schema in general, the following constructs within the IFC EXPRESS schema are not mapped.

- All EXPRESS constraints (WHERE rules, UNIQUE rules, global RULE's)
- All EXPRESS FUNCTION declarations
- All EXPRESS attributes being INVERSE (with exceptions) or DERIVE attributes

  NOTE Depending on the configuration file, some INVERSE attributes may be mapped into XML Schema constructs by inverting the EXPRESS attribute relationship. For data exchange it is sufficient to provide only one direction of bidirectional attributes. In that respect ifcXML-based data exchange is similar to STEP-based data exchange, but in some cases uses the INVERSE attribute instead of the explicit attribute. The decision to use an INVERSE attribute is taken case-by-case in order to simplify the XML data structure.

## 2.2 Creation of ifcXML schema and root element

The ifcXML4 shall only contain a single unit of serialization, i.e. the data section of a single ifc file. It is therefore structured as a *"uos document"* file according to ISO 10303-28 chapter 4.1.3 and 9.2.2. The unit of serialization may contain a *header* element that corresponds to the HEADER section of an ifc file. The name of the unit of serialization element, the root element for each ifcXML4 file, is governed by the *uosElement* configuration option:

Configuration for ifcXML4:

```
<cnf:uosElement name="ifcXML"/>
```

The name space of the ifcXML4 schema is governed by the configuration file, the actual name space designator is provided by the buildingSMART Model Support Group when publishing an official ifcXML release. The schema location of the XML Schema for validating ifcXML4 files is determined at the same time and the XML Schema is uploaded to the official buildingSMART server.

Configuration for ifcXML4:

```
<cnf:schema targetNamespace="http://www.buildingsmart-tech.org/ifcXML/IFC4/final"
   elementFormDefault="qualified" attributeFormDefault="unqualified">
   embed-schema-items="true"
   <cnf:namespace prefix="ifc"
      alias="http://www.buildingsmart-tech.org/ifcXML/IFC4/final"/>
</cnf:schema>
```

The additional configuration option *embed-schema-items="true"* determines, that the resulting XML Schema is a single file having a single target name space. It incorporates the ISO 10303-28 base definitions into the target name space.

XSD of the unit of serialization

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:ifc="http://www.buildingsmart-tech.org/ifcXML/IFC4/final"
   targetNamespace="http://www.buildingsmart-tech.org/ifcXML/IFC4/final"
   elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:element name="uos" type="ifc:uos" abstract="true"/>
   <xs:complexType name="uos" abstract="true">
      <xs:sequence>
         <xs:element name="header" minOccurs="0">
            <!-- header element declaration -->
         </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="optional"/>
      <xs:attribute name="express" type="ifc:Seq-anyURI" use="optional"/>
      <xs:attribute name="configuration" type="ifc:Seq-anyURI" use="optional"/>
   </xs:complexType>
```

XSD of the target name space specific unit of serialization, the ifcXML element

```
   <xs:element name="ifcXML" type="ifc:ifcXML" substitutionGroup="ifc:uos"/>
   <xs:complexType name="ifcXML">
      <xs:complexContent>
         <xs:extension base="ifc:uos">
            <xs:choice minOccurs="0" maxOccurs="unbounded">
               <xs:element ref="ifc:Entity"/>
            </xs:choice>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
```

ifcXML file (using the URL published by buildingSMART MSG for schema location)

```
<?xml version="1.0" encoding="UTF-8"?>
<ifcXML
   xmlns="http://www.buildingsmart-tech.org/ifcXML/IFC4/final"
   xmlns:ifc="http://www.buildingsmart-tech.org/ifcXML/IFC4/final"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.buildingsmart-tech.org/ifcXML/IFC4/final
   http://www.buildingsmart-tech.org/ifcXML/IFC4/final/ifcXML4.xsd"
   id="ifcXML4">
   <!-- any content -->
</ifcXML>
```

**Table 1: ifcXML4 schema information**

~~NOTE An URL for the final XSD schema location will be published by buildingSMART MSG after finalizing the XSD schema definition.~~

NOTE The 'schemaLocation' enables to check an ifcXML file against the IFC XML schema definition. Further information about the EXPRESS source schema and used configuration settings can be added through the optional attributes 'express' and 'configuration'. Similar to 'schemaLocation' it should either be a single URI, or a list of URIs of which the first is a URN and all of the others are possible locations of the text.

## 2.3 Mapping of HEADER information

The unit of serialization element, i.e. the *ifcXML* element, may contain administrative data within the *header* element. The information captured in the header elements corresponds to the information within the HEADER section of an ifc file, as described in ISO 10303-21.

NOTE The 'description' attribute of the FILE_DESCRIPTION entity (being a set of string) in the HEADER section of an ifc file is mapped into the documentation element (being a single string). The target string is a concatenation of the source strings.

XSD of the *header* element

```
<xs:element name="header" minOccurs="0">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="name" type="xs:string" minOccurs="0"/>
         <xs:element name="time_stamp" type="xs:dateTime" minOccurs="0"/>
         <xs:element name="author" type="xs:string" minOccurs="0"/>
         <xs:element name="organization" type="xs:string" minOccurs="0"/>
         <xs:element name="preprocessor_version" type="xs:string" minOccurs="0"/>
         <xs:element name="originating_system" type="xs:string" minOccurs="0"/>
         <xs:element name="authorization" type="xs:string" minOccurs="0"/>
         <xs:element name="documentation" type="xs:string" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

IFC file

```
HEADER;
FILE_DESCRIPTION(( 'ViewDefinition [notYetAssigned]',
   'Comment [manual creation of example file]'),'2;1');
FILE_NAME(
   'basic_shape_CSG.ifc',
   '2012-06-18T18:00:00',
   ('Thomas Liebich'),
   ('buildingSMART International'),
   'IFC text editor',
   'IFC text editor',
   'reference file created for the IFC4 specification');
FILE_SCHEMA(('IFC4_RC4'));
ENDSEC;
```

ifcXML file

```
<header>
   <name>basic_shape_CSG.ifcxml</name>
   <time_stamp>2012-06-19T00:40:06</time_stamp>
   <author>Thomas Liebich</author>
   <organization>buildingSMART International</organization>
   <preprocessor_version>ifcXML text editor</preprocessor_version>
   <originating_system>IFC text editor</originating_system>
   <authorization>reference file created for the IFC4 specification</authorization>
   <documentation>ViewDefinition [notYetAssigned]
      Comment [manual creation of example file]</documentation>
</header>
```

Table 2: ifcXML4 header information

## 2.4 Mapping of EXPRESS entities

Instance elements can be written "by-value" and "by-reference".

- "by-value" instance elements shall define a local XML identifier, which is defined by the *id* attribute and can be used as a reference.
- "by-reference" instance elements contain a reference to a "by-value" instance element, which is defined by the *ref* attribute.

  NOTE An entity instance shall be written exactly once as a “by-value” instance element, which is similar to an .ifc file. To ensure data integrity it is required that the referenced "by-value" instance element is contained in the same ifcxml file.

XSD *nillable* definition of an element to be used “by-value” and “by-reference”

```
<xs:element name="IfcCartesianPoint" type="ifc:IfcCartesianPoint"
     substitutionGroup="ifc:IfcPoint" nillable="true"/>
```

ifcXML file using “by-value” and “by-reference” instance elements for *IfcCartesianPoint* ”i1942”

```
<IfcLocalPlacement id="i1937">
  <IfcAxis2Placement3D id="i1940">
    <Location xsi:nil="true" ref="i1942" xsi:type="IfcCartesianPoint"/>
  </IfcAxis2Placement3D>
</IfcLocalPlacement>
<IfcCartesianPoint id="i1942" Coordinates="0. 0. 0."/>
```

**Table 3: Example for “by-value” and “by-reference” instance elements**

## 2.5 Mapping of EXPRESS attributes

The EXPRESS attributes that are declared within the EXPRESS entity are mapped either into XML attributes, or into a sequence of inline XML elements. Only the explicit EXPRESS attributes are mapped by default, the inverse (with exceptions) and derived EXPRESS attributes are not mapped.

NOTE The inverse and derived EXPRESS attributes are also not included in an .ifc file.

The detailed mapping specification for EXPRESS attributes depends on their data type and on whether they are aggregates or not.

The conversion is determined by the *exp-attribute* configuration at attribute scope or, if such configuration is not provided, by *concrete-attribute*, *entity-attribute* in <option> scope. The *concrete-attribute* and *entity-attribute* configurations determine the default handling of EXPRESS attributes for the entire IFC schema.

- The *concrete-attribute* configuration applies a default mapping to all EXPRESS attributes having a non-entity datatype (e.g. having a simple or enumerated data type). The setting is *"attribute-content"* which results into mapping all attributes with non-entity datatype into XML attributes. Exceptions apply to aggregated attributes of non-entity datatype (in particular to aggregates of strings).
- The *entity-attribute* configuration applies a default mapping to all EXPRESS attributes having an entity data type (including attributes of SELECT datatype). The setting is "double-tag" which results into creating a pair of nested XML elements, one for the attribute, and one for the entity that populates it. Exceptions apply to all attributes with entity datatype that are not an aggregate.

The *tagless* attribute determines the handling of aggregated attributes only and is explained in 2.5.2.

Configuration for ifcXML4:

```
<cnf:option  concrete-attribute="attribute-content"
             entity-attribute="double-tag"
             tagless="unspecified" />
```

### 2.5.1 Non-aggregated EXPRESS attributes

Non-aggregated EXPRESS attributes are mapped according to their data type.

#### 2.5.1.1 Simple and Enumerated data type

All EXPRESS attributes of a given ENTITY that are non-aggregates and have a fundamental data type being a simple or enumerated type, are translated into an XML attribute of the XML element corresponding to the given ENTITY.

Configuration for ifcXML4:

Stated for all such EXPRESS attributes within the global <option> scope.

```
<cnf:option concrete-attribute="attribute-content" />
```

EXPRESS (skipping non explicit attributes and constraints)

```
ENTITY IfcActorRole;
   Role : IfcRoleEnum;
   UserDefinedRole : OPTIONAL IfcLabel;
   Description : OPTIONAL IfcText;
END_ENTITY;
```

XSD

```
<xs:element name="IfcActorRole" type="ifc:IfcActorRole"
   substitutionGroup="ifc:Entity" nillable="true"/>
<xs:complexType name="IfcActorRole">
   <xs:complexContent>
      <xs:extension base="ifc:Entity">
         <xs:attribute name="Role" type="ifc:IfcRoleEnum" use="optional"/>
         <xs:attribute name="UserDefinedRole" type="ifc:IfcLabel" use="optional"/>
         <xs:attribute name="Description" type="ifc:IfcText" use="optional"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

IFC File

```
#20000= IFCACTORROLE(.ARCHITECT., $, 'general architect');
```

ifcXML File

```
<IfcActorRole id="i1546" Role="architect"  Description="general architect"/>
```

**Table 4: Example of non-aggregated simple and enumerated attributes in ifcXML4**

NOTE According to conversion rules of ISO10303-28 all enumeration items are in lower case.

NOTE The values of the *id* attributes in the ifcXML file do not have to match the instance numbers (#) within the ifc file. An id attribute has to start with a non-numeric character, hence the convention to have an "i" as prefix.

#### 2.5.1.2 SELECT data type

All EXPRESS attributes of a given ENTITY that are non-aggregates and have a fundamental data type being a SELECT type, are translated into a pair of nested XML elements, one corresponding to the attribute name, and the second corresponding to the name of the data type of select item instantiated. It is either the name of the entity in case of an entity data type, or of a specific wrapper element in case of a non-entity data type. The SELECT datatype corresponds to an *xs:group* definition in XML schema.

NOTE If the select item is a simple, defined or enumerated data type, the corresponding XML element has the name of the EXPRESS data type with the suffix "-wrapper".

Configuration for ifcXML4:

Stated for all such EXPRESS attributes within the global <option> scope.

```
<cnf:option entity-attribute="double-tag"/>
```

EXPRESS

```
ENTITY IfcTextStyleForDefinedFont
 SUBTYPE OF (IfcPresentationItem);
   Colour : IfcColour;
   BackgroundColour : OPTIONAL IfcColour;
END_ENTITY;
TYPE IfcColour = SELECT
   (IfcColourSpecification
   ,IfcPreDefinedColour);
END_TYPE;
```

XSD

```
<xs:element name="IfcTextStyleForDefinedFont"
   type="ifc:IfcTextStyleForDefinedFont"
   substitutionGroup="ifc:IfcPresentationItem" nillable="true"/>
<xs:complexType name="IfcTextStyleForDefinedFont">
   <xs:complexContent>
      <xs:extension base="ifc:IfcPresentationItem">
         <xs:sequence>
            <xs:element name="Colour">
               <xs:complexType>
                  <xs:group ref="ifc:IfcColour"/>
               </xs:complexType>
            </xs:element>
            <xs:element name="BackgroundColour" nillable="true" minOccurs="0">
               <xs:complexType>
                  <xs:group ref="ifc:IfcColour"/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

IFC File

```
#314= IFCTEXTSTYLEFORDEFINEDFONT(#315, $);
#315= IFCCOLOURRGB('NOTE colour',1.,0.,0.);
```

ifcXML File

```
<IfcTextStyleForDefinedFont id="i34">
  <Colour>
    <IfcColourRgb id="i35" Name="NOTE colour" Red="1." Green="0." Blue="0."/>
  </Colour>
</IfcTextStyleForDefinedFont>
```

**Table 5: Example of non-aggregated SELECT type attributes in ifcXML4**

#### 2.5.1.3 ENTITY data type

All EXPRESS attributes of a given ENTITY that are non-aggregates and have a fundamental data type being an entity type, are translated into an XML element within the XML element corresponding to the given ENTITY. The name of the element representing the entity shall be the name of the EXPRESS attribute.

NOTE If the entity type has subtypes, the instantiated XML document shall contain the element with an *xsi:type* attribute holding the entity (sub)type name that is instantiated.

Configuration for ifcXML4:

Stated for all such EXPRESS attributes individually. One configuration setting is required for each attribute. It overrides the general setting of *entity-attribute* in <option> scope. This <attribute> configuration setting is auto-generated from the IFC EXPRESS schema.

Manual configuration option for ifcXML4:

```
<cnf:entity select="name of entity">
  <cnf:attribute select="name of attribute" exp-attribute="attribute-tag" />
</cnf:entity>
```

Example:

```
<cnf:entity select="IfcPlacement">
  <cnf:attribute select="Location" exp-attribute="attribute-tag" />
</cnf:entity>
```

EXPRESS (skipping inheritance and constraints)

```
ENTITY IfcPlacement;
 ABSTRACT SUPERTYPE OF (...)
 SUBTYPE OF (IfcGeometricRepresentationItem);
   Location : IfcCartesianPoint;
END_ENTITY;
```

XSD

```
<xs:element name="IfcPlacement" type="ifc:IfcPlacement" abstract="true"
  substitutionGroup="ifc:IfcGeometricRepresentationItem" nillable="true"/>
<xs:complexType name="IfcPlacement" abstract="true">
  <xs:complexContent>
    <xs:extension base="ifc:IfcGeometricRepresentationItem">
      <xs:sequence>
        <xs:element name="Location" type="ifc:IfcCartesianPoint"
          nillable="true"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

IFC File (*IfcAxis2Placement3D* as a subtype of *IfcPlacement*)

```
#46= IFCAXIS2PLACEMENT3D(#42,$,$);
#42= IFCCARTESIANPOINT((0.,0.,0.));
```

ifcXML File (by-reference representation)

```
<IfcAxis2Placement3D id="i1940">
   <Location xsi:nil="true" ref="i1942"/>
</IfcAxis2Placement3D>
<IfcCartesianPoint id="i1942" Coordinates="0. 0. 0."/>
```

ifcXML File (by-value representation)

```
<IfcAxis2Placement3D id="i1940">
   <Location id="i1942" Coordinates="0. 0. 0." xsi:type="IfcCartesianPoint"/>
</IfcAxis2Placement3D>
```

**Table 6: Example of non-aggregated entity-type attribute**

### 2.5.2 Aggregated EXPRESS attributes

The conversion is determined by the *exp-attribute* configuration (*concrete-attribute*, *entity-attribute* in <option> scope), and the *tagless* configuration attribute.

#### 2.5.2.1 Simple and Enumerated data type

All EXPRESS attributes of a given ENTITY that are aggregates and have a fundamental data type being a simple type of INTEGER, REAL, NUMBER, BOOLEAN or LOGICAL or an enumerated type, are translated into an XML attribute of the XML element corresponding to the given ENTITY. The value of that XML element is a space delimited list of such simple values or enumerators.

Configuration:

Stated for all such EXPRESS attributes within the global <option> scope.

```
<cnf:option concrete-attribute="attribute-content" tagless="unspecified"/>
```

EXPRESS (skipping derived attributes and constraints)

```
ENTITY IfcCartesianPoint
 SUBTYPE OF (IfcPoint);
   Coordinates : LIST [1:3] OF IfcLengthMeasure;
END_ENTITY;

TYPE IfcLengthMeasure = REAL;
END_TYPE;
```

XSD

```
<xs:element name="IfcCartesianPoint" type="ifc:IfcCartesianPoint"
   substitutionGroup="ifc:IfcPoint" nillable="true"/>
<xs:complexType name="IfcCartesianPoint">
   <xs:complexContent>
      <xs:extension base="ifc:IfcPoint">
         <xs:attribute name="Coordinates" use="optional">
            <xs:simpleType>
               <xs:restriction>
                  <xs:simpleType>
                     <xs:list itemType="ifc:IfcLengthMeasure"/>
                  </xs:simpleType>
                  <xs:maxLength value="3"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
<xs:simpleType name="IfcLengthMeasure">
   <xs:restriction base="xs:double"/>
</xs:simpleType>
```

IFC File

```
#42= IFCCARTESIANPOINT((0.,0.,0.));
```

ifcXML File

```
<IfcCartesianPoint id="i1942" Coordinates="0. 0. 0."/>
```

**Table 7: Example of aggregated simple data type**

All EXPRESS attributes of a given ENTITY that are aggregates and have a fundamental data type being a simple type of STRING or BINARY are treated by default as a sequence of inline XML elements having the name of the EXPRESS defined type, or the generated name for an EXPRESS simple data type.

NOTE These names get the suffix "-wrapper". The outline element grouping the aggregate has the name of the EXPRESS attribute.

Configuration for ifcXML4:

Stated for all such EXPRESS attributes within the global <option> scope.

```
<cnf:option concrete-attribute="attribute-content" tagless="unspecified"/>
```

EXPRESS (skipping other attributes and constraints)

```
ENTITY IfcPostalAddress;
 SUBTYPE OF (IfcAddress);
   ..
   AddressLines : OPTIONAL LIST [1:?] OF IfcLabel;
   ..
END_ENTITY;
```

XSD

```
<xs:element name="IfcPostalAddress" type="ifc:IfcPostalAddress"
   substitutionGroup="ifc:IfcAddress" nillable="true"/>
<xs:complexType name="IfcPostalAddress">
   <xs:complexContent>
      <xs:extension base="ifc:IfcAddress">
         <xs:sequence>
            <xs:element name="AddressLines" nillable="true" minOccurs="0">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="ifc:IfcLabel-wrapper" maxOccurs="unbounded"/>
                  </xs:sequence>
                  <xs:attribute ref="ifc:itemType" fixed="ifc:IfcLabel-wrapper"/>
                  <xs:attribute ref="ifc:cType" fixed="list"/>
                  <xs:attribute ref="ifc:arraySize" use="optional"/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
         ..
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

IFC File

```
#74= IFCPOSTALADDRESS(.OFFICE.,'AEC3 Germany',$,$,
                      ('AEC3 Deutschland GmbH', 'Wendl-Dietrich-Str. 16'),
                      $,'Muenchen',$,'80634','Germany');
```

ifcXML File

```
<IfcPostalAddress id="i74" Purpose="office" Description="AEC3 Germany"
   Town="Muenchen" PostalCode="80634" Country="Germany">
   <AddressLines>
      <IfcLabel-wrapper>AEC3 Deutschland GmbH</IfcLabel-wrapper>
      <IfcLabel-wrapper>Wendl-Dietrich-Str. 16</IfcLabel-wrapper>
   </AddressLines>
</IfcPostalAddress>
```

**Table 8: Example of aggregated STRING data type that can contain space characters**

Manual override of the default configuration in ifcXML4

This default behaviour can be overridden on a per attribute basis for those aggregates of fundamental datatype being STRING if the XML white space delimited list does not change the semantic of the original attribute value or of fundamental datatype being BINARY whose values will be octet-strings (multiples of 8 bits). It is then translated into an XML attribute having the name of the EXPRESS attribute and being a white space delimited list.

Configuration for ifcXML4:

Stated for all such EXPRESS attributes individually. One configuration setting is required for each attribute. It overrides the general setting of *tagless* in <option> scope. This <attribute> configuration setting has to be done manually.

Manual configuration option for ifcXML4:

```
<cnf:entity select="name of entity">
  <cnf:attribute select="name of attribute" tagless="true" />
</cnf:entity>
```

Example:

```
<cnf:entity select="IfcTelecomAddress">
  <cnf:attribute select="TelephoneNumbers" tagless="true" />
</cnf:entity>
```

EXPRESS (skipping other attributes and constraints)

```
ENTITY IfcTelecomAddress
 SUBTYPE OF (IfcAddress);
   TelephoneNumbers : OPTIONAL LIST [1:?] OF IfcLabel;
   ..
END_ENTITY;
```

XSD

```
<xs:element name="IfcTelecomAddress" type="ifc:IfcTelecomAddress"
substitutionGroup="ifc:IfcAddress" nillable="true"/>
<xs:complexType name="IfcTelecomAddress">
   <xs:complexContent>
      <xs:extension base="ifc:IfcAddress">
         <xs:attribute name="TelephoneNumbers" use="optional">
            <xs:simpleType>
               <xs:restriction>
                  <xs:simpleType>
                     <xs:list itemType="ifc:IfcLabel"/>
                  </xs:simpleType>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
         ..
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

IFC File

```
#75= IFCTELECOMADDRESS(.OFFICE.,'AEC3 Germany',$,
       ('+49-89-18703223', '+49-171-2628789'),('+49-89-18703224'),$,
       ('tl@aec3.de', 'tl@aec3.com'),'www.aec3.com',$);
```

ifcXML File

```
<IfcTelecomAddress id="i75" Purpose="office" Description="AEC3 Germany"
   TelephoneNumbers="+49-89-18703223 +49-171-2628789"
   FacsimileNumbers="+49-89-18703224"
   ElectronicMailAddresses="tl@aec3.de tl@aec3.com"
   WWWHomePageURL="www.aec3.com"/>
```

**Table 9: Example of aggregated STRING data type that shall not contain space characters**

#### 2.5.2.2 ENTITY and SELECT data type

All EXPRESS attributes of a given ENTITY that are aggregates and have a fundamental data type being a select type or an entity type are translated into an XML element within the XML element corresponding to the given ENTITY holding the aggregation and a list of additional inline XML elements for the members of that aggregate. The name of the element holding the aggregation is the EXPRESS attribute name. The names of inline elements are the EXPRESS entity names of the instantiations.

Configuration:

Stated for all such EXPRESS attributes within the global <option> scope.

```
<cnf:option entity-attribute="double-tag" />
```

EXPRESS (skipping constraints)

```
ENTITY IfcRelAggregates
 SUBTYPE OF (IfcRelDecomposes);
   RelatingObject : IfcObjectDefinition;
   RelatedObjects : SET [1:?] OF IfcObjectDefinition;
END_ENTITY;
```

XSD

```
<xs:element name="IfcRelAggregates" type="ifc:IfcRelAggregates"
   substitutionGroup="ifc:IfcRelDecomposes" nillable="true"/>
<xs:complexType name="IfcRelAggregates">
   <xs:complexContent>
      <xs:extension base="ifc:IfcRelDecomposes">
         <xs:sequence>
            <xs:element name="RelatingObject" type="ifc:IfcObjectDefinition"
               nillable="true"/>
            <xs:element name="RelatedObjects">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="ifc:IfcObjectDefinition"
                        maxOccurs="unbounded"/>
                  </xs:sequence>
                  <xs:attribute ref="ifc:itemType" fixed="ifc:IfcObjectDefinition"/>
                  <xs:attribute ref="ifc:cType" fixed="set"/>
                  <xs:attribute ref="ifc:arraySize" use="optional"/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

IFC File

```
#8= IFCRELAGGREGATES('2YBqaV_8L15eWJ9DA1sGmT',$,$,$,#1895,(#1928));
```

ifcXML File

```
<IfcRelAggregates GlobalId="2YBqaV_8L15eWJ9DA1sGmT">
   <RelatingObject xsi:nil="true" ref="i1895" xsi:type="IfcProject"/>
   <RelatedObjects>
      <IfcBuilding xsi:nil="true" ref="i1928"/>
   </RelatedObjects>
</IfcRelAggregates>
```

Table 10: Example of aggregated ENTITY data type

### 2.5.3 Multi-dimensional aggregated EXPRESS attributes

The conversion is determined by the *exp-attribute* configuration (*concrete-attribute*, *entity-attribute* in <option> scope), the *tagless* configuration and the *flatten* configuration attribute.

#### 2.5.3.1 Simple and Enumerated data type

All EXPRESS attributes of a given ENTITY that are aggregates of aggregates (multi-dimensional aggregates) and have a fundamental data type being a simple type of INTEGER, REAL, NUMBER, BOOLEAN or LOGICAL or an enumerated type, and are "rectangular" are translated into an XML

attribute of the XML element corresponding to the given ENTITY. The value of that XML element is a space delimited list of such simple values or enumerators.

NOTE "Rectangular" means that all nested aggregates are of fixed size with min=max. This is not required for the first aggregate, which can be of an unknown size like [min=0: max=?].

Configuration for "rectangular" nested aggregations:

Stated for all such EXPRESS attributes individually. One configuration setting is required for each attribute. It overrides the general setting of *tagless* and *concrete-attribute* in <option> scope. This <attribute> configuration setting has to be done manually.

Manual configuration option for ifcXML4:

```
<cnf:entity select="name of entity">
  <cnf:attribute select="name of attribute" tagless="true"
   exp-attribute="attribute-content"/>
</cnf:entity>
```

Example:

```
<cnf:entity select="IfcCartesianPointList3D">
  <cnf:attribute select="CoordList" tagless="true"
   exp-attribute="attribute-content"/>
</cnf:entity>
```

NOTE flatten="true" is the default setting and therefore also applies if flatten is not stated within the <attribute> scope.

EXPRESS (definition of a "rectangular" nested aggregation)

```
ENTITY IfcCartesianPointList3D
 SUBTYPE OF (IfcCartesianPointList);
   CoordList : LIST [1:?] OF LIST [3:3] OF IfcLengthMeasure;
END_ENTITY;
```

XSD

```
<xs:element name="IfcCartesianPointList3D" type="ifc:IfcCartesianPointList3D"
   substitutionGroup="ifc: IfcCartesianPointList" nillable="true"/>
<xs:complexType name="IfcCartesianPointList3D">
   <xs:complexContent>
      <xs:extension base="ifc: IfcCartesianPointList">
         <xs:attribute name="CoordList" use="optional">
            <xs:simpleType>
               <xs:restriction>
                  <xs:simpleType>
                     <xs:list itemType="ifc:IfcLengthMeasure"/>
                  </xs:simpleType>
                  <xs:minLength value="3"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

IFC File

```
#1022= IFCCARTESIANPOINTLIST3D(((-500.,-500.,0.),(500.,-500.,0.),(500.,500.,0.),
   (-500.,500.,0.), (-500.,-500.,2000.),(500.,-500.,2000.),(500.,500.,2000.),
   (-500.,500.,2000.)));
```

ifcXML File

```
<IfcCartesianPointList3D id="i1976" CoordList="-500. -500. 0. 500. -500. 0. 500.
   500. 0. -500. 500. 0. -500. -500. 2000. 500. -500. 2000. 500. 500. 2000. -500.
   500. 2000."/>
```

**Table 11: Example of nested aggregations translated into XML attributes as used for "rectangular" nested aggregations of INTEGER, REAL, NUMBER, BOOLEAN, LOGICAL or ENUMERATION data type**

Configuration for "non-rectangular" nested aggregations:

All EXPRESS attributes of a given ENTITY that are "non-rectangular" aggregates of aggregates (multi-dimensional aggregates) are treated as a sequence of inline XML elements having the name of the EXPRESS defined type, or the generated name for an EXPRESS simple data type. The outline element grouping the aggregate has the name of the EXPRESS attribute. It is important to insert the pos attribute denoting the position of the element within the original multi-dimensioning structure.

NOTE These names get the suffix "-wrapper". The outline element grouping the aggregate has the name of the EXPRESS attribute.

NOTE The index of the 'pos' attribute for the first LIST element is 1.

The configuration is stated for all such EXPRESS attributes individually. One configuration setting is required for each attribute. It overrides the general setting of *tagless* and *concrete-attribute* in <option> scope. This <attribute> configuration setting has to be done manually.

Manual configuration option for ifcXML4:

```
<cnf:entity select="name of entity">
  <cnf:attribute select="name of attribute" tagless="false"/>
</cnf:entity>
```

Example:

```
<cnf:entity select="IfcStructuralLoadConfiguration">
   <cnf:attribute select="Locations" tagless="false"/>
</cnf:entity>
```

NOTE flatten="true" is the default setting and therefore also applies if flatten is not stated within the <attribute> scope.

EXPRESS (definition of a "non-rectangular" nested aggregation, without constraints)

```
ENTITY IfcStructuralLoadConfiguration
 SUBTYPE OF (IfcStructuralLoad);
   Values : LIST [1:?] OF IfcStructuralLoadOrResult;
   Locations : OPTIONAL LIST [1:?] OF UNIQUE LIST [1:2] OF IfcLengthMeasure;
END_ENTITY;
```

XSD (with tagless="false" for Locations attribute)

```
<xs:element name="IfcStructuralLoadConfiguration"
   type="ifc:IfcStructuralLoadConfiguration"
   substitutionGroup="ifc:IfcStructuralLoad" nillable="true"/>
<xs:complexType name="IfcStructuralLoadConfiguration">
   <xs:complexContent>
      <xs:extension base="ifc:IfcStructuralLoad">
         <xs:sequence>
            ..
            <xs:element name="Locations" nillable="true" minOccurs="0">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="ifc:IfcLengthMeasure-wrapper" minOccurs="1"
                        maxOccurs="unbounded"/>
                  </xs:sequence>
                  <xs:attribute ref="ifc:itemType"
                     fixed="ifc:IfcLengthMeasure-wrapper"/>
                  <xs:attribute ref="ifc:cType" fixed="list-unique list"/>
                  <xs:attribute ref="ifc:arraySize" use="optional"/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

IFC File

```
#721= IFCSTRUCTURALLOADCONFIGURATION($,(#701,#702,#703),((0.0),(3.5),(7.0)));
#701= IFCSTRUCTURALLOADSINGLEFORCE($, 10., $, $, $, $, $);
#702= IFCSTRUCTURALLOADSINGLEFORCE($, 25., $, $, $, $, $);
#703= IFCSTRUCTURALLOADSINGLEFORCE($, 50., $, $, $, $, $);
```

ifcXML File

```
<IfcStructuralLoadConfiguration id="i721">
   <Values>
      <IfcStructuralLoadSingleForce id="i701" ForceX="10."/>
      <IfcStructuralLoadSingleForce id="i702" ForceX="25."/>
      <IfcStructuralLoadSingleForce id="i703" ForceX="50."/>
   </Values>
   <Locations>
      <IfcLengthMeasure-wrapper pos="1 1">0.</IfcLengthMeasure-wrapper>
      <IfcLengthMeasure-wrapper pos="2 1">3.5</IfcLengthMeasure-wrapper>
      <IfcLengthMeasure-wrapper pos="3 1">7.</IfcLengthMeasure-wrapper>
   </Locations>
</IfcStructuralLoadConfiguration>
```

**Table 12: Example of nested aggregations that require a sequence of inline elements as for instance needed for "non-rectangular" nested aggregations**

#### 2.5.3.2 ENTITY and SELECT data type

Similar to "non-rectangular" nested aggregations of simple or enumerated type all EXPRESS attributes of a given ENTITY that are aggregates of aggregates (multi-dimensional aggregates) and having a fundamental data type being a SELECT type or an ENTITY type are translated into an XML element within the XML element corresponding to the given ENTITY holding the aggregation and a list of additional inline XML elements for the members of that aggregate. The name of the element holding the aggregation is the EXPRESS attribute name.

The names of inline elements are the EXPRESS entity names of the instantiations. It is important to insert the pos attribute denoting the position of the element within the original multi-dimensioning structure.

Configuration:

Stated for all such EXPRESS attributes individually. One configuration setting is required for each attribute. It overrides the general setting of *tagless* and *concrete-attribute* in <option> scope. This <attribute> configuration setting shall be done manually.

Manual configuration option for ifcXML4:

```
<cnf:entity select="name of entity">
  <cnf:attribute select="name of attribute" tagless="false"/>
</cnf:entity>
```

Example:

```
<cnf:entity select="IfcBSplineSurface">
   <cnf:attribute select="ControlPointsList" tagless="false"/>
</cnf:entity>
```

NOTE flatten="true" is the default setting and therefore also applies if flatten is not stated within the <attribute> scope.

EXPRESS (definition of nested aggregation of ENTITY, without constraints and other attribute definitions)

```
ENTITY IfcBSplineSurface
 ABSTRACT SUPERTYPE OF (ONEOF
   (IfcBSplineSurfaceWithKnots))
 SUBTYPE OF (IfcBoundedSurface);
   ..
   ControlPointsList : LIST [2:?] OF LIST [2:?] OF IfcCartesianPoint;
   ..
END_ENTITY;
```

XSD

```
<xs:element name="IfcBSplineSurface" type="ifc:IfcBSplineSurface" abstract="true"
   substitutionGroup="ifc:IfcBoundedSurface" nillable="true"/>
<xs:complexType name="IfcBSplineSurface" abstract="true">
   <xs:complexContent>
      <xs:extension base="ifc:IfcBoundedSurface">
         <xs:sequence>
            <xs:element name="ControlPointsList">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="ifc:IfcCartesianPoint"
                        minOccurs="4" maxOccurs="unbounded"/>
                  </xs:sequence>
                  <xs:attribute ref="ifc:itemType" fixed="ifc:IfcCartesianPoint"/>
                  <xs:attribute ref="ifc:cType" fixed="list list"/>
                  <xs:attribute ref="ifc:arraySize" use="optional"/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
         ..
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

IFC File (instantiation of non-abstract subtype *IfcBSplineSurfaceWithKnots*, incomplete example)

```
#87= IFCBSPLINESURFACEWITHKNOTS(3, 3,
       ((#1,#2,#3,#4,#6),(#11,#12,#13,#14,#15,#16),(#21,#22,#23,#24,#25,#26), ..),
       .UNSPECIFIED., .F., .F., .F., ... , .UNSPECIFIED.);
```

ifcXML File (incomplete example with mix of “by-reference” and “by-value” definition)

```
<IfcBSplineSurfaceWithKnots id="i89" UDegree="3" VDegree="3"
  SurfaceForm="unspecified" ...>
    <ControlPointsList>
      <IfcCartesianPoint xsi:nil="true" ref="i270" pos="1 1"/>
      <IfcCartesianPoint xsi:nil="true" ref="i271" pos="1 2"/>
      <IfcCartesianPoint xsi:nil="true" ref="i272" pos="1 3"/>
      <IfcCartesianPoint xsi:nil="true" ref="i273" pos="1 4"/>
      <IfcCartesianPoint id="i290" Coordinates="1. 0. -1." pos="1 5"/>
      ..
    </ControlPointsList>
    ..
</IfcBSplineSurfaceWithKnots>
```

**Table 13: Example of nested aggregation of ENTITY type**

## 2.6 Handling of INVERSE attributes

The EXPRESS definition of IFC makes use of inverse attribute definitions that enable using both directions of a reference between entity instances. Whereas inverse attributes improve accessibility of information when working with model data they are not exported into an ifc file. This is because inverse attributes are redundant information that can be re-constructed when importing IFC data. Then the inverse attribute enables a named path to traverse in the opposite direction of the relationship between two entities. The XML schema language does not support definition of inverse attributes in the same way. Exporting both, the direct and the inverse attribute explicitly within the XML data file would otherwise increase the file size and impose the danger of having redundant and potentially conflicting data. Therefore it was decided to skip inverse attribute definitions in the ifcXML XSD. But there are exceptions where inverse attributes are a better choice to be available in an ifcXML file instead of the direct attributes. For those exceptions the following configuration settings are used.

Configuration for the use of INVERSE instead of explicit attributes:

The following relationships are inverted, which means that the explicit attribute is removed by the keep="false" setting and the inverse attribute is added by the <inverse> configuration element.

NOTE The configuration of inverse attributes (“attribute-tag” or “double-tag”) depends on the kind of inverse attributes.

Manual configuration option for ifcXML4:

```
<cnf:entity select="name of entity">
  <cnf:attribute select="name of explicit attribute" keep="false"/>
</cnf:entity>
<cnf:entity select="name of entity">
  <cnf:inverse select="name of inverse" exp-attribute="config setting for inverse"/>
</cnf:entity>
```

Example:

```
<cnf:entity select="IfcGeometricRepresentationContext">
   <cnf:inverse select="HasSubContexts" exp-attribute="double-tag"/>
</cnf:entity>
<cnf:entity select="IfcGeometricRepresentationSubContext">
   <cnf:attribute select="ParentContext" keep="false"/>  </cnf:entity>
```

EXPRESS (definition for using inverse instead of explicit attributes, without constraints and other attribute definitions)

```
ENTITY IfcGeometricRepresentationContext
 SUBTYPE OF (IfcRepresentationContext);
   ..
INVERSE
   HasSubContexts : SET [0:?] OF IfcGeometricRepresentationSubContext
                                 FOR ParentContext;
END_ENTITY;

ENTITY IfcGeometricRepresentationSubContext
 SUBTYPE OF (IfcGeometricRepresentationContext);
   ParentContext : IfcGeometricRepresentationContext;
   ..
END_ENTITY;
```

XSD

```
<xs:element name="IfcGeometricRepresentationContext"
   type="ifc:IfcGeometricRepresentationContext"
   substitutionGroup="ifc:IfcRepresentationContext" nillable="true"/>
<xs:complexType name="IfcGeometricRepresentationContext">
   <xs:complexContent>
      <xs:extension base="ifc:IfcRepresentationContext">
         <xs:sequence>
            ..
            <xs:element name="HasSubContexts" nillable="true" minOccurs="0">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="ifc:IfcGeometricRepresentationSubContext"
                        minOccurs="0" maxOccurs="unbounded"/>
                  </xs:sequence>
                  <xs:attribute ref="ifc:itemType"
                     fixed="ifc:IfcGeometricRepresentationSubContext"/>
                  <xs:attribute ref="ifc:cType" fixed="set"/>
                  <xs:attribute ref="ifc:arraySize" use="optional"/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
         ..
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

IFC File

```
#53= IFCGEOMETRICREPRESENTATIONCONTEXT($,'Model',3,1.0000000E-5,#46,#$);
#46= IFCAXIS2PLACEMENT3D(#42,$,$);
#61= IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Body','Model',*,*,*,*,#53,
                                          $,.MODEL_VIEW.,$);
```

ifcXML File

```
<IfcGeometricRepresentationContext id="i1914" ContextType="Model"
CoordinateSpaceDimension="3" Precision="1.000000000E-5">
   <IfcAxis2Placement3D id="i1918">
      <Location xsi:nil="true" ref="i1942" xsi:type="IfcCartesianPoint"/>
   </IfcAxis2Placement3D>
   <HasSubContexts>
      <IfcGeometricRepresentationSubContext id="i1917" ContextIdentifier="Body"
         ContextType="Model" TargetView="model_view"/>
   </HasSubContexts>
</IfcGeometricRepresentationContext>
```

**Table 14: Example of inverse relationships**

NOTE The re-declared as derived attributes are masked as * in the ifc file, those attributes are removed in the ifcXML schema by using the *complexContent* by *restriction*. Therefore they do not appear in the ifcXML file for *IfcGeometricRepresentationSubContext*.

# 3 Conversion rules for ifcXML4 for specific MVD's

Beside the ifcXML XSD for the entire schema of IFC4 (ifcXML4) there will be use case dependent configurations that further optimizes XML schema definition to fulfil specific exchange requirements. These configuration settings apply to the sub schemas generated for a specific MVD. Additionally to the configuration settings described above the followings setting is used:

- keep: to disallow specific attributes, in particular the *IfcStrippedOptional* type definition that is used in EXPRESS sub schema definitions as a marker for not used attributes.

Further details about these configuration settings will be provided for each MVD.

# Appendix – ifcXML4 configuration settings

The configuration file is created in a two-step process. The first step is based on a script that creates a baseline configuration, which in a second step is manually extended to fulfil all needs as described above.

NOTE In many cases the global or default setting is explicitly redefined although not necessary. This is done to clearly state the used configuration setting and thus to avoid misinterpretation of default settings.

## Automatically added attribute definitions

The baseline configuration is automatically created for non-aggregated attributes with a fundamental data type being an ENTITY. Currently, the following rule applies:

- An “attribute-tag” setting is used for all ENTITY data types.

## Manually added attribute definitions for ifcXML4 (entire schema)

The following settings are added on a case by case basis when generating the XML schema from the entire IFC4 schema.

### “tagless” settings for non-nested aggregations

STRING:

The default setting of STRING aggregations is changed to tagless = “true” (i.e. list of string values without a space character) for the following elements:

```
<cnf:entity select="IfcPerson">
  <cnf:attribute select="MiddleNames" tagless="true"/>
  <cnf:attribute select="PrefixTitles" tagless="true"/>
  <cnf:attribute select="SuffixTitles" tagless="true"/>
</cnf:entity>
<cnf:entity select="IfcTelecomAddress">
  <cnf:attribute select="TelephoneNumbers" tagless="true"/>
  <cnf:attribute select="FacsimileNumbers" tagless="true"/>
  <cnf:attribute select="ElectronicMailAddresses" tagless="true"/>
  <cnf:attribute select="MessagingIDs" tagless="true"/>
</cnf:entity>
<cnf:entity select="IfcSurfaceTexture">
  <cnf:attribute select="Parameter" tagless="true"/>
</cnf:entity>
```

The following STRING aggregations can contain space characters and therefore must use the setting tagless=”false”. Although similar to the global configuration setting it was decided to add an individual configuration:

```
<cnf:entity select="IfcClassification">
  <cnf:attribute select="ReferenceTokens" tagless="false"
    exp-attribute="double-tag"/>
</cnf:entity>
<cnf:entity select="IfcPostalAddress">
  <cnf:attribute select="AddressLines" tagless="false"
    exp-attribute="double-tag"/>
</cnf:entity>
```

```
<cnf:entity select="IfcTextStyleFontModel">
  <cnf:attribute select="FontFamily" tagless="false"
    exp-attribute="double-tag"/>
</cnf:entity>
```

BINARY:

The default setting of BINARY aggregations (i.e. list of BINARY values being octet-strings (multiples of 8 bits) is changed to tagless = “true” for the following elements:

```
<cnf:entity select="IfcPixelTexture">
  <cnf:attribute select="Pixel" tagless="true" exp-attribute="attribute-content"/>
</cnf:entity>
```

## “tagless” settings for nested aggregations

The following "rectangular" nested aggregations of the fundamental data type being a simple type use the tagless="true" setting:

```
<cnf:entity select="IfcCartesianPointList3D">
  <cnf:attribute select="CoordList" tagless="true"
    exp-attribute="attribute-content"/>
</cnf:entity>
<cnf:entity select="IfcColourRgbList">
  <cnf:attribute select="ColourList" tagless="true"
    exp-attribute="attribute-content"/>
</cnf:entity>
<cnf:entity select="IfcIndexedTriangleTextureMap">
  <cnf:attribute select="TexCoordIndex" tagless="true"
    exp-attribute="attribute-content"/>
</cnf:entity>
<cnf:entity select="IfcTextureVertexList">
  <cnf:attribute select="TexCoordsList" tagless="true"
    exp-attribute="attribute-content"/>
</cnf:entity>
<cnf:entity select="IfcTriangulatedFaceSet">
  <cnf:attribute select="CoordIndex" tagless="true"
    exp-attribute="attribute-content"/>
  <cnf:attribute select="NormalIndex" tagless="true"
    exp-attribute="attribute-content"/>
</cnf:entity>
```

The following "non-rectangular" nested aggregations use the tagless="false" setting:

```
<cnf:entity select="IfcBSplineSurface">
  <cnf:attribute select="ControlPointsList" tagless="false"/>
</cnf:entity>
<cnf:entity select="IfcRationalBSplineSurfaceWithKnots">
  <cnf:attribute select="WeightsData" tagless="false"/>
</cnf:entity>
<cnf:entity select="IfcStructuralLoadConfiguration">
  <cnf:attribute select="Locations" tagless="false"/>
</cnf:entity>
```

## "attribute-tag" for BINARY attributes of unknown bit-size

BINARY attributes of an unknown size or being a non-octet string require information about the so called extra-bits. This information cannot be added to XML attributes and thus need to be mapped to a sequence of inline elements using the “attribute-tag” setting. This setting is necessary for:

```
<cnf:entity select="IfcBlobTexture">
  <cnf:attribute select="RasterCode" exp-attribute="attribute-tag"/>
</cnf:entity>
```

## Use of inverse instead of explicit attributes

The following relationships are inverted, which means that the explicit attribute is removed by the keep="false" setting and the inverse attribute is added by the <inverse> configuration element.

```
<cnf:entity select="IfcGeometricRepresentationContext">
  <cnf:attribute select="WorldCoordinateSystem" exp-attribute="double-tag"/>
  <cnf:attribute select="TrueNorth" exp-attribute="attribute-tag"/>
  <cnf:inverse select="HasSubContexts" exp-attribute="double-tag"/>
</cnf:entity>
<cnf:entity select="IfcGeometricRepresentationSubContext">
  <cnf:attribute select="ParentContext" keep="false"/>
</cnf:entity>
<cnf:entity select="IfcTessellatedFaceSet">
  <cnf:attribute select="Coordinates" exp-attribute="attribute-tag"/>
  <cnf:attribute select="Normals" exp-attribute="attribute-tag"/>
  <cnf:inverse select="HasColours" exp-attribute="attribute-tag"/>
  <cnf:inverse select="HasTextures" exp-attribute="double-tag"/>
</cnf:entity>
<cnf:entity select="IfcIndexedColourMap">
  <cnf:attribute select="MappedTo" keep="false"/>
  <cnf:attribute select="Overrides" exp-attribute="attribute-tag"/>
  <cnf:attribute select="Colours" exp-attribute="attribute-tag"/>
</cnf:entity>
<cnf:entity select="IfcIndexedTextureMap">
  <cnf:attribute select="MappedTo" keep="false"/>
  <cnf:attribute select="TexCoords" exp-attribute="attribute-tag"/>
</cnf:entity>
<cnf:entity select="IfcMaterial">
  <cnf:inverse select="HasRepresentation" exp-attribute="attribute-tag"/>
</cnf:entity>
<cnf:entity select="IfcMaterialDefinitionRepresentation">
  <cnf:attribute select="RepresentedMaterial" keep="false"/>
</cnf:entity>
<cnf:entity select="IfcMaterialDefinition">
  <cnf:inverse select="HasProperties" exp-attribute="double-tag"/>
</cnf:entity>
<cnf:entity select="IfcMaterialProperties">
  <cnf:attribute select="Material" keep="false"/>
</cnf:entity>
<cnf:entity select="IfcProfileDef">
  <cnf:inverse select="HasProperties" exp-attribute="double-tag"/>
</cnf:entity>
<cnf:entity select="IfcProfileProperties">
  <cnf:attribute select="ProfileDefinition" keep="false"/>
</cnf:entity>
<cnf:entity select="IfcProductDefinitionShape">
```

```
    <cnf:inverse select="HasShapeAspects" exp-attribute="double-tag"/>
  </cnf:entity>
  <cnf:entity select="IfcShapeAspect">
    <cnf:attribute select="PartOfProductDefinitionShape" keep="false"/>
  </cnf:entity>
  <cnf:entity select="IfcRepresentationItem">
    <cnf:inverse select="StyledByItem" exp-attribute="attribute-tag"/>
  </cnf:entity>
  <cnf:entity select="IfcStyledItem">
    <cnf:attribute select="Item" keep="false"/>
  </cnf:entity>
```

## Acknowledgment

This work would not have been able without the Norwegian Building Authority (DiBK) funding work for buildingSMART International and the research project Mefisto (http://www.mefisto-bau.de/objective/lang/en), partially funded by the German Ministry of Education and Research, funding work for AEC3 Deutschland GmbH. We would like to thank both parties for supporting this work.