BIM/IFC models for QTO

Below is a concise, **Python-first** recipe that shows exactly how to plug an **IFC model** (provided with the QRF) into your **Quantity-Take-Off (QTO)** pipeline. It follows the **external-mode** pattern described in the BIM literature: the IFC file is **only a data source**; all logic runs in Python.

---

1. Install the core IFC library

```
pip install ifcopenshell pandas openpyxl
```

2. Load the IFC file and list every structural element

```
import ifcopenshell, ifcopenshell.util.element as util
from pathlib import Path

model = ifcopenshell.open(Path("rfx_structural.ifc"))

# Example: grab every IfcBeam, IfcColumn, IfcMember, etc.
elements = (model.by_type("IfcBeam") +
            model.by_type("IfcColumn") +
            model.by_type("IfcMember"))
```

3. Extract the quantities you need

```
rows = []
for e in elements:
    # 1. Identity
    name   = e.Name or e.GlobalId
    ifc_ent= e.is_a()
```

```
    # 2. Geometry quantities (IfcElementQuantity)
    qs = util.get_psets(e).get("Pset_ElementQuantity", {})
    length_m = float(qs.get("Length", 0))
    weight_kg= float(qs.get("Weight", 0))     # if the modeller exported it
    area_m2  = float(qs.get("SurfaceArea", 0))

    # 3. Material grade (IfcMaterial)
    mat = util.get_material(e)
    grade = mat.Name if mat else "Unknown"

    rows.append({
        "Item"      : name,
        "Type"      : ifc_ent,
        "Material"  : grade,
        "Length_m"  : length_m,
        "Weight_kg" : weight_kg,
        "Area_m2"   : area_m2
    })
```

4. Build a Pandas DataFrame → instant QTO table

```
import pandas as pd

df = pd.DataFrame(rows)
# Aggregate identical sections
qto = (df
       .groupby(["Type", "Material"], as_index=False)
       .agg({"Length_m":"sum",
             "Weight_kg":"sum",
             "Area_m2":"sum"}))
```

5. Export to Excel (ready for BOQ merge)

```
qto.to_excel("IFC_QTO.xlsx", index=False)
```

6. Optional: validate IFC quality first

Use **BIMvision** or **Solibri Anywhere** (free viewers) to visually inspect the model and confirm that all **Pset_ElementQuantity** properties are populated.

7. Handling federated models (multiple IFC files)

If the QRF supplies **several partial IFC files**, merge them once:

- **BIMvision → IFC Merge plugin** (permanent merge), or
- **IfcOpenShell** (memory merge) if you want to stay in Python.

8. When IFC lacks quantities

If the IFC only has geometry, compute **volume/length/area** yourself:

```
import ifcopenshell.geom as geom
settings = geom.settings()
shape = geom.create_shape(settings, e)
volume = shape.geometry.volume
```

9. Keep the workflow **MVD-compliant**

The NBIMS QTO guide recommends exporting with the **CDB-2010 MVD** view (or later). Ask the designer to tick that option in Revit/ArchiCAD so that **Pset_ElementQuantity** and **IfcMaterial** are automatically embedded.

Summary of the integration pattern

Step	Tool / Library	Purpose
IFC ingestion	`IfcOpenShell`	Parse geometry & properties
Quantity extraction	`IfcElementQuantity` or auto-calc	Length, area, volume, weight
Data shaping	`Pandas`	Group, sum, clean
Output	`openpyxl`	Excel BOQ ready for pricing

Step	Tool / Library	Purpose
Visual QC	BIMvision / Solibri	Confirm model quality

This external-mode approach keeps your Python stack lightweight, avoids STAAD-Pro geometry duplication, and produces **code-compliant QTO tables** in minutes

---

Revision #1
Created 2025-07-18 01:59:43 UTC by Admin
Updated 2025-07-18 02:01:34 UTC by Admin