Walk-in Slide: AU 2014 Social Media Feed

1. Click on the link below, this will open your web browser

http://aucache.autodesk.com/social/visualization.html

2. Use “Extended Display” to project the website on screen if you plan to work on your computer. Use “Duplicate” to display same image on screen and computer.
SD6060 - Autodesk® Revit® 2015, IFC Technical Overview

Angel Velez
Senior Principal Engineer
Class summary

This class will start with a technical overview of the IFC format, including the concepts of schemas and Model View Definitions (MVDs) and how they are used in Revit software. We will then cover the major design concepts for the .NET code, particularly the linker, with the intention of understanding how to make simple modifications to the open source. The class will look at, but not emphasize, Revit 2015 software IFC open source .NET code and the associated Revit software API.
Key learning objectives

At the end of this class, you will be able to:

- Explore the general structure of the IFC file format.
- Understand what’s included in Revit 2015 software for IFC support.
- Study the overall design of the Revit software IFC open source code.
- Learn how to make simple changes to the code.

Note that the class hand-out contains more details; please refer to it for detailed code and instructions not in this presentation.
Introduction to IFC
What is IFC? Standard terms

- IFC = Industry Foundation Classes
  - The format used to exchange data between applications

- Schema = The definition of a particular version of IFC
  - IFC2x3, IFC4, IFC4 Addendum 1

- MVD = Model View Definition
  - The subset of IFC used for a particular workflow
ABCs of IFC (IFC)

IFC = Industry Foundation Classes

- Started in 1994 by Autodesk and industry consortium
- “Classes” refers to initial intention to create common C++ objects to define architectural concepts; now file format based on STEP and ifcXML
- IAI (International Alliance for Interoperability) created to support and promote IFC, now buildingSMART
ABCs of IFC (Schema)

Schema = The definition of a particular version of IFC

- Active versions:
  - IFC2x2 – supported since 2005, generally obsolete
  - IFC2x3 – supported since 2007, most common version
  - IFC4 – partial support since 2012, in progress
    - Will be replaced by IFC4 Addendum 1 before end of 2014
ABCs of IFC (MVD)

MVD = Model View Definition

- An MVD is the subset of IFC suited for a particular workflow. Examples are:
  - Coordination View
  - Structural Analysis View
  - FM (Facilities Handover) View (a.k.a. COBie)
- An IFC file must be generated based on some MVD, perhaps with compatible “Add-ons” such as:
  - QTO (Quantity Take-off)
  - 1st or 2nd level space boundaries
IFC4 Design Transfer MVD (a small part of it …)
IFC top level structure
**IFC top level format to Revit comparison**

- **IfcProject** = Revit Document
- **IfcSite** = Revit Site (if it has any geometry)
- **IfcBuilding** = Revit parameters in Project Information
- **IfcBuildingStorey** = Revit Levels
- **IfcBuildingElements** = Revit Elements
ENTITY IfcWall;
  ENTITY IfcRoot;
    GlobalId : IfcGloballyUniqueId;
    OwnerHistory : IfcOwnerHistory;
    Name : OPTIONAL IfcLabel;
    Description : OPTIONAL IfcText;
  ENTITY IfcObjectDefinition;
  INVERSE
    HasAssignments : SET OF IfcRelAssigns FOR RelatedObjects;
    IsDecomposedBy : SET OF IfcRelDecomposes FOR RelatingObject;
    Decomposes : SET [0:1] OF IfcRelDecomposes FOR RelatedObjects;
    HasAssociations : SET OF IfcRelAssociates FOR RelatedObjects;
  ENTITY IfcObject;
    ObjectType : OPTIONAL IfcLabel;
  INVERSE
    IsDefinedBy : SET OF IfcRelDefines FOR RelatedObjects;
  ENTITY IfcProduct;
    ObjectPlacement : OPTIONAL IfcObjectPlacement;
    Representation : OPTIONAL IfcProductRepresentation;
  INVERSE
    ReferencedBy : SET OF IfcRelAssignsToProduct FOR RelatingProduct;
  ENTITY IfcElement;
    Tag : OPTIONAL IfcIdentifier;
Inside the Revit open source
Why open source?

- Customer flexibility and customization
- Countrification
- Updates independent of Revit’s release cycle
- Allow for outside contributions
Modification

- Need Subversion (SVN) to download and modify source code
- ZIP file included for those without some version of SVN
- All local modifications allowed, uploading is restricted
- Contributions controlled by Open Source steering committee
- All contributions are welcome!
Download the code

- Step-by-step details in class hand-out
- Create a directory for the source code
- Checkout the code
- Update the project files
- Optional: change the version number
What is (and isn’t) the open source

- The open source includes:
  - Export IFC
  - Link IFC
  - Top-level call to Open IFC native code
- The open source doesn’t include (yet):
  - Some export code that hasn’t been ported to open source
  - Open IFC
- All of Link IFC, minus the hooks into Revit, are in the open source
Export IFC overview
Overview of the export code

- The top level code resides entirely in Exporter.cs
- Registered as an external application via IExternalDBApplication

```csharp
private void OnApplicationInitialized(object sender, EventArgs eventArgs)
{
    SingleServerService service =
    ExternalServiceRegistry.GetService(ExternalServices.BuiltInExternalServices.IFCExporterService) as SingleServerService;
    if (service != null)
    {
        Exporter exporter = new Exporter();
        service.AddServer(exporter);
        service.SetActiveServer(exporter.GetServerId());
    }
}
```
Overview of the export code, pt. 2

- The Exporter class must inherit from IExporterIFC
  - Contains the implementation of the exporter
- The entry point for the export is:
  
  ```csharp
  ```
  - `doc`: current document being exported.
  - `exporterIFC`: initialized in native code, and allows interaction between native and .NET code
  - `filterView`: optional, for "Current View Only" export.
**ExporterIFCUtils class**

- IFC exporter relies entirely on Revit API
- ExporterIFCUtils extends API for IFC export-specific functionality.
  - Access to legacy elements with no API support
  - Access to what’s left of the original native export
  - Over time, intend to obsolete most functions or move to other classes if generally useful
ExportIFC Function

This is the main entry point for exporting to IFC. It consists of three main parts:

- **BeginExport**
- **Element traversal**
- **EndExport**

```c
...  
BeginExport(exporterIFC, document, filterView);

InitializeElementExporters();
if (m_ElementExporter != null)
    m_ElementExporter(exporterIFC, document);

EndExport(exporterIFC, document);
...  
```
BeginExport

The BeginExport does the initialization of the export, and creates the top-level entities necessary for the rest of export. This includes:

- Initializing IFCFile based on schema.
- Initializing property sets and quantities to use.
  - Can be overridden – we’ll come back to that.
- Creating unique and top-level IFC entities, include IfcProject, IfcBuilding, and IfcBuildingStoreys.
- Creating commonly used directions and Cartesian points for re-use.
Element traversal

InitializeElementExporters creates the set of delegates that order the element traversal of the document. By default, elements are processed in the following order:

- Spatial elements (rooms, areas, MEP spaces)
- Most non-spatial elements (primarily, elements with 3D geometry)
- Containers (e.g. beam systems, area schemes)
- Grids
- MEP connectors

This order can be overridden in a custom exporter.
Element traversal, pt. 2

- Each element is handled as generically as possible
- Creates 1 or more IfcBuildingElement entities
- GUIDs are generated, consistently if possible
  - Code in place to ensure no GUID duplication
- Relational data may be postponed to EndExport
  - Need to have element to handle map complete before creation
- Hand-out has annotated copy of ExportFooting
EndExport

- Creates any element relations (e.g. wall connection data) cached during element traversal
- Minor amount of internal clean-up
- Finally writes out the IFC file
DYI Export, pt. 1

- 2 ways of customizing the exporter:
  - Modify the open source code directly
  - Create your own exporter that calls the open source code
- Remember to increment version number
Modify the exporter: add a property set

private static void InitPSMTypelnformation(List<PropertySetDescription> commonPropertySets)
{
    PropertySetDescription psm = new PropertySetDescription();
    psm.Name = "Pset_ManufacturerTypeInformation";
    psm.EntityTypes.Add(IFC.EntityType.IfcElement);
    psm.AddEntry(PropertySetEntry.CreateIdentifier("ArticleNumber"));
    psm.AddEntry(PropertySetEntry.CreateLabel("ModelReference"));
    psm.AddEntry(PropertySetEntry.CreateLabel("ModelLabel"));
    PropertySetEntry ifcPSE = PropertySetEntry.CreateLabel("Manufacturer");
    ifcPSE.RevitBuiltInParameter = BuiltInParameter.ALL_MODEL_MANUFACTURER;
    psm.AddEntry(ifcPSE);
    psm.AddEntry(PropertySetEntry.CreateLabel("ProductionYear"));
    
    if (ExportSchema == IFCVersion.IFC4)
    {
        psm.AddEntry(PropertySetEntry.CreateIdentifier("GlobalTradeItemNumber"));
        psm.AddEntry(PropertySetEntry.CreateEnumeratedValue("AssemblyPlace", PropertyType.Label, typeof(Toolkit.IFC4).
        PsetManufacturerTypeInformation_AssemblyPlace));
    }
    commonPropertySets.Add(psm);
}
Modify the exporter: add a property set, pt. 2

- Function won’t be called directly during the export of an element
- Registered as a property set as part of:

  ```csharp
  private static void InitCommonPropertySets(IList<IList<PropertySetDescription>> propertySets, IFCVersion fileVersion)
  ```

- PropertySetDescription contains the definition of 1 IFC property set
  - May have to deal with different definitions for different schemas
- PropertySetEntry contains the definition of one IFC property
- InitPropertySets is the sole function called by InitializePropertySets
  - InitializePropertySets can be overridden
DIY Export, pt. 2

Instead of modifying the exporter directly, a user can create an exporter based on the existing IFC exporter. To do this, you will need to do the following:

- Create a new ExporterApplication class in your custom workspace
- Create a new Exporter class, to override the base exporter
- Override virtual functions as necessary
- Details are in the hand-out
Export customization ideas

- Support for a new IFC entity from data accessible via Revit API
- Support for elements that are non-standard IFC entities
- Support for extended properties for materials
- Support for non-geometric data gathered by custom UI or extensible storage (e.g. file header, user information, zones)
- Support for additional UI options to choose between different export needs
Link IFC overview
Overview of the link code

- Import Registered as a service like the export code
  - Revit.IFC.Import/Importer.cs

- Uses same Import IFC class settings as Open IFC

- Link and Open IFC both go through the Open Source
Overview of the link code, pt. 2

- Code path dependent on the import intent and action
  - IFCImportIntent: Parametric vs. Reference
  - IFCImportAction: Open vs. Link
  - Link IFC = IFCImportAction.Link + IFCImportIntent.Reference
  - Open IFC = IFCImportAction.Open + IFCImportIntent.Parametric
  - Open IFC branch calls ProcessIFCProject to do majority of work

- Two steps: Process and Create
  - Managed by the static IFCImportFile.Create function
Link IFC: Process
IFC entity -> Revit class

- Link code split up into data classes paralleling IFC:
  - IfcProduct -> IFCProduct (in IfcProduct.cs)
  - IfcSite -> IFCSite (in IfcSite.cs)
  - IfcStyledItem -> IFCStyledItem (in IfcStyledItem.cs)
- Some "basic" IFC entities stored directly as existing class
  - IfcCartesianPoint -> XYZ
- Some derived IFC entities don’t yet have a direct counterpart
  - IfcWallStandardCase -> IFCElement
Revit classes

- IFC abstract entities -> Revit abstract classes
- IFC entity fields -> Revit class data fields
- Base class: IFCEntity

```
ENTITY IfcSite
  SUBTYPE OF (IfcSpatialStructureElement);
    RefLatitude : OPTIONAL IfcCompoundPlaneAngleMeasure;
    RefLongitude : OPTIONAL IfcCompoundPlaneAngleMeasure;
    RefElevation : OPTIONAL IfcLengthMeasure;
    LandTitleNumber : OPTIONAL IfcLabel;
    SiteAddress : OPTIONAL IfcPostalAddress;
END_ENTITY;
```

```java
public class IFCSite : IFCSpatialStructureElement {
  double? m_RefLatitude = null;
  double? m_RefLongitude = null;
  double m_RefElevation = 0.0;
  string m_LandTitleNumber = null;
  ...
}
```
Populate Revit classes by calling Process() function for each entity contained in top-level IfcProject
High-level Process() function calls base level function

```cpp
virtual protected void Process(IFCAnyHandle item)
{
    Id = item.StepId;
    EntityType = IFCAnyHandleUtil.GetEntityType(item);
    IFCImportFile.TheFile.EntityMap.Add(Id, this);
    IFCImportFile.TheLog.AddProcessedEntity(EntityType);
}
```
Link IFC: Create
Create

- Create() function works on data created in Process() step
- Create() function may create different data dependent on the import intent and action
  - Currently only covers link case
  - Could create non-Revit data also
Link IFC Requirements for Created Elements

- Fast
- High fidelity
- Maintain parameter data
- Proper category
- Proper materials
- Read-only
- Maintain associativity with IFC file
DirectShapes

- DirectShape element satisfies those requirements
  - API only
  - Maintains parameter data
  - Settable category
  - Settable materials
  - Read-only shape
- Less functionality than parametric Revit elements
  - Can be referenced
- Work in progress
Link IFC case studies
Linking Grids

IfcGrid Entity

- UAxes
- VAxes

- Rectangular grid

- Triangular grid

Revit API Grid Element

- UAxes
- VAxes
- WAxes

- Radial grid
Linking Grids, pt. 2

IfcGrid Entity

- Per Level or Building
- 2D grid lines
- No extra graphics
- Expects 2-3 directions
- (IfcGridAxis) optional name

Revit API Grid Element

- Per Building
- 3D element
- Grid type controls appearance
- Any number of directions
- Required unique name

Issues not limited to link
Linking Grids, pt. 3

- Keep track of “identical” IfcGridAxes
  - TODO: Set vertical extents of the grid

- Use Grid element, not DirectShape
  - Use default grid type to determine graphical display

- Code in IFCGrid.cs and IFCGridAxis.cs
Linking Zones

IfcZone Entity

Revit API Zone Element

A collection of IfcSpaces and IfcZones
**Linking Zones**

### IfcZone Entity

ENTITY IfcZone;
ENTITY IfcRoot;
  GlobalId : IfcGloballyUniqueId;
  OwnerHistory : IfcOwnerHistory;
  Name : OPTIONAL IfcLabel;
  Description : OPTIONAL IfcText;
END_ENTITY;

ENTITY IfcObjectDefinition;

ENTITY IfcObject;
  ObjectType : OPTIONAL IfcLabel;
  IsDefinedBy : SET OF IfcRelDefines FOR RelatedObjects;
END_ENTITY;

ENTITY IfcGroup;
  IsGroupedBy : IfcRelAssignsToGroup FOR RelatingGroup;
END_ENTITY;

ENTITY IfcAssignments;
  IsDecomposedBy : SET OF IfcRelDecomposes FOR RelatedObjects;
  Decomposes : SET [0:1] OF IfcRelDecomposes FOR RelatedObjects;
  HasAssociations : SET OF IfcRelAssociates FOR RelatedObjects;
END_ENTITY;

ENTITY IfcRelAssigns;
  Source : IfcRelAssignsToGroup FOR RelatingGroup;
END_ENTITY;

### Revit API Zone Element

**Zone Class**

<table>
<thead>
<tr>
<th>Members</th>
<th>See Also</th>
<th>Send Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provides access to the Zone Element in Autodesk Revit.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Namespace</td>
<td>Reference Model Information</td>
<td>Assembly: Autodesk Revit API Version 2014.8.0.0</td>
</tr>
</tbody>
</table>

**Syntax**

**C#**

```csharp
public class Zone : Element
```

**Visual Basic**

```vbnet
Public Class Zone
    Inherits Element
```

**Visual C++**

```cpp
public ref class Zone : public Element
```

**Remarks**

This object derived from the Element base object and such supports all the methods of that object such as the ability to retrieve the parameters of that object. **This object is only available in Revit API**

---

A collection of IfcSpaces and IfcZones  
A collection of MEP spaces (only)
Creating an IfcZone in Revit

- Create an IfcZone subcategory
- Create DirectShapes with a grouping of the space geometry
- Add an "IfcZone" shared parameter to the spaces
- Code in IFCZone.cs
Conclusion
Certification

- IFC2x3 Coordination View 2.0 export certified
  - Revit 2013 and Revit LT 2014 to current
  - Working on Import certification
Session Feedback

- Via the Survey Stations, email or mobile device
- AU 2015 passes given out each day!
- Best to do it right after the session
- Instructors see results in real-time
Students, educators, and schools now have
FREE access to Autodesk design software & apps.
Download at www.autodesk.com/education
Earn your professional Autodesk Certification at AU

Visit the AU Certification Lab