# Detailed Assessment of IFC 4.3 Experimental

URL Specification: <https://standards.buildingsmart.org/IFC/DEV/IFC4_3/RC1/HTML/>

The IFC data model provides a rich but highly complex way of representing physical objects, primarily focused on buildings and structures. Compared to the UML model backing NAS, IFC uses a different modeling language called EXPRESS.

There is a hierarchy of entity types that loosely match up to the OWT concept of Entity, and these can be mapped more or less directly. IFC entities have a number of unique direct attributes identified by name and sequence number. These attributes are inherited by derived entity types and tend to be fairly general purpose.

There is also a mechanism to nonintrusively add additional attributes organized as property sets (in the general case) or quantity sets linked in by entity relationships – specifically, the IsDefinedBy relationship. These are somewhat similar to “information” or “description” features in NAS/GGDM but are used much more widely in IFC. In some cases, these map directly to built-in attributes in OWT; in others it may make sense to treat them as either information entities in the same way, or to treat them as domains organizing attributes. The main difference between property sets and quantity sets is that quantity sets has a method of measurement and consists solely of measured numeric types, although property sets can also have measured numeric types.

Compared to GGDM/NAS/OWT, the IFC data model tends to much more heavily use both entity inheritance / subclassing and relationships to informational content rather than direct flat attribution. One very substantial data model mapping challenge is that many qualitative attributes (aka enumerations or terms) are unconstrainted natural language text in IFC – possibly in any human language or referencing any national standard - so unless other standards prescribe the term lists for these attributes, the mapping is going to be very imprecise.

On e other challenge in mapping property sets in particular is that the definitions are only available in German, French, and Japanese in the experiment 4.3 version. The release 4.1 version does have English definitions of equivalent properties in most (but not all) cases.

The various “IfcQuantity” subclasses should probably be mapped to NAS Quantity types to be cross-referenced to attributes and measures.

Property set attributes can have various broad types:

* Single (scalar) values
* Bounded (interval range) values
* Enumerated (term) values
* List values
* Table (key/value map) values – effectively metadata subattributes

These types can be text or they can be numeric with measures.

The relationship mechanism also describes a rich set of spatial relationships such as IsNestedBy, IsDecomposedBy, and so on.

# IfcKernel - Common Base Types

## IfcRoot

Every IFC definition – objects (entities), properties (attributes), and relationships derives from the Root type and inherit some common attributes from it.

Direct Attributes:

* GlobalId – 128-bit UUID uniquely identifying the object instance (mapped to UFI)
* OwnerHistory – one or more revisions describing revision history (TODO on mapping)
* Label – text name of the object instance (mapped to SNA)
* Description – exchange text for the object instance (mapped to MEM)

## IfcObjectDefinition

Abstract base class of IfcObject. No attributes, only defines valid sets of relationships. Good place to look for deciding what kinds of spatial relationships are generally applicable.

## IfcObject

Generalization: IfcObjectDefinition

All IFC objects consider IfcObject as a base type. It only has one main attribute and some relationships.

Direct Attributes:

* ObjectType – text label for object type, values subclass-dependent (mapped to F\_CODE)

## IfcContext

Abstract base class of IfcProject and IfcProjectLibrary. This basically defines a dataset that contains one ore more sites, facilities, etc.

We may want to consider mapping this to data source metadata in the future.

## IfcProduct

Generalization: IfcObject

The IfcProduct, despite the more specific sounding name, is a general base type for all objects with geometry and spatial location. The attributes actually describe geometric representation. Subclasses can constraint what types of geometry are actually valid such as point (center of gravity), box, etc.

Direct Attributes:

* ObjectPlacement – actually describes the coordinate system for the representation, which can either be an absolute location relative to the overall project world space, a location relative to another object, a location on a grid, or a location on a parametric linear curve
* Representation – describes the geometry as a set of shape aspects, which can either be discrete geometry (points, curves, surfaces, volumes, etc.) or topology (b-rep) – multiple different domains of shapes can be specified

## IfcSpatialElement

Generalization: IfcProduct

IfcSpatialElement is an abstract base class for both overall building structures and zones. It primarily adds relationships specific to other elements but also has a hook to link in systems (such as resource providers) that service the element.

Direct Attributes:

* LongName – augments Name attribute inherited from IfcObject (mapped to FNA)

## IfcSpatialStructuralElement

Generalization: IfcSpatialElement

IfcSpatialStructuralElement is an abstract base class for all overall structures (but not individual building component elements).

Direct Attributes:

* CompositionType – basically identifies whether it is standalone, an aggregation of other components, or a part of some other element; this does not map directly to OWT but is used as a conditional selector to map IFC types to OWT types

Property Sets:

* Pset\_PropertyAgreement – legal agreements on property ownership, use, or restrictinos

# IfcControlExtension

This set of definitions covers construction process controls like costs, permits, and work schedules. We can ignore this part for now.

# IfcProcessExtension

This set of definitions covers time-sequenced activities such as tasks, events, and work schedules. We can ignore this part for now.

# IfcProductExtension – Overall Building Types

This set of definitions describes environment-level objects such as buildings, facilities, sites, bridges, and so on. Most of these have existing GGDM counterparts but some do not.

## IfcFacility

Generalization: IfcSpatialStructuralElement

IfcFacility is an abstract base class for specific constructions including buildings, bridges, roads, marine facilities, tunnels, and so on. As such it does NOT map to the OWT/GGDM entity type of Facility necessarily – the mapping is both subclass and attribution dependent.

The Facility does not have any unique attributes or relationships and is only used for clarifying semantics of base and derived types.

## IfcBuilding

Generalization: IfcFacility

Overall building definition. In IFC it subclasses IfcFacility. Individual components are linked in by relationships. In most (but not all cases), floor levels defined by IfcBuildingStorey are actually the primary owner of individual components.

Per the documentation, this can actually represent one of three different things depending on the CompositionType attribute inherited from IfcFacility:

* COMPLEX: a building complex that maps to OWT Facility
* ELEMENT: a single building that maps to OWT Building (AL013)
* PARTIAL: a partial section of a building (no mapping?)

Direct Attributes:

* ElevationOfRefHeight – altitude above terrain of the 0.0 height level for measuring floor levels – “usually top of construction slab” (recommend adding to OWT – cross-check with NAS?)
* ElevationOfTerrain – minimum altitude of sloped terrain around the building (we can ignore this?)
* BuildingAddress – postal address of building (TODO figure out how to map this)

Property sets:

* Pset\_BuildingCommon – random grab bag of properties not belonging to other buckets
  + ConstructionMethod – mapped to
  + NumberofStoreys – mapped to
  + YearOfConstruction – mapped to
  + IsLandmarked – mapped to
* Pset\_BuildingUse – zoning and vacancy information
  + MarketCategory – overall use like residential/commercial/industrial
  + MarketSubcategory – specific types like single-family, multi-family, etc.
  + PlanningControlStatus – actual government zoning of property
* Pset\_BuildingUseAdjacent – overall context of zoning information in area
* Pset\_OutsideDesignCriteria – temperature, humidity, and wind extremes planned for in the design of the building (can skip for now)

Quantity sets:

* Qto\_BuildingBaseQuantities – geometry and similar properties
  + Height – maps to HGT
  + EavesHeight – height of eaves above terrain, maybe useful to add
  + FootprintArea – maps to ARA
  + Various other floor areas and volumes (probably not needed)

## IfcBuildingStorey

Generalization: IfcSpatialStructuralElement

The building storey describes a logical floor level – a horizontal spatial partition of a building.

Depending on the CompositionType, it can either represent a full single level (conventional) or a complex level composed of multiple partial levels (i.e. mezzanines, stages, catwalks, etc.) related by aggregation.

Building storey has no direct attributes.

Property sets:

* Pset\_BuildingStoreyCommon – general properties of the level
  + EntranceLevel – Boolean indicating whether it is a level that has an external entrance (recommend adding to OWT)
  + AboveGround – Boolean indicating whether the level is above the terrain surface (recommend adding to OWT or mapping if already available)
  + LoadBearingCapacity – maximum force level can hold – recommend adding to OWT
* Pset\_PropertyAgreement – legal ownership, use, and restrictions – same as for building
* Pset\_ThermalLoadAggregate – properties describing HVAC thermal load of level
* Pset\_ThermalLoadDesignCriteria – properties describing the HVAC thermal loads that the level was designed for

Quantity sets:

* Qto\_BuildingStoreyBase\_Quantities
  + GrossHeight – height from floor surface to next floor surface (map to HGT)
  + NetHeight – height from floor surface to ceiling surface (add to OWT either directly or separate ceiling thickness?)
  + Various properties for perimeter, surface, and volume

## IfcElement

Generalization: IfcElement

Abstract base type for a variety of objects for building interiors and civil infrastructure. By itself it doesn’t mean much semantically, but it is the attachment point for a lot of relationships including physical connections, openings, boundaries, coverings, and interference between things.

Direct attributes:

* Tag – local identifier such as serial number or feature index (consider mapping to FID?)

Property sets:

* Pset\_EnvironmentalImpactValues – various resource consumption metrics for energy, waer, hazardous, inert, radioactive, and non-hazardous waste, and various climate change impacts – some of these attributes seem useful for civil infrastructure and for CBRNE modeling
* Pset\_EnvironmentalImpactIndicators – threshold values for the environment impact values for them to be considered a problem
* Pset\_Condition – human readable assessment of condition on a particular date
* Pset\_ManufacturerOccurrence – serial number and manufacturing references to the specific part(s) instances used
* Pset\_ManufacturerTypeInformation – general description of the type of part(s) such as manufacturer, model reference, production year, place of assembly
* Pset\_ServiceLife – mean time between failure and service life duration – maybe useful for civil infrastructure
* Pset\_Warranty – warranty information

## IfcBuiltElement

Generalization: IfcElement

Abstract base type for individual building components.

It does not add any attributes or relationships so it’s just for semantic clarify of derived types.

## IfcFeatureElement

Generalization: IfcElement

Abstract base type for describing any entities that modify the appearance or structure of another built element such as openings, coverings, or decoration - this is feature in the interior decoration sense, not in the geospatial sense.

This entity does not define anything new and exists solely for semantic clarification.

## IfcFeatureSubtraction

Generalization: IfcFeatureElement

Abstract base type for describing entities that modify another element by removing geometry from it via Boolean subtractive geometry.

The entity only defines a new relationship for voiding another element.

## IfcOpeningElement

Generalization: IfcFeatureElementSubtraction

Type that represents any kind of void in another solid element (such as a wall or slab) – it can either be a full opening through the solid object serving as an aperture, or a recess within the object that does not fully penetrate it.

Direct Attributes

* PredefinedType – describes whether it’s a full opening or just a recess

Property sets

* Pset\_OpeningElementCommon
  + Status – can describe whether it is planned, currently exists, is slated for demolition, is temporary (difficulty: freeform natural language text)
  + Purpose – functional role such as human access, ventilation (difficulty: freeform natural language text)
  + FireExit – Boolean on whether it’s a fire exit (maybe useful?)
  + ProtectedOpening – Boolean on whether the door is fire-protected

Quantity Sets:

* Qto\_OpeningElementBaseQuantities
  + Width – maps to WID or LZN?
  + Height - maps to HGT
  + Depths – maps to THI or WID?

# IfcSharedBldgElements

## IfcWall

Generalization: IfcBuiltElement

Basic vertical partitions – can be load-bearing but not required to be. Can be simple or complex constructions (i.e. decomposed into panels, fasteners, studs, etc.)

Direct Attributes

* PredefinedType – various types which could be useful, some may induce different feature types in OWT
  + MOVABLE – folding walls etc. that are more furniture than walls
  + PARAPET – partial walls primarily used as safety edges
  + PARTITIONING – walls formed of layers like drywall typically used for interior partitions
  + PLUMBINGWALL – walls primarily for privacy in bathrooms
  + SHEAR – walls intended to handle shear loads
  + SOLIDWALL – walls consisting of solid construction typically load-bearing and fire safe
  + STANDARD – (unrelated?) describes geometry as being extruded linear
  + POLYGON – (unrelated?) describes geometry as being polygon footprint
  + RETAININGWALL – walls used to organize or partition soil layers
  + WAVEWALL – seawalls to protect against waves

## IfcDoor

Generalization: IfcBuiltElement

Doors are the actual panel that controls access through a wall opening. In addition to conventional doors, this also seems to cover other forms of entry control such as gates and barriers. In terms of relationships, doors can exist in one of three states:

* Fills the void for a wall opening (IfcOpeningElement)
* Is a component of a wall (IfcCurtainWall)
* Is a standalone object not part of anything else (door blown off the hinges?)

Direct Attributes

* OverallWidth – maps to WID or LZN?
* OverallHeight – maps to HGT
* PredefinedType – enumeration, different types may map to different feature codes
  + DOOR – maps to Door (UF003)
  + GATE – maps to Gate (AP040)
  + TRAPDOOR – maps to Trapdoor
  + BOOM\_BARRIER – think GGDM has this?
  + TURNSTILE – probably should be added to OWT as a feature type

## IfcSlab

Generalization: IfcBuiltElement

Slabs describe the actual physical horizontal surface forming a floor or roof covering. It is only the structure of the item and does not include cosmetic items like floor coverings (carpets, tiles) or ceiling coverings (tiles, etc.).

Depending on the ObjectType value inherited all the way from IfObject, the slab can represent a pretty wide variety of GGDM feature types:

* FLOOR – maps to floor slabs and bridge decks
* ROOF – maps to the roof slab either flat or sloped (complex roofs are multiple slabs?)
* LANDING – floor slab for a stair landing in a complex stair set
* BASESLAB – floor slab anchored to the ground and thus is part of the foundation
* APPROACH\_SLAB – bridge component slab on approach from embankment
* PAVING – road surface component slab
* WEARING – road surface, but on a bridge span
* SIDEWALK – slabs forming a sidewalk
* TRACK\_SLAB – slab that is the base of a railway or similar track transport