EnergyPlus - Moving from IDF to JDF (JSON)

Mark Adams	*Oak Ridge National Laboratory*
Jason Glazer	*GARD Analytics*
Michael J. Witte	*GARD Analytics*

Contribution Number	6169
Presentation Date	2017-08-07
Presentation Time	13:45

Outline

- Tutorial/Training Materials
- EnergyPlus IDF
- Why Change?
- EnergyPlus JDF (JSON)
- EnergyPlus JDD (JSON Schema)
- Modifying JDF
- IDF-Based Pre-processors
- Looking Forward

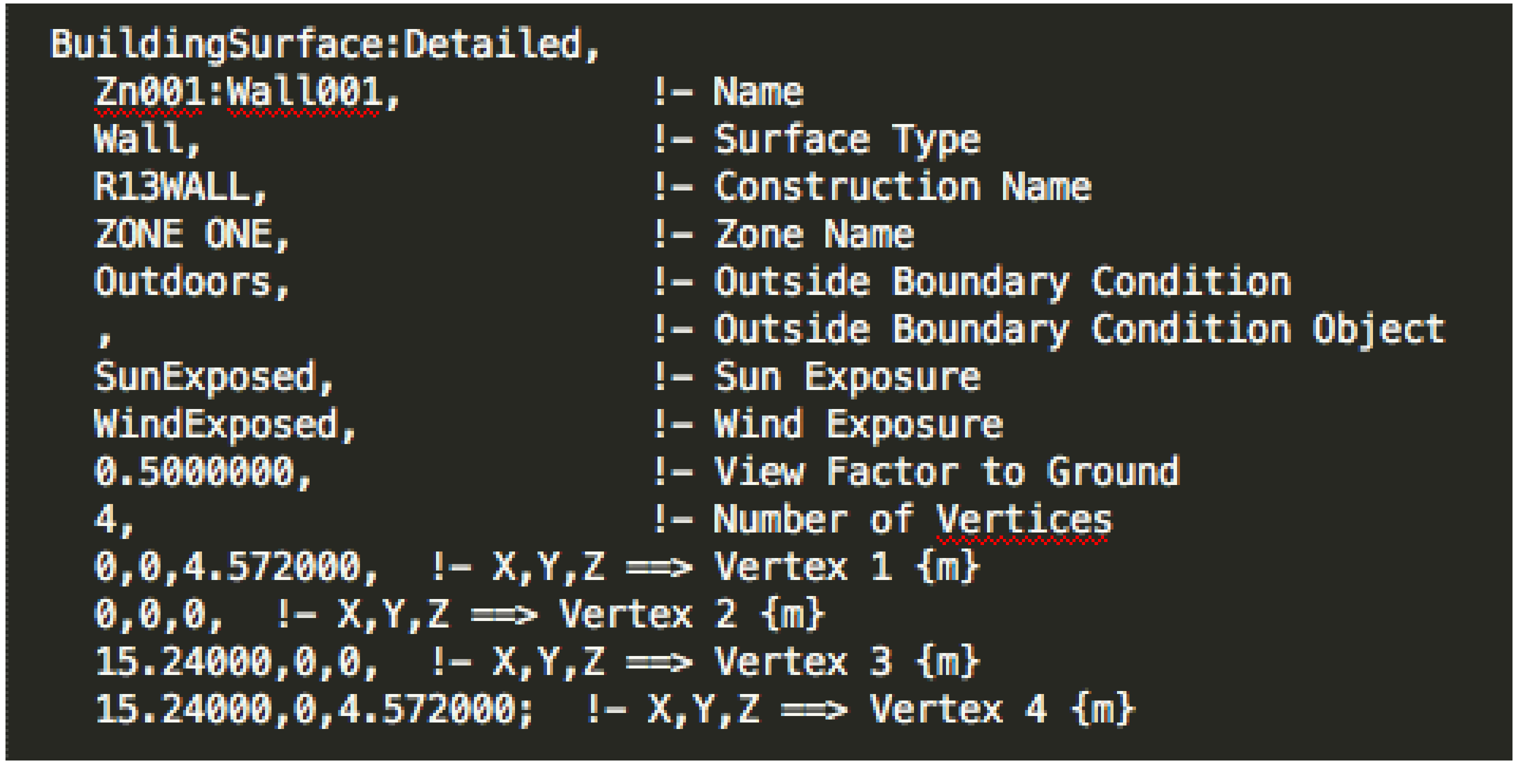
Tutorial/Training Materials

- https://github.com/ORNL-BTRIC/IBPSA-BuildingSim-2017-JSON

- Contains:
  - Example IDF
  - Example JDF
  - JSON-enabled EnergyPlus version
  - README with walk-through instructions
  - Python script to show validation outside E+

EnergyPlus IDF

- Comma separated variable (CSV)-like input



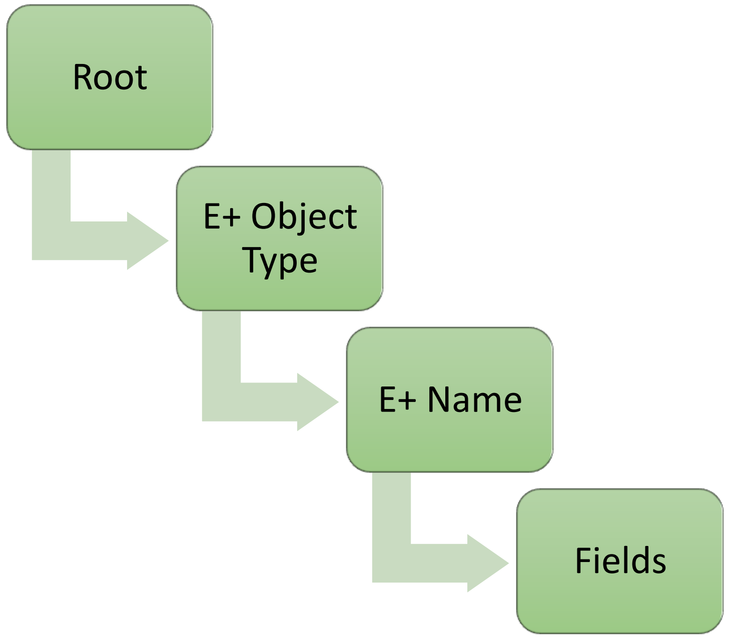
```
BuildingSurface:Detailed,
  Zn001:Wall001,            !- Name
  Wall,                     !- Surface Type
  R13WALL,                  !- Construction Name
  ZONE ONE,                 !- Zone Name
  Outdoors,                 !- Outside Boundary Condition
  ,                         !- Outside Boundary Condition Object
  SunExposed,               !- Sun Exposure
  WindExposed,              !- Wind Exposure
  0.5000000,                !- View Factor to Ground
  4,                        !- Number of Vertices
  0,0,4.572000,  !- X,Y,Z ==> Vertex 1 {m}
  0,0,0,  !- X,Y,Z ==> Vertex 2 {m}
  15.24000,0,0,  !- X,Y,Z ==> Vertex 3 {m}
  15.24000,0,4.572000;  !- X,Y,Z ==> Vertex 4 {m}
```

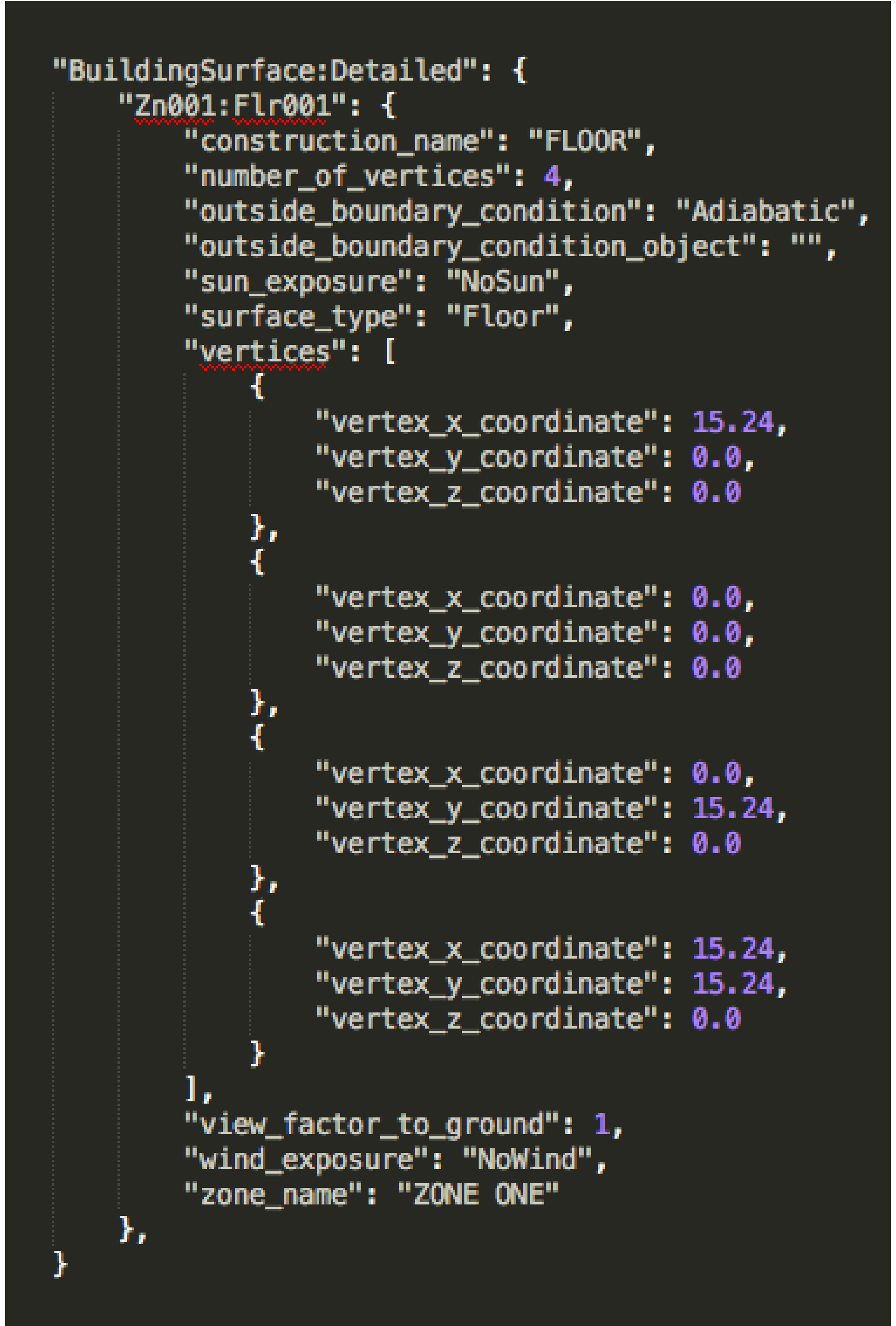
Why change?

- Difficult to parse
  - Custom E+ parser
  - Must re-implement E+ parsing for any third party tool
- Fields are referenced by index
  - Must add fields to end, otherwise need to transition IDF object.
- Can only have one group of extensible fields per object
  - For example, BuildingSurface:Detailed can only have vertices
- Difficult to validate IDF against IDD without running EnergyPlus
- EnergyPlus team and Department of Energy (DOE) priority
  - Top 10 UserVoice suggestion
  - Will lead to better internal code structure, better maintainability

EnergyPlus JDF (JSON)

- ***Format name change***: *JDF -> epJSON*
- Uses Javascript Object Notation (JSON) based on standards RFC 7159 and ECMA-404
- Key/Value pairs



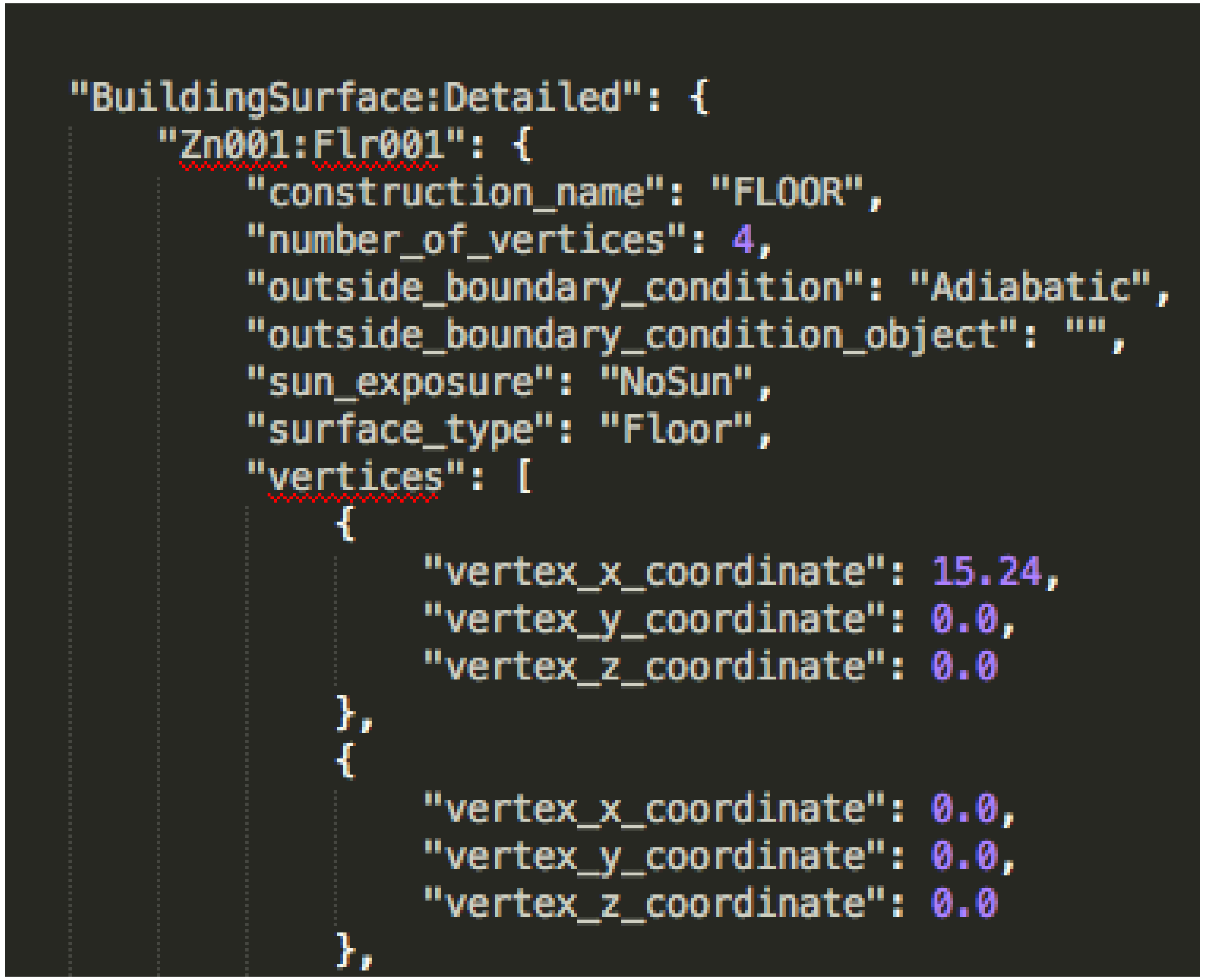


```json
"BuildingSurface:Detailed": {
    "Zn001:Flr001": {
        "construction_name": "FLOOR",
        "number_of_vertices": 4,
        "outside_boundary_condition": "Adiabatic",
        "outside_boundary_condition_object": "",
        "sun_exposure": "NoSun",
        "surface_type": "Floor",
        "vertices": [
            {
                "vertex_x_coordinate": 15.24,
                "vertex_y_coordinate": 0.0,
                "vertex_z_coordinate": 0.0
            },
            {
                "vertex_x_coordinate": 0.0,
                "vertex_y_coordinate": 0.0,
                "vertex_z_coordinate": 0.0
            },
            {
                "vertex_x_coordinate": 0.0,
                "vertex_y_coordinate": 15.24,
                "vertex_z_coordinate": 0.0
            },
            {
                "vertex_x_coordinate": 15.24,
                "vertex_y_coordinate": 15.24,
                "vertex_z_coordinate": 0.0
            }
        ],
        "view_factor_to_ground": 1,
        "wind_exposure": "NoWind",
        "zone_name": "ZONE ONE"
    },
}
```

EnergyPlus JDF (JSON)

Advantages

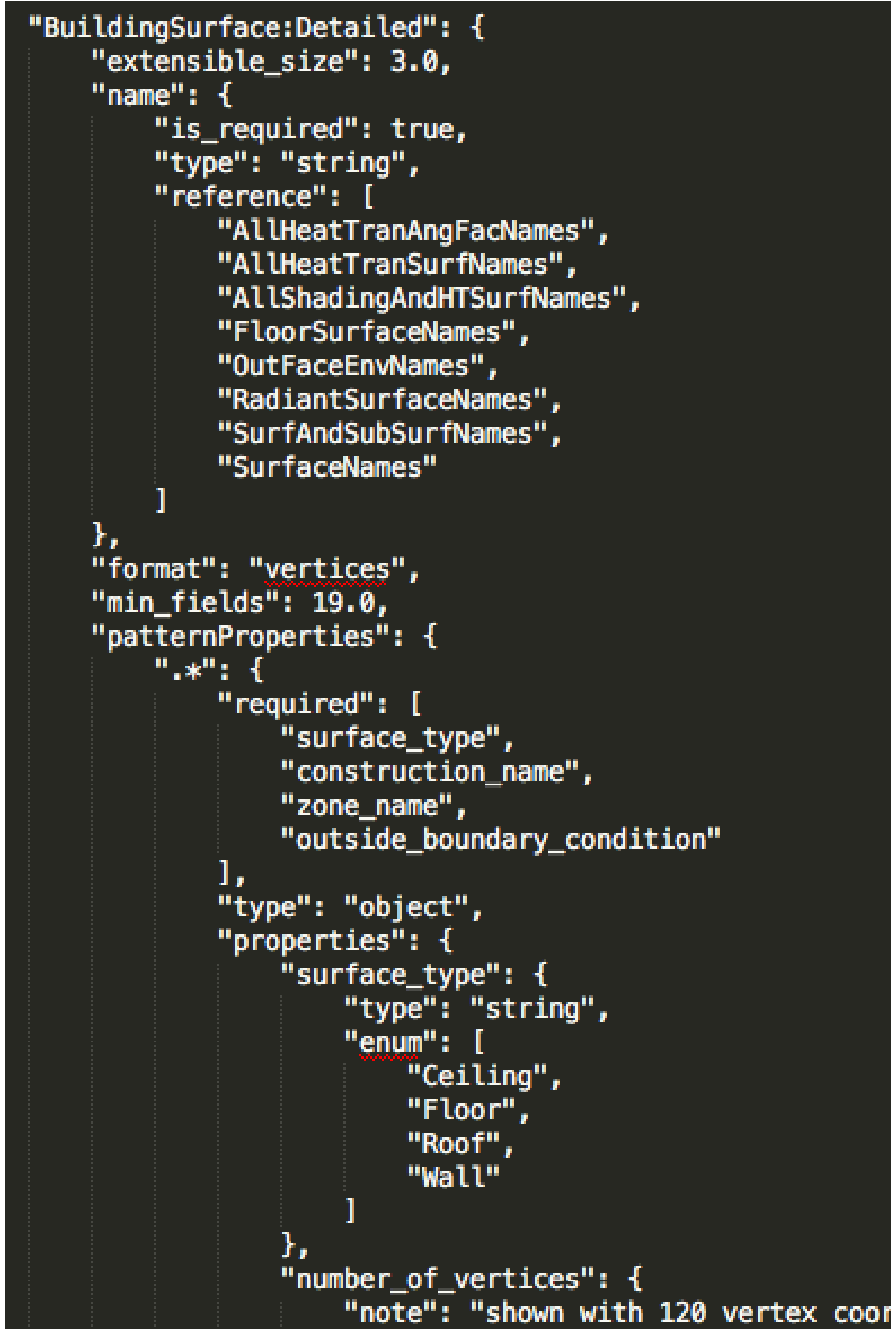
- Key/value based, not positional
- Unlimited length extensible fields
- Multiple extensible fields
- Nearly all languages support JSON parsing
- Easy to add and remove fields, no translation
- Can have extraneous fields
- 1.6x to 5.4x speedup processing input



```
"BuildingSurface:Detailed": {
    "Zn001:Flr001": {
        "construction_name": "FLOOR",
        "number_of_vertices": 4,
        "outside_boundary_condition": "Adiabatic",
        "outside_boundary_condition_object": "",
        "sun_exposure": "NoSun",
        "surface_type": "Floor",
        "vertices": [
            {
                "vertex_x_coordinate": 15.24,
                "vertex_y_coordinate": 0.0,
                "vertex_z_coordinate": 0.0
            },
            {
                "vertex_x_coordinate": 0.0,
                "vertex_y_coordinate": 0.0,
                "vertex_z_coordinate": 0.0
            },
```

EnergyPlus JDD (JSON Schema)

- ***Format name change***: *JDD -> epJSON Schema*
- Uses widely accepted JSON Schema for validation
  - Conceptually similar to XML Schema (XSD)
- Contains all information from IDD
- Automatically generated from IDD

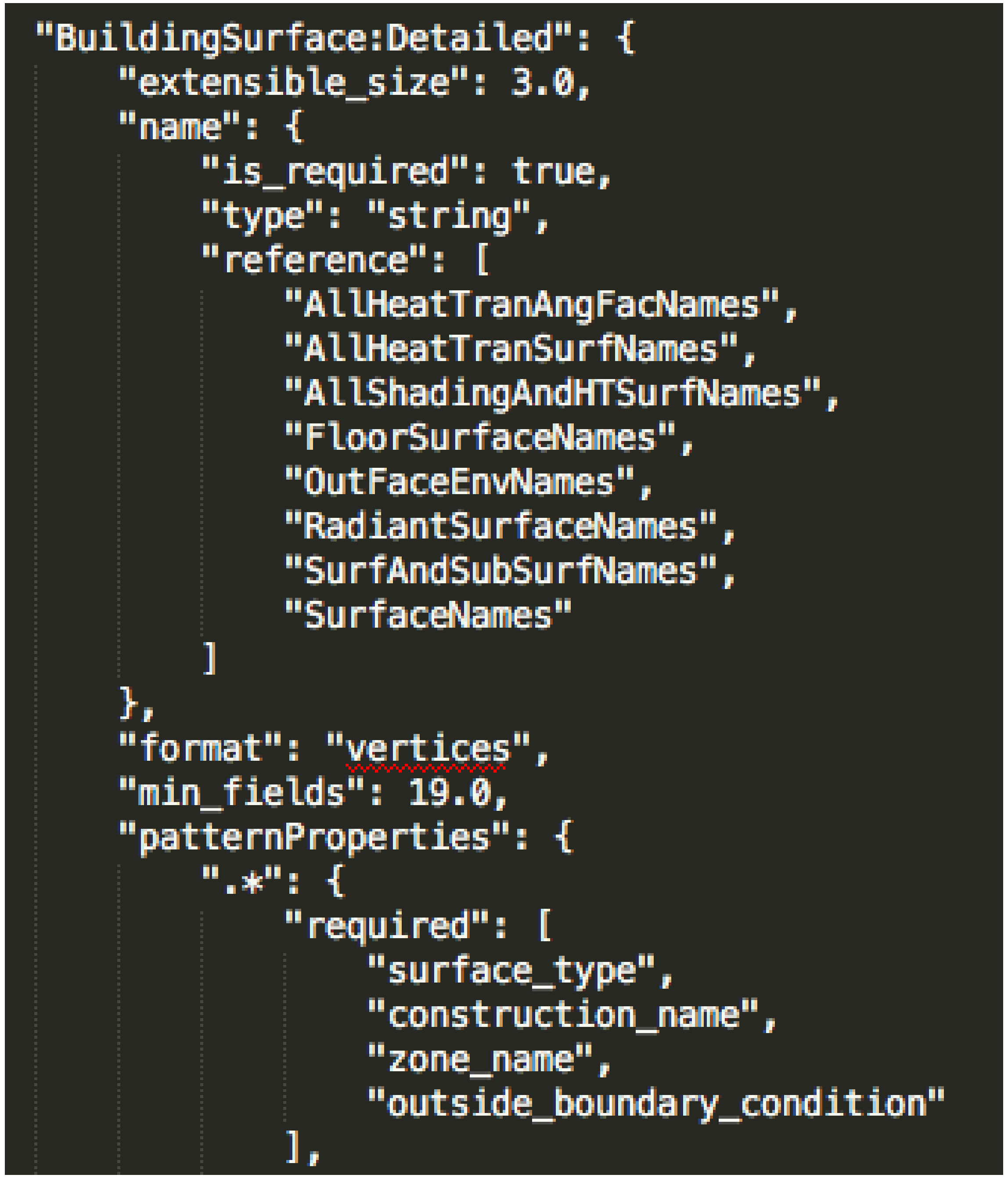


```
"BuildingSurface:Detailed": {
    "extensible_size": 3.0,
    "name": {
        "is_required": true,
        "type": "string",
        "reference": [
            "AllHeatTranAngFacNames",
            "AllHeatTranSurfNames",
            "AllShadingAndHTSurfNames",
            "FloorSurfaceNames",
            "OutFaceEnvNames",
            "RadiantSurfaceNames",
            "SurfAndSubSurfNames",
            "SurfaceNames"
        ]
    },
    "format": "vertices",
    "min_fields": 19.0,
    "patternProperties": {
        ".*": {
            "required": [
                "surface_type",
                "construction_name",
                "zone_name",
                "outside_boundary_condition"
            ],
            "type": "object",
            "properties": {
                "surface_type": {
                    "type": "string",
                    "enum": [
                        "Ceiling",
                        "Floor",
                        "Roof",
                        "Wall"
                    ]
                },
                "number_of_vertices": {
                    "note": "shown with 120 vertex coor
```

EnergyPlus JDD (JSON Schema)

Advantages

- End user can validate JDF against JDD
- Most languages support JSON Schema validation
- No need to write custom validator
- Standardized, explicit programmatic approach to validation
- Future – allows for more complex validation



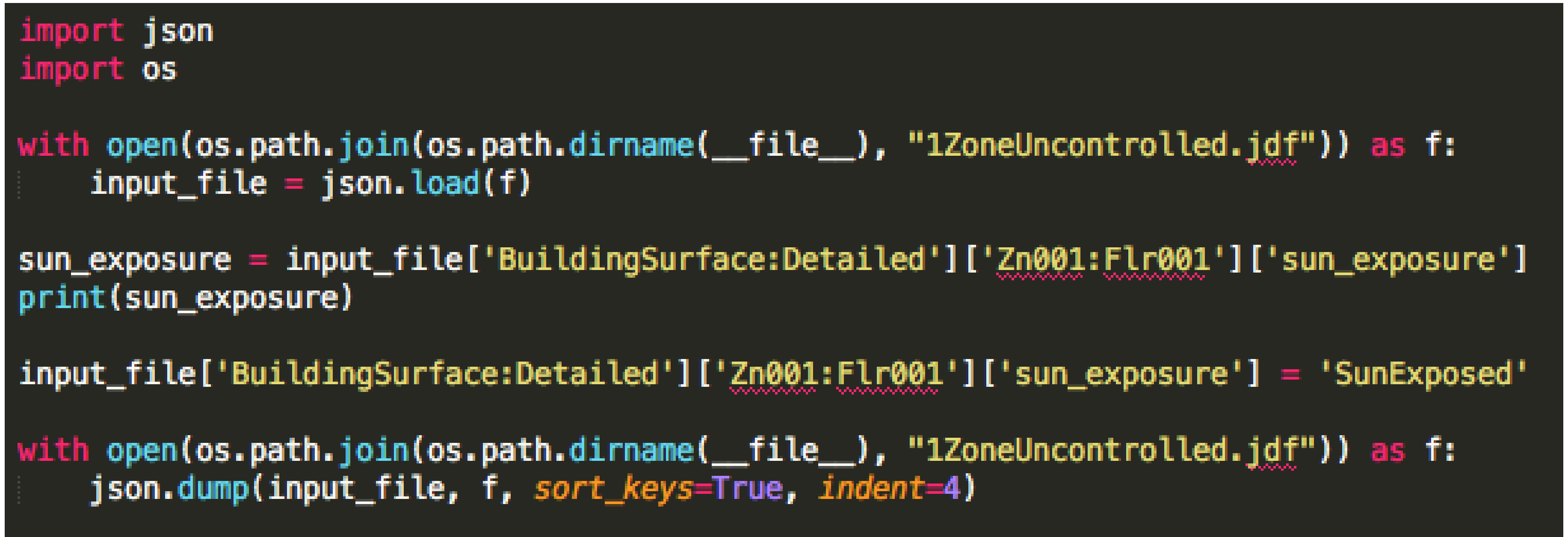
```
"BuildingSurface:Detailed": {
    "extensible_size": 3.0,
    "name": {
        "is_required": true,
        "type": "string",
        "reference": [
            "AllHeatTranAngFacNames",
            "AllHeatTranSurfNames",
            "AllShadingAndHTSurfNames",
            "FloorSurfaceNames",
            "OutFaceEnvNames",
            "RadiantSurfaceNames",
            "SurfAndSubSurfNames",
            "SurfaceNames"
        ]
    },
    "format": "vertices",
    "min_fields": 19.0,
    "patternProperties": {
        ".*": {
            "required": [
                "surface_type",
                "construction_name",
                "zone_name",
                "outside_boundary_condition"
            ],
```

Input Validation Changes

- EnergyPlus input is now case-sensitive
    - Automatically taken care of during translation from IDF to JDF
- More strict validation requirements
    - Previous input processor allowed undocumented inputs as valid
    - “Choice” enumerations were not enforced during input processing
- Fields must be accessed by key (name) instead of index (position)

Modifying JDF

- Easy to programmatically alter JDF
- Can use any language that supports JSON
- Can use existing JSON editing tools
- No need to write IDF parser first



```python
import json
import os

with open(os.path.join(os.path.dirname(__file__), "1ZoneUncontrolled.jdf")) as f:
    input_file = json.load(f)

sun_exposure = input_file['BuildingSurface:Detailed']['Zn001:Flr001']['sun_exposure']
print(sun_exposure)

input_file['BuildingSurface:Detailed']['Zn001:Flr001']['sun_exposure'] = 'SunExposed'

with open(os.path.join(os.path.dirname(__file__), "1ZoneUncontrolled.jdf")) as f:
    json.dump(input_file, f, sort_keys=True, indent=4)
```

IDF-Based Pre-Processors

- Existing pre-processors will not support JDF
  - Must be run on IDF first then translate IDF->JDF
  - Programs could be rewritten in future to work on JDF

- ParametricPreprocessor
  - Parametric:*
- EP-Macro
  - imf → idf
- ExpandObjects
  - HVACTemplate:* → No replacement planned
  - GroundHeatTransfer:* → Replaced with integrated foundation objects

Looking Forward

- *Tentative EnergyPlus team and DOE transition plan*
  - **EnergyPlus 8.8** : JDF input used internally and as experimental input (user feedback and schema changes), automatic IDF -> JDF translation within EnergyPlus
  - **EnergyPlus 8.9** : JDF becomes 1st class citizen along with IDF.
  - **EnergyPlus 9.0 - 9.3** : Deprecate IDF input (deprecation notices in documentation and warning message from command line), but still have automatic translation within EnergyPlus
  - **EnergyPlus 10.0** : Remove IDF input, freeze IDD/IDF, and move translation program out of EnergyPlus
- Existing transition utilities and JDF translation utility will always provide a path to move legacy IDF to JDF
- JSON will provide better supported, easier to use, and more programmatically accessible format

Questions and Discussion

Mark Adams	adamsmb@ornl.gov
Jason Glazer	jglazer@gard.com
Michael J. Witte	mjwitte@gard.com