

Session 9

IFC for Interoperability Part 2

Martijn de Riet, MdR Advies | Bimforge

Class Description

This class is a follow-up on the class presented during RTC Europe 2013. In this class we will be looking into how to implement IFC in Revit. Where the first IFC for Interoperability class focussed on the general principle of IFC, this class will be more of a practical implementation. The basic knowledge of IFC, what it is and how it works, is considered a familiar subject to the attendees. Instead we will be looking at ways to enhance the implementation of IFC in your project with several best practises: - What do all those fancy buttons in the Open Source IFC Exporter mean? - IFC for MEP: leveraging Type Enumerations. - Using Key Schedules to set IFC Type. - Creating custom Property Sets and adding properties - Setting up Classifications to add to your IFC Export. - The difference between IfcSpace and IfcZone and how to leverage this. - Other various enhancements made to the Autodesk Open Source IFC Exporter in the last year. In short: this class will give you working knowledge to embrace the full potential of the IFC scheme, improve your exports and create more manageable, data-rich IFC files that can fully leverage the power of IFC: the ability to transfer Information throughout the entire Design, Build and Maintain process.



About the Speaker:

Martijn de Riet is owner of MdR Advies, a consultancy firm specialized in information handling in complex workflows. His main focus lies in developing and implementing company and project processes that span multiple building phases, disciplines and software platforms. Besides this, Martijn is co-founder of Bimforge, a company that specializes in Information Processing. With a completely different view on BIM, Bimforge aims at separating the User Interface of BIM (e.g. the "Model") from the actual Information. Thus creating the possibility to develop a BIM workflow completely independent from the tools and file formats restricting current BIM implementations.

Martijn is a member of the Dutch Revit User Group, co-author of the Dutch RUG "Best Practises" for Revit and author / creator of the Dutch RUG Revit Standards Master Template.

He provides lectures for companies, technical universities, seminars and such on a regular basis.

Introduction

This class is a follow-up for the 2013 "IFC For Interoperability" class. Since I did most of the basic stuff in that first class, this will probably be somewhat more advanced. If you're entirely new to the world of IFC, I can suggest some light bedtime reading material and even a 45min class from Autodesk University. However, since it's all my own work that might come off as a little cocky. So I'm not going to do so.

Just kidding.

I added the list at the end of this handout.

So what subjects won't we be talking about:

- Import and Export mapping tables.
- ParameterMappingTables
- General Modelling Requirements when working with IFC.
- Hacking Revit using IFC

These are all topics covered in earlier sessions. Is there anything interesting left you ask? How about:

- Differences between IFC2x3 and IFC4.
- Changes in IFC export with Revit 2015
- Changes in IFC import with Revit 2015
- IFC Linking with Revit 2015
- Using Type Enumerations
- Exporting Areas and Zones
- Open Source Exporter improvements:
 - UI changes
 - Classifications
 - Custom PropertySets for Export
 - Exporting Linked Files
- Combining IFC's in a single model

IFC2x3 vs IFC4

You might not have noticed but earlier this year IFC4 was officially released. Since this is the first IFC version release I consciensely witnessed I was kind of expecting a big fuss, like when Autodesk (or any other software company) releases a new version. Even more so since Revit has a yearly update cycle while IFC versions stay current for about 8 years.

Unfortunately I was somewhat disappointed: the whole release thing was somewhat, well, let's call it unobtrusive. At best. Being slightly less polite I would think the BuildingSmart PR department is run by the same people responsible for the IFC User Guide.

Anyway: Revit was (one of) the first modelling tools to support IFC4. Which, in all fairness, only means Revit is able to process IFC4 files that do not use any of the new IFC4 capabilities on geometry or type definition. In other words: Revit can process an upgraded IFC2x3 (on export or import) but not a native IFC4 file fully (or partially) leveraging IFC4 capabilities.

However: it is not necessarily a bad thing. Up to this date, roughly 8 months later, I still haven't seen any software that does support IFC with full functionality. By putting the possibility on the table (the Open Source Exporter also supports IFC4) it is an option at the very least.

Even more so, it's not even possible to get IFC4 certified yet.

BuildingSmart is still working on the certification process. As far as I'm concerned they can take all the time they want if that leads to a more stern certification.

But it does lessen the feeling of urgency for most software vendors to start (fully) supporting IFC4.

Added functionality in IFC

So what are the differences between IFC2x3 and IFC4 anyway? Well, when it comes to stuff a user needs to worry about: not so much. A few new Classes were added, some new Properties defined.

Besides that, most of the work concentrated on expanding and improving the (geometric) object definitions. Which has resulted, to be fair in (potentially) quite spectacular results.

```

#48= IFCDERIVEDUNITELEMENT(#19,-1);
#49= IFCDERIVEDUNIT((#47,#48),LINEARSTIFFNESSUNIT,,$);
#50= IFCDERIVEDUNITELEMENT(#29,1);
#51= IFCDERIVEDUNITELEMENT(#19,-1);
#52= IFCDERIVEDUNITELEMENT(#26,-1);
#53= IFCDERIVEDUNIT((#50,#51,#52),ROTATIONALSTIFFNESSUNIT,,$);
#54= IFCTABLECOLUMN($,Type,Type name,$,$);
#55= IFCTABLECOLUMN($,Types,$,$);
#56= IFCTABLE('Dimensions',(#92,#98),(#55,#57,#60,#63,#64,#69,#72,#75,#79,#80,#81,#88));
#57= IFCTABLECOLUMN($,hoogte,$,$,$);
#58= IFCTABLE('Identity Data',(#93,#99),(#55,#59,#61,#62,#67,#68,#70,#73,#74,#87));
#59= IFCTABLECOLUMN($,Keynote,$,$,$);
#60= IFCTABLECOLUMN($,opstand_dikte,$,$,$);
#61= IFCTABLECOLUMN($,Fire Rating,$,$,$);
#62= IFCTABLECOLUMN($,Manufacturer,$,$,$);
#63= IFCTABLECOLUMN($,width,$,$,$);
#64= IFCTABLECOLUMN($,breedte,$,$,$);
#65= IFCTABLE('Analytical Properties',(#94,#100),(#55,#66,#71,#76,#77,#78));
#66= IFCTABLECOLUMN($,Solar Heat Gain Coefficient,$,$,$);
#67= IFCTABLECOLUMN($,Assembly Code,$,$,$);
#68= IFCTABLECOLUMN($,URL,$,$,$);
#69= IFCTABLECOLUMN($,luik_dikte,$,$,$);
#70= IFCTABLECOLUMN($,Model,$,$,$);
#71= IFCTABLECOLUMN($,Heat Transfer Coefficient (u),$,$,$);
#72= IFCTABLECOLUMN($,luik_oversetek,$,$,$);
#73= IFCTABLECOLUMN($,Cost,$,$,$);
#74= IFCTABLECOLUMN($,Description,$,$,$);
#75= IFCTABLECOLUMN($,Thickness,$,$,$);
#76= IFCTABLECOLUMN($,Visual Light Transmittance,$,$,$);
#77= IFCTABLECOLUMN($,Thermal Resistance (R),$,$,$);
#78= IFCTABLECOLUMN($,Analytic Construction,$,$,$);
#79= IFCTABLECOLUMN($,rough width,$,$,$);
#80= IFCTABLECOLUMN($,Height,$,$,$);
#81= IFCTABLECOLUMN($,lengte,$,$,$);
#82= IFCTABLE('Materials and Finishes',(#95,#101),(#55,#83,#84));
#83= IFCTABLECOLUMN($,opstand_materiaal,$,$,$);
#84= IFCTABLECOLUMN($,luik_materiaal,$,$,$);
#85= IFCTABLE('Construction',(#96,#102),(#55,#86,#89));
#86= IFCTABLECOLUMN($,Construction Type,$,$,$);
#87= IFCTABLECOLUMN($,Type Comments,$,$,$);
#88= IFCTABLECOLUMN($,Rough Height,$,$,$);
#89= IFCTABLECOLUMN($,Function,$,$,$);
#90= IFCTABLE('IFC Parameters',(#97,#103),(#55,#91));
#91= IFCTABLECOLUMN($,operation,$,$,$);
#92= IFCTABLEROW((IFCLABEL('DL1592Z_dagmaat1500x0900'),IFCPOSITIVELENGTHMEASURE(299.999999544),IFCPOSITIVELENGTHMEASURE(59.999999908),IFCLABEL(''),IFCLABEL(''),IFCLABEL('Staka'),IFCLABEL(''),IFCLABEL('DL1592Z')),IFREAL(0.0),IFREAL(0.0)),.F.);
#93= IFCTABLEROW((IFCLABEL('DL1592Z_dagmaat1500x0900'),IFCLABEL(''),IFCLABEL(''),IFCLABEL('Staka'),IFCLABEL(''),IFCLABEL('DL1592Z')),IFREAL(0.0),IFREAL(0.0)),.F.);
#94= IFCTABLEROW((IFCLABEL('DL1592Z_dagmaat1500x0900'),IFCLABEL(''),IFCLABEL(''),IFCLABEL('Staka'),IFCLABEL(''),IFCLABEL('DL1592Z')),IFREAL(0.0),IFREAL(0.0)),.F.);
#95= IFCTABLEROW((IFCLABEL('DL1592Z_dagmaat1500x0900'),IFCLABEL(''),IFCLABEL(''),IFCLABEL('Staka'),IFCLABEL(''),IFCLABEL('DL1592Z')),IFREAL(0.0),IFREAL(0.0)),.F.);
#96= IFCTABLEROW((IFCLABEL('DL1592Z_dagmaat1500x0900'),IFCLABEL(''),IFCLABEL(''),IFCLABEL('Staka'),IFCLABEL(''),IFCLABEL('DL1592Z')),IFREAL(0.0),IFREAL(0.0)),.F.);
#97= IFCTABLEROW((IFCLABEL('DL1592Z_dagmaat1500x0900'),IFCLABEL(''),IFCLABEL(''),IFCLABEL('Staka'),IFCLABEL(''),IFCLABEL('DL1592Z')),IFREAL(0.0),IFREAL(0.0)),.F.);
#98= IFCTABLEROW((IFCLABEL('DL97ZZ_dagmaat0900x0700'),IFCPOSITIVELENGTHMEASURE(299.999999544),IFCPOSITIVELENGTHMEASURE(59.999999908),IFCLABEL(''),IFCLABEL(''),IFCLABEL('Staka'),IFCLABEL(''),IFCLABEL('DL97ZZ')),IFREAL(0.0),IFREAL(0.0)),.F.);
#99= IFCTABLEROW((IFCLABEL('DL97ZZ_dagmaat0900x0700'),IFCLABEL(''),IFCLABEL(''),IFCLABEL('Staka'),IFCLABEL(''),IFCLABEL('DL97ZZ')),IFREAL(0.0),IFREAL(0.0)),.F.);
#100= IFCTABLEROW((IFCLABEL('DL97ZZ_dagmaat0900x0700'),IFCLABEL(''),IFCLABEL(''),IFCLABEL('Staka'),IFCLABEL(''),IFCLABEL('DL97ZZ')),IFREAL(0.0),IFREAL(0.0)),.F.);
#101= IFCTABLEROW((IFCLABEL('DL97ZZ_dagmaat0900x0700'),IFCLABEL(''),IFCLABEL(''),IFCLABEL('Staka'),IFCLABEL(''),IFCLABEL('DL97ZZ')),IFREAL(0.0),IFREAL(0.0)),.F.);
#102= IFCTABLEROW((IFCLABEL('DL97ZZ_dagmaat0900x0700'),IFCLABEL(''),IFCLABEL(''),IFCLABEL('Staka'),IFCLABEL(''),IFCLABEL('DL97ZZ')),IFREAL(0.0),IFREAL(0.0)),.F.);
#103= IFCTABLEROW((IFCLABEL('DL97ZZ_dagmaat0900x0700'),IFCLABEL(''),IFCLABEL(''),IFCLABEL('Staka'),IFCLABEL(''),IFCLABEL('DL97ZZ')),IFREAL(0.0),IFREAL(0.0)),.F.);
#104= IFCMETRIC('Dimensions',$,NOTDEFINED,,$,2014-05-08T11:53:10',,EQUALITY,$,$,0.0);
#105= IFCMETRIC('Identity Data',$,NOTDEFINED,,$,2014-05-08T11:53:10',,EQUALITY,$,$,0.0);
#106= IFCMETRIC('Analytical Properties',$,NOTDEFINED,,$,2014-05-08T11:53:10',,EQUALITY,$,$,0.0);
#107= IFCMETRIC('Materials and Finishes',$,NOTDEFINED,,$,2014-05-08T11:53:10',,EQUALITY,$,$,0.0);
#108= IFCMETRIC('Construction',$,NOTDEFINED,,$,2014-05-08T11:53:10',,EQUALITY,$,$,0.0);
#109= IFCMETRIC('IFC Parameters',$,NOTDEFINED,,$,2014-05-08T11:53:10',,EQUALITY,$,$,0.0);
#110= IFCBOJECTIVE('',$,NOTDEFINED,,$,2014-05-08T11:53:10',,EQUALITY,$,$,0.0);
#111= IFCDORTYPE('15vZDJAhv9ee6Ingppa4v8',#6,27_DO_RB_dakluik_staka,$,$,$,NOTDEFINED,.,T,,$);
#112= IFCREDECLARES('3eu0Q6vSr3pa88zs_vPQ',#6,$,$,#23,(#111));
#114= IFCRELASSOCIATESCONSTRAINT('1ohnOSITXFWv2GF2Bcu8m2',#6,ConstraintAssoc,'constraint Associates',(#111),$,#110);
ENDSEC;
END-ISO-10303-21;

```

Image 1: Code example

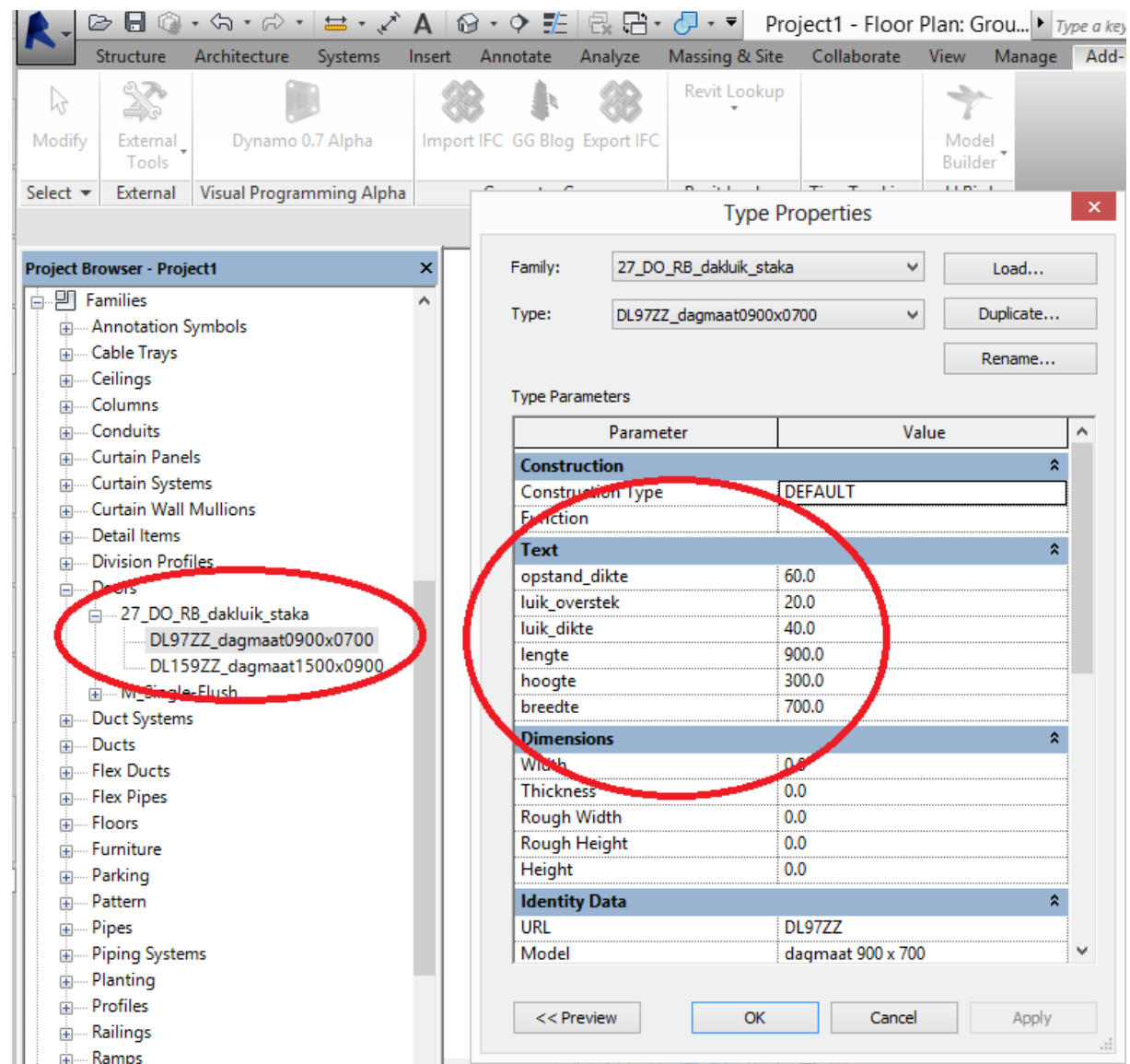


Image 2: Two separate types in Revit

Now, to be fair: these IFCs were not created using the Revit Exporter. And the import was not done using the standard Revit importer. This example was created by Jon Mirtschin from GeometryGym and imported with his custom Revit importer. Nonetheless: it wouldn't have been possible without the expanded geometry definition options defined in IFC4.

What else?

To give you a list of the biggest advancements, in the order Jon Mirtschin describes them in his handout from RTC Australasia 2014 (Session 11: Generative BIM using IFC4):

- Documentation and Guidance improvements
- Advanced Model View Definitions
- Point Cloud and Triangulated Face Set
- Introducing NURBS
- Improved Shape Representations
- Introducing Cardinal Points
- Improved MEP Connectivity and System Models
- Improved Structural Analysis options
- Improved Energy Analysis options
- Simplified IfcXML

At this point I would refer you to Jon on any of these subjects: www.geometrygym.com. Except for the first one: Documentation improvements.

Where the IFC2x3 website (<http://goo.gl/hVs0vW>) is an absolute nightmare, being setup in some sort of stone age kind of way, vaguely resembling the first website I ever made in Microsoft FrontPage 12 years ago, the IFC4 website (<http://goo.gl/HRHd55>) looks pretty much up to date.

They still didn't think of putting in a "Dummy" page with some basic explanations though. However, there are some great improvements:

- A Table of Contents and Introduction page have been added
- An Index page.
- Some code examples.
- And the best one: using Ctrl+F opens up your browser search options. Whereas the FrontPage PoS that is the IFC2x3 website cannot handle this, the IFC4 website is searchable. Especially when searching the index page, you'll get results rather quickly.

When to use Ifc4?

Well, like I said: it's not really usable in the outside world. So why bother?

Nonetheless, I decided to redo my import and export mapping tables and rewrite them to Ifc4. First of all: the documentation is way better, so it's easier to get your stuff mapped correctly. Second: it has more logic to it, less exceptions and more predefined types which mapping just so much easier (especially when using Revit 2015).

Besides that: in case you use a mapping not present in Ifc2x3 the exporter will default to known Ifc2x3 Classes and Types making the Ifc4 versions "Userdefined" ones. So there's no chance of losing information upon export really.

And when it really bogs out, and has no idea what to do with your import you get a need little html-based error report as shown below.

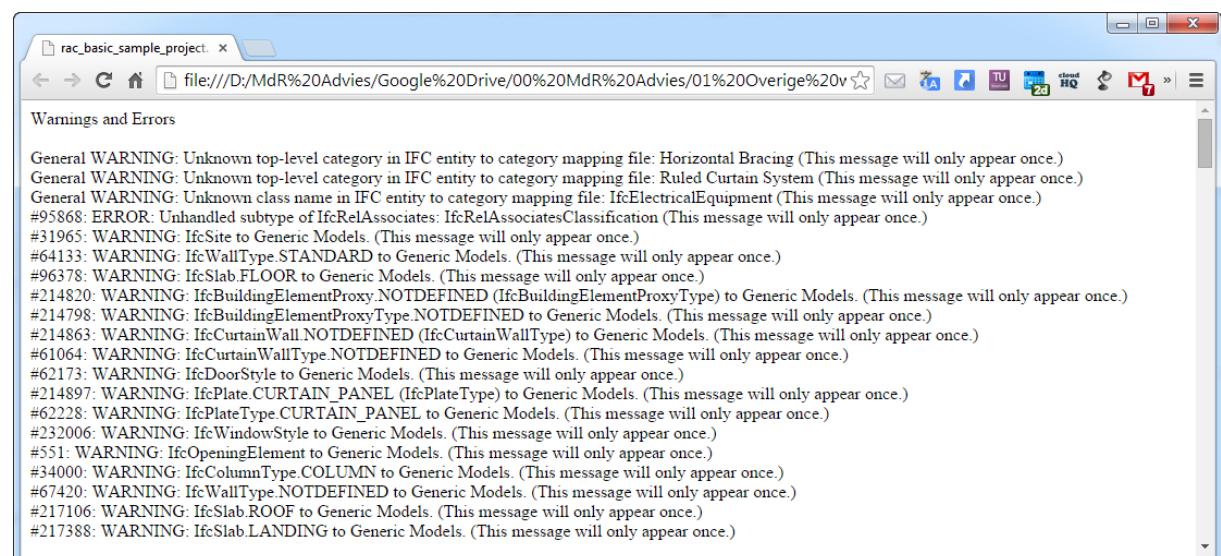


Image 3: Error report generated when exporting with IFC4 mapping and re-importing with IFC2x3 mapping

Changes in IFC handling for Revit 2015

Exporting an IFC

Not much going on here. In the out of the box exporter that is. We'll deal with the Open Source Exporter later on.

Opening an IFC

There is some pretty need stuff happening here.

First off: a nice little checkbox appeared in the Open IFC dialogue box: "AutoJoin Elements". This one is particularly handy when importing models containing prefabricated Wall components. In previous versions these would be auto-joined upon import. Unchecking this options means the auto-join function is disabled.

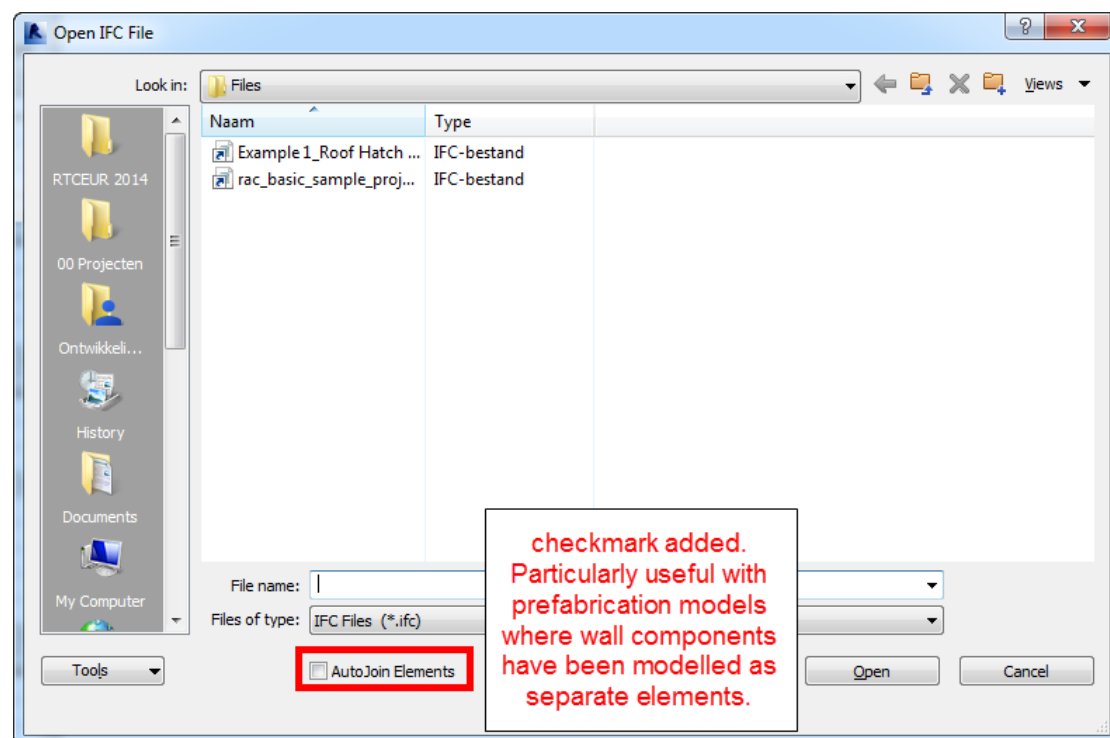


Image 4: AutoJoin Elements option

Let's look at the 2015 Import Options vs the old 2014 Import Options.

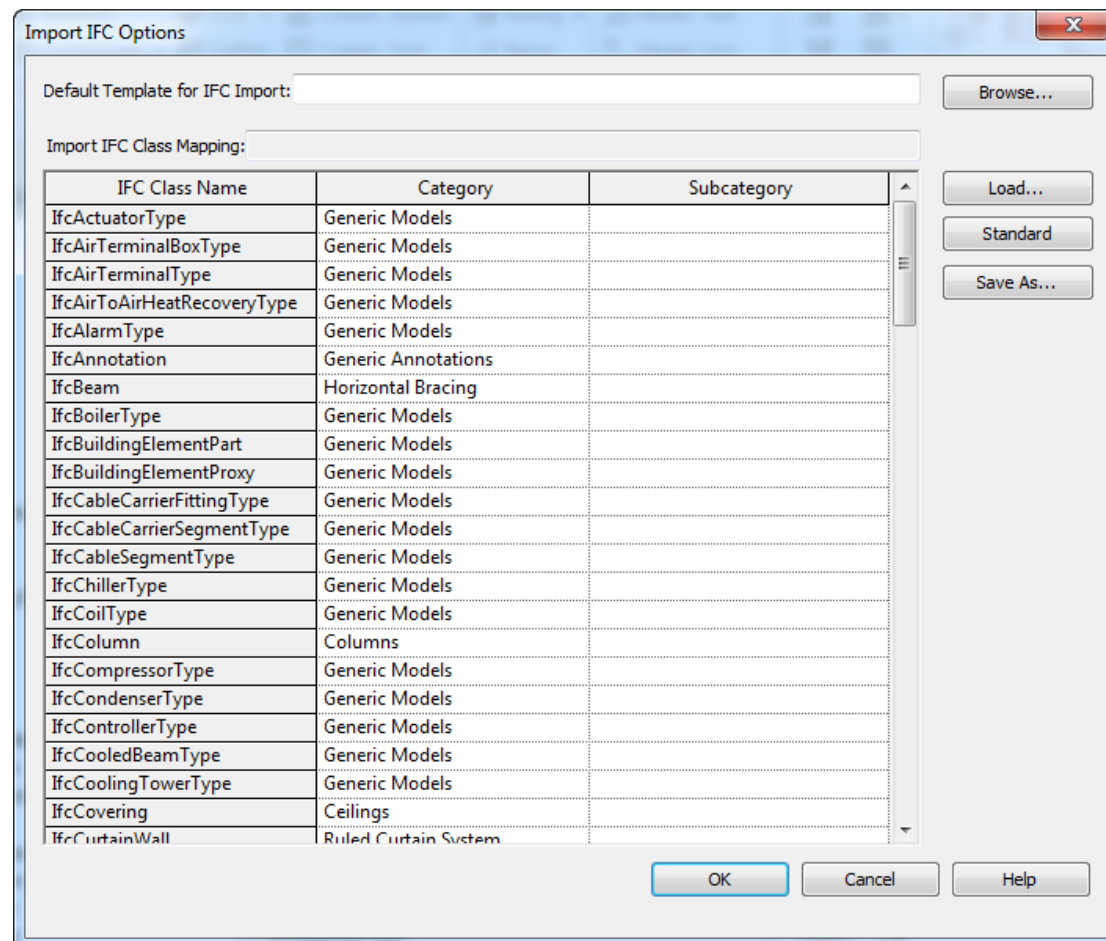


Image 5: Revit 2014 IFC Import options

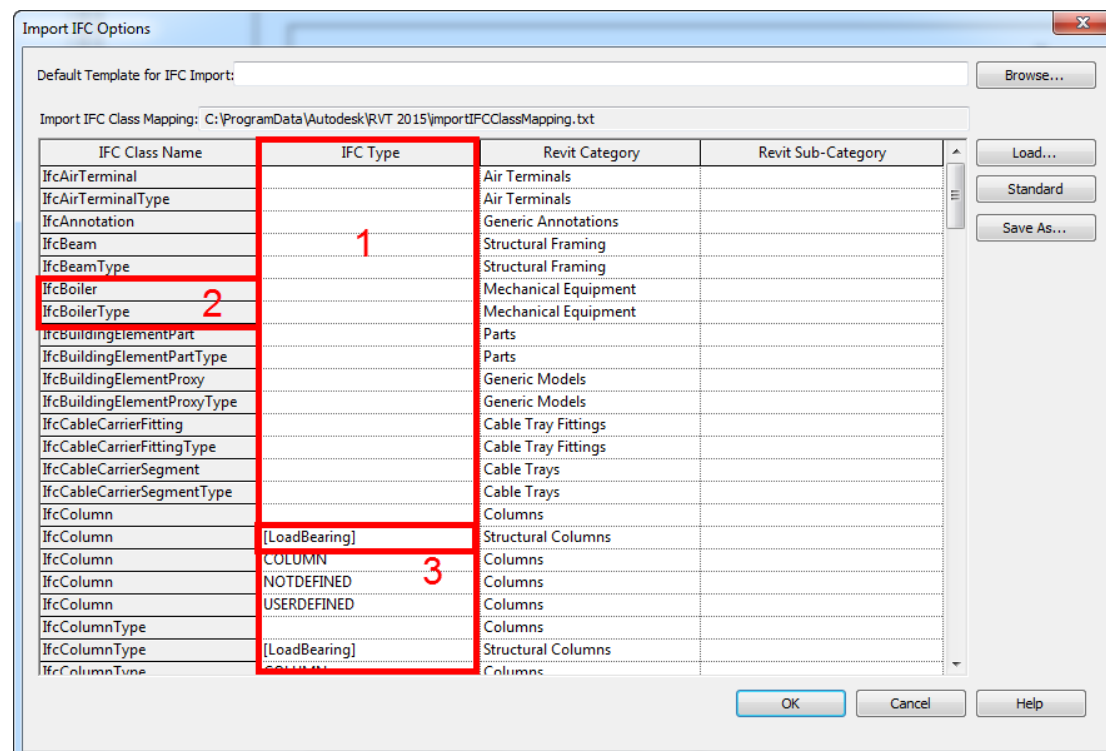


Image 6: Revit 2015 IFC Import options

There's basically 3 major changes:

1. The IFC Type can be used when importing. A lot of IfcClasses have so-called "Type Enumerations", which can roughly be compared to Revit Family Types. For instance an IfcAlarmType (IFC4 only) has the Enumerations:
 - a. BELL: An audible alarm.
 - b. BREAKGLASSBUTTON: An alarm activation mechanism in which a protective glass has to be broken to enable a button to be pressed.
 - c. LIGHT: A visual alarm.
 - d. MANUALPULLBOX: An alarm activation mechanism in which activation is achieved by a pulling action.
 - e. SIREN: An audible alarm.
 - f. WHISTLE: An audible alarm.
 - g. USERDEFINED: User-defined type.
 - h. NOTDEFINED:

These alarm types go into different Revit Categories (Communication Devices, Fire Safety Devices, Lighting Devices) and therefore couldn't be properly mapped in Revit 2014. Now they can, since you can define different Revit (Sub)Categories for each IFC Type.

2. Both the IfcClass (e.g. IfcBoiler) and the Object Instance (e.g. IfcBoilerType), an actual occurrence, are now listed. Differentiating between the two is overkill in most situations. You basically only need the xxxType addition if you're planning on defining TypeEnumerations.

However, some software will use the IfcClassType syntax and others will use the IfcClass syntax. Listing them both means Revit does not have any issues with either of these.

Please do note that some IfcClasses do not have an IfcClassType defined. For example the IfcSite does not have a Type version. This is because there really should only be one instance of IfcSite in your project. By definition you can't have multiple types.

3. In some cases the IfcClass definition is somewhat crooked. For instance when it comes to Columns. IfcColumns can be both bearing and architectural, which is kind of weird (since they are radically different types of objects). The Revit 2015 Importer

solves this issue by looking at the property "LoadBearing". If this property is labelled "True" the Column is mapped to Structural Columns. If not, it is mapped to the Architectural Columns. It could be great if this functionality is expanded towards other Properties and Classes.

4. Then there's a few minor changes such as the deprecation of IfcElectricalEquipment (which isn't a real Ifc Class anyway) and addition of previously uncontrollable types such as IfcSpace (which isn't available for mapping in the 2014 importer).

IFC Linking with Revit 2015

A (potentially) spectaculair new development is the ability to Link IFCs into your Revit project. This makes the handling of IFC imports a lot faster and easier...

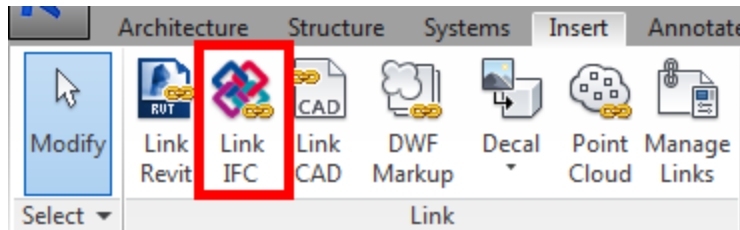


Image 7: IFC Linking added to the Insert toolbar

To understand what can (and cannot) be done with Linking we need to take a thorough look at the process under the hood.

1. When pushing the "Link IFC" button you can select an IFC file to be linked (keep an eye on the Filetype the link screen is referring to), see image 8.
2. When selecting an IFC file, Revit will create two additional files, besides the original IFC file (nr1), as shown in Image 9:
a Revit project file based on the IFC (nr2) and
a Shared Parameter text file (nr3).
3. The project template file that is used is the first one listed in the File Locations tab of the Revit Options, see Image 10. Keep in mind that this template file should not contain additional Phases or Levels to your Project Template file because chances are this will screw up your import, see Image 11 and Image 12.

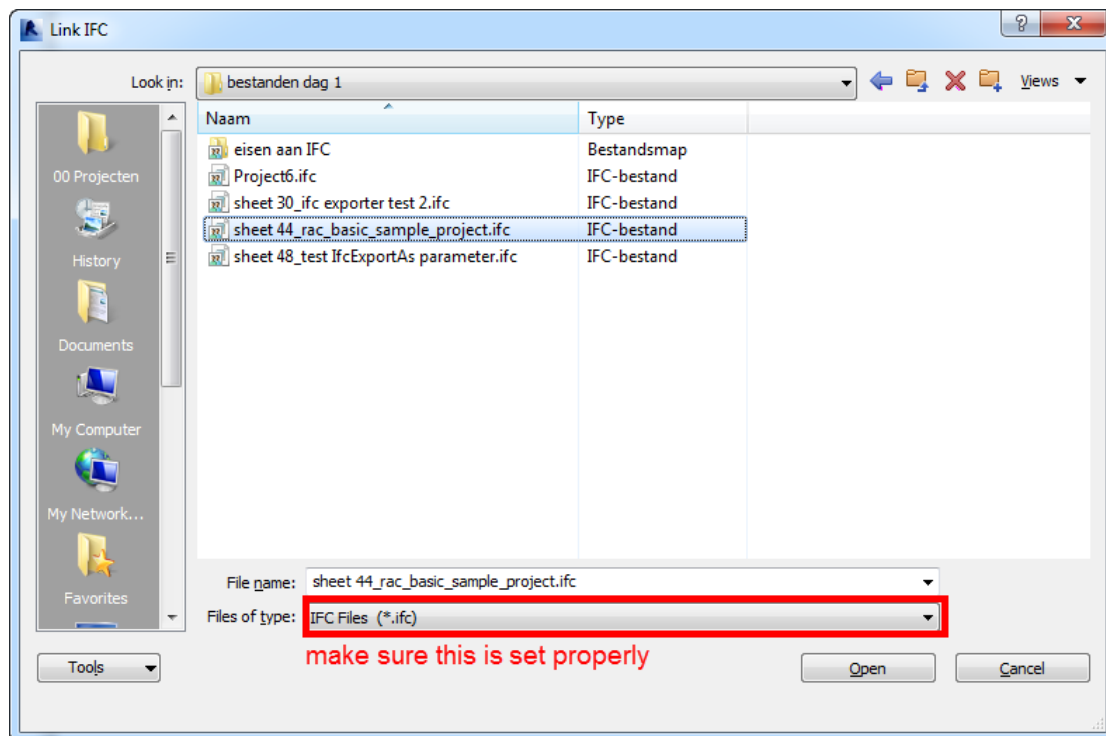


Image 8: Set the IFC filetype properly.

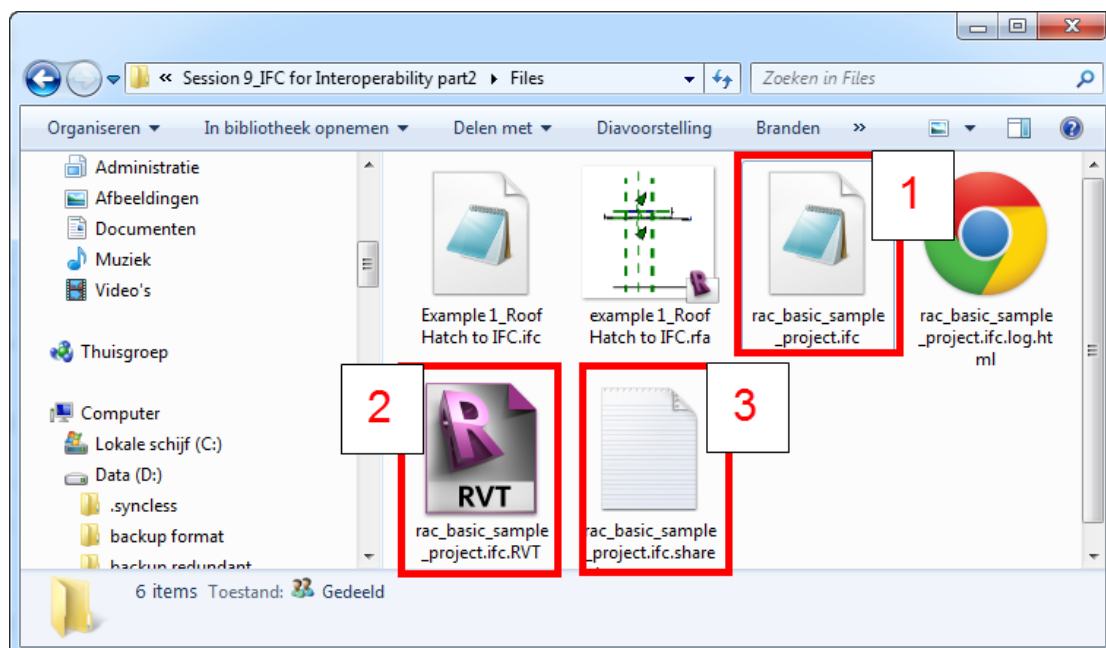


Image 9: Additional files created by the IFC Linking process

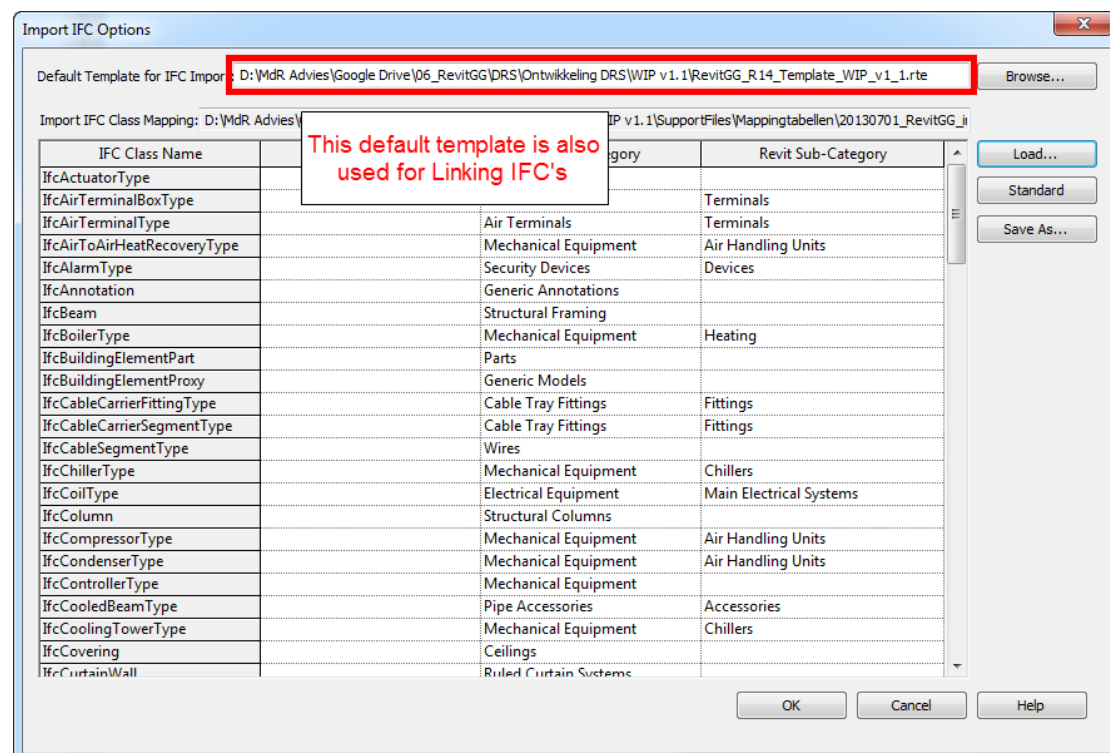


Image 10: Template used for Linking IFC's

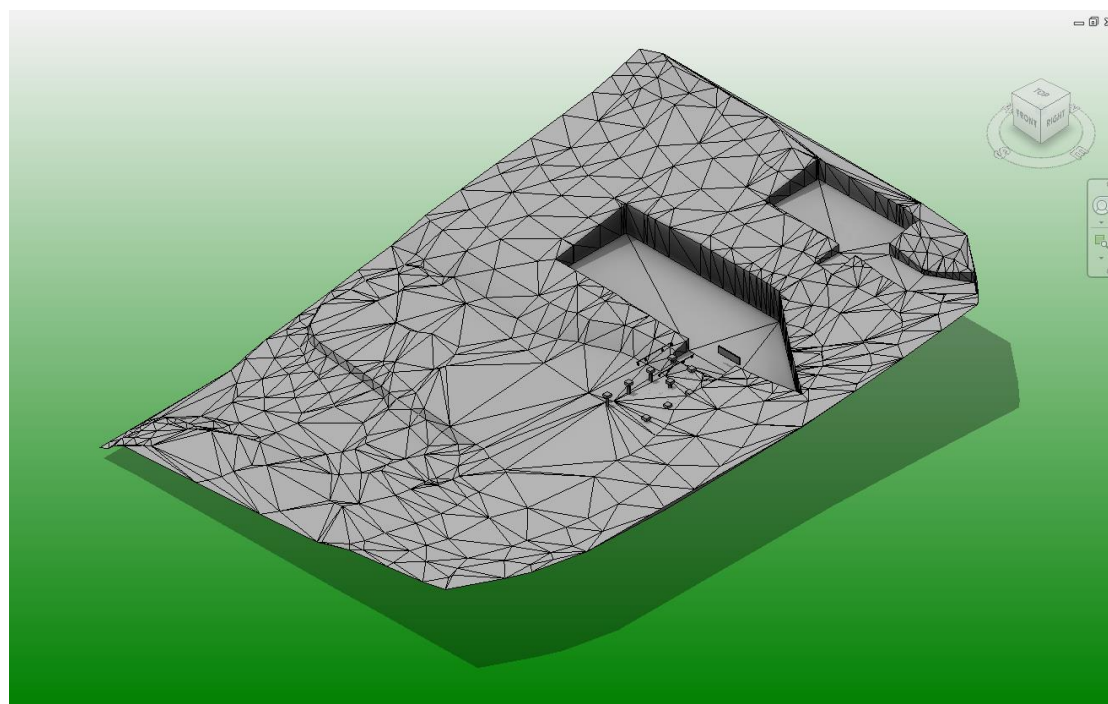


Image 11: IFC Linking template with different Phasing and Levels then the IFC file linked in

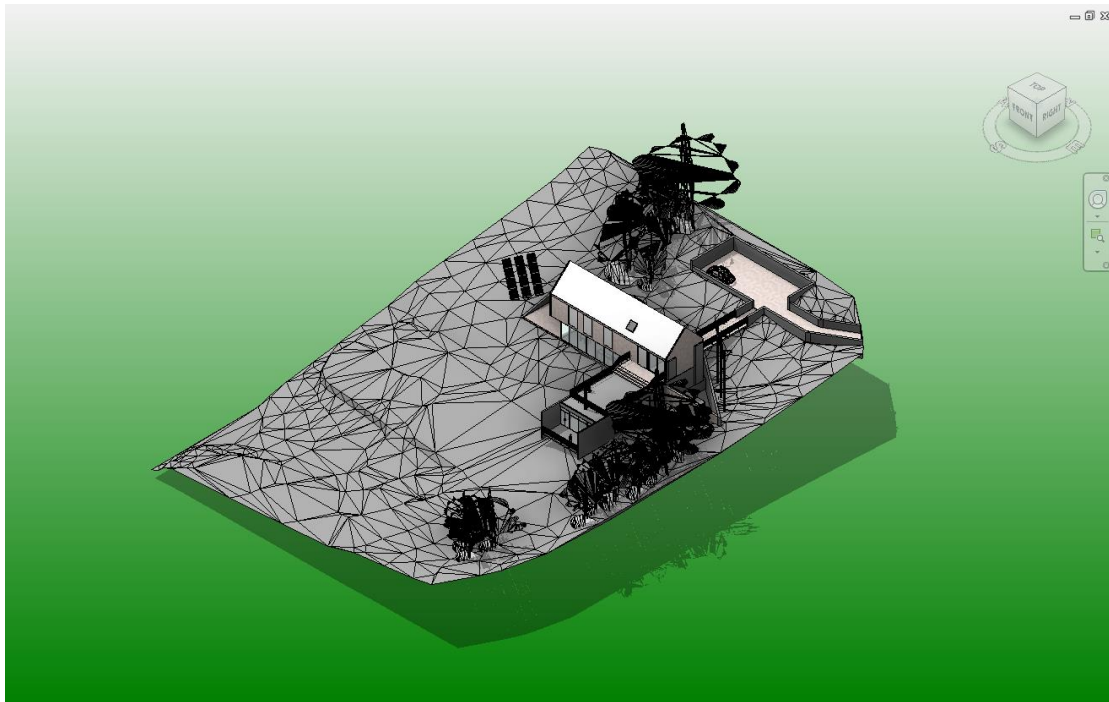


Image 12: Same Link with corresponding Levels and Phases

Revit Open Source IFC Exporter

Not only the BuildingSmart PR department needs to take a course in effective communications. The same goes for the Autodesk PR guys and gals. For an Autodesk department widely considered to be overly active when it comes to promoting questionable software enhancements as earthshaking improvements they have a huge blind spot for anything related to IFC.

With the release of Revit 2015 there was also a radical, ground breaking development with the Open Source IFC Exporter project. Because "Exporter" does not cover it anymore: for quite some time now the OS Exporter also contains the code for the IFC Importer. Autodesk has always been the only firm that not only talked the talk but also walked the walk and did not put their Exporter code in a propriety format but made it Open. And now they do the same for the Importer code without even so much as a single line of text in the 2015 product sheets...

So, in short, the Open Source Exporter Project now also holds the code for an Open Source Importer. Start coding people!

Export options vanilla Exporter

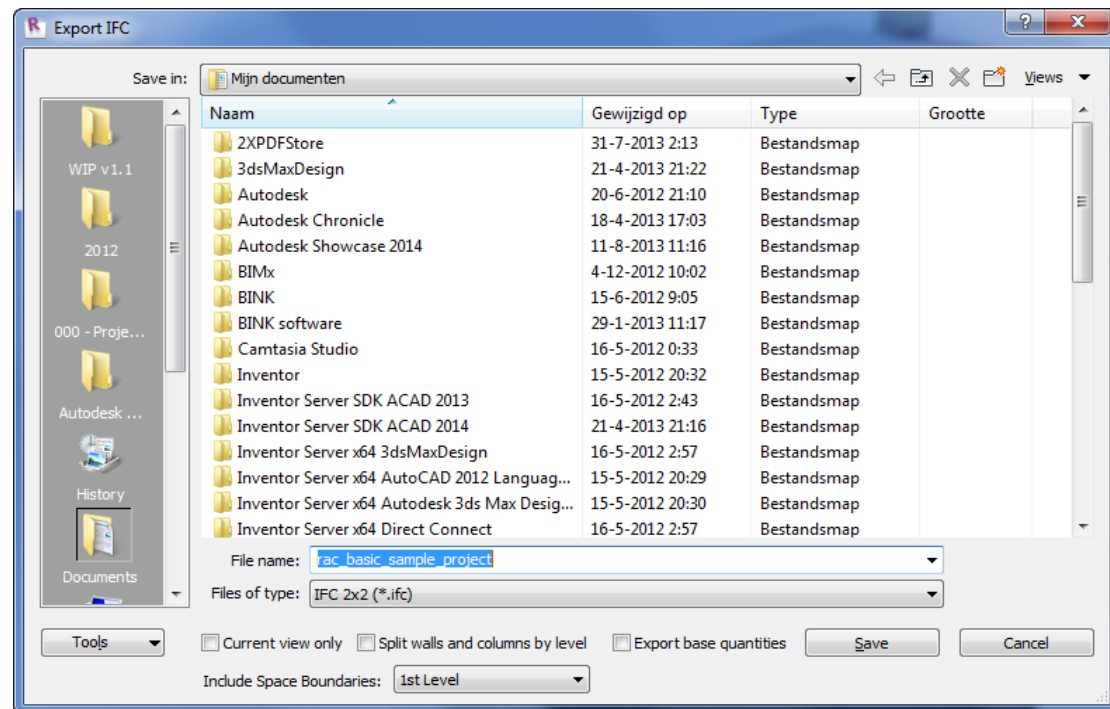


Image 13: Vanilla exporter from Revit2013

The regular exporter has a few basic options. These are also all present at the OS Exporter Alternate UI.

Current View Only

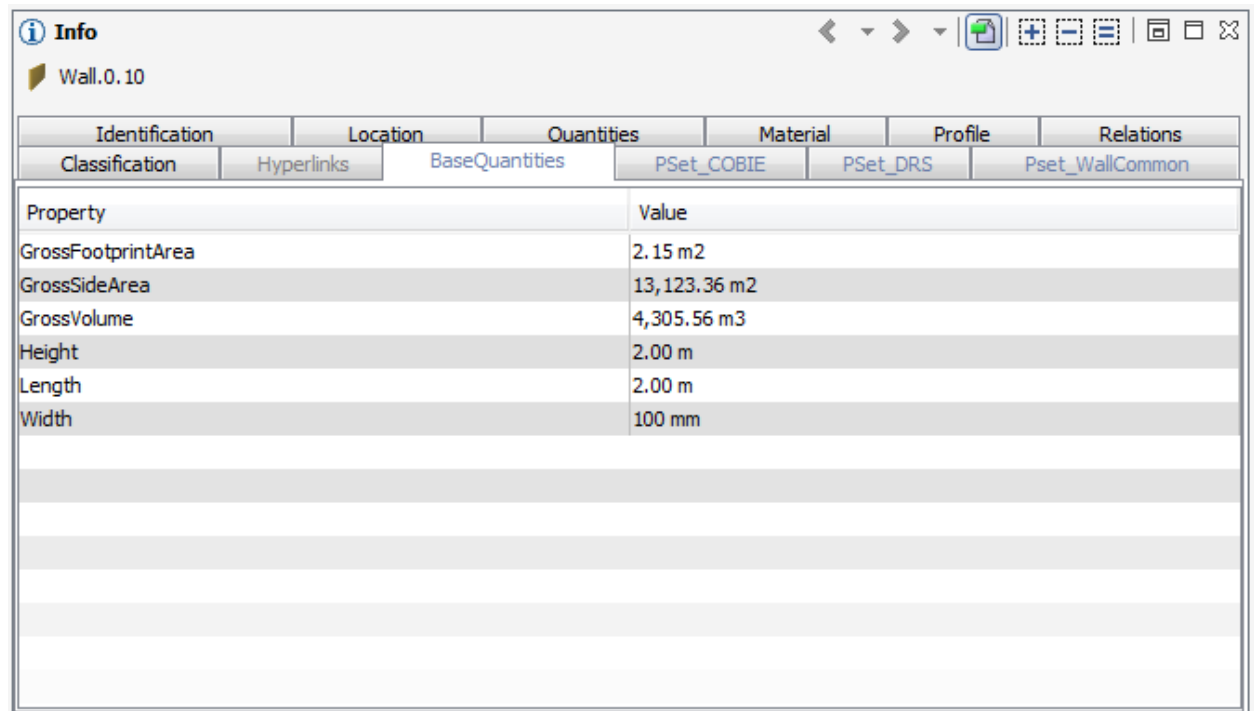
This option defines whether the entire model is exported, or only objects visible in the current view.

Split walls and columns by level:

Cut walls and columns that have a height of multiple levels into pieces by level. Keep in mind that this is an option with known issues in the past. It has some re-occurring problems with splitting elements with an offset to the Level they are hosted on.

Export Base Quantities

Export an IFC property set for certain categories (floors, walls) that contain quantity parameters such as Gross Area, based on measurements of the actual objects (and not parameter values inside Revit).



Property	Value
GrossFootprintArea	2.15 m2
GrossSideArea	13,123.36 m2
GrossVolume	4,305.56 m3
Height	2.00 m
Length	2.00 m
Width	100 mm

Image 14: Base Quantities

As you can see there's some weird stuff going on here (the wall I selected is 2x2m and 0.1m wide. This does NOT add up to 13.123m2 of GrossSideArea. Probably a conversion issue between metric and imperial that is filed as a bug and will be fixed soon.

Export Space Boundaries

This includes the relationships between Spaces and their Boundaries (e.g. surrounding Walls, Floors, Doors, Windows, and so on) in the export. There are 3 options:

None: Rooms/Spaces are exported as static components without any relation to their boundaries.

1st Level: Room/Spaces are exported with basic reference of their boundaries.

2nd Level: Rooms/Spaces are exported with full relationships to their boundaries, including their properties for materials, energy transmissions and such.

IMPORTANT NOTICE: 2nd Level Room Boundaries only works when you have a model suitable for energy analysis. In other words: you need

bounding objects that have this information present AND the proper project settings applied!!!

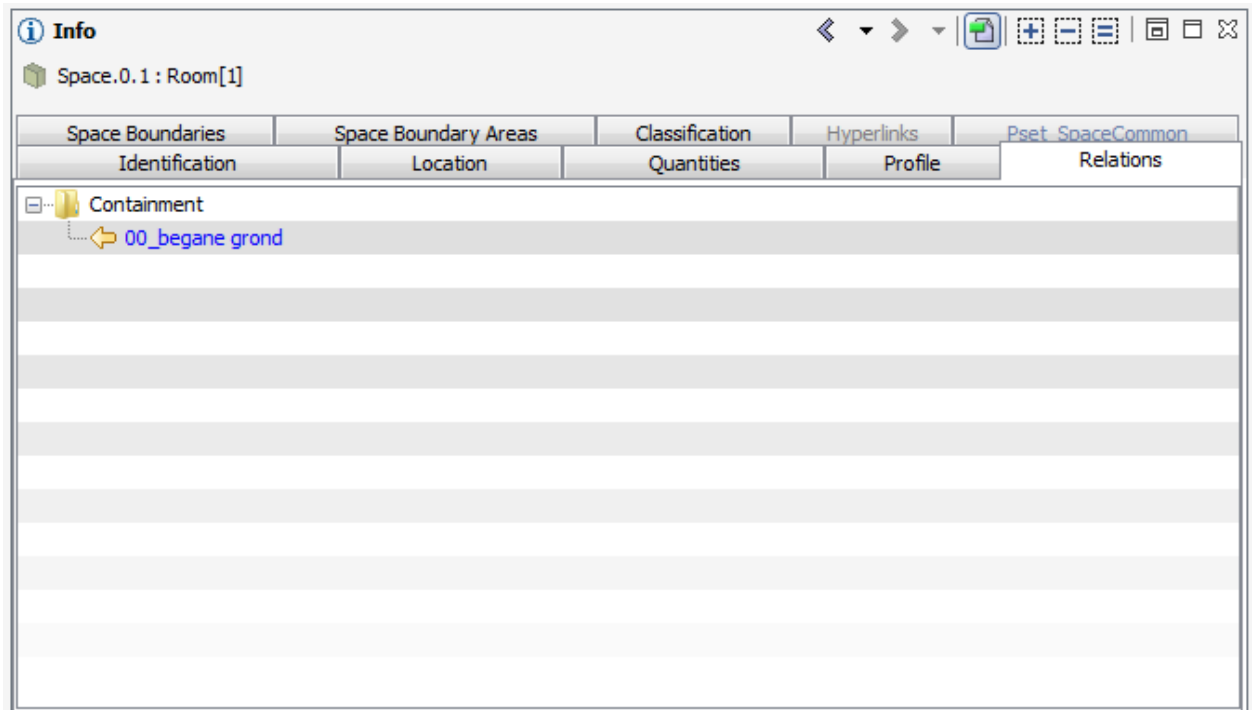


Image 15: No Space Boundaries

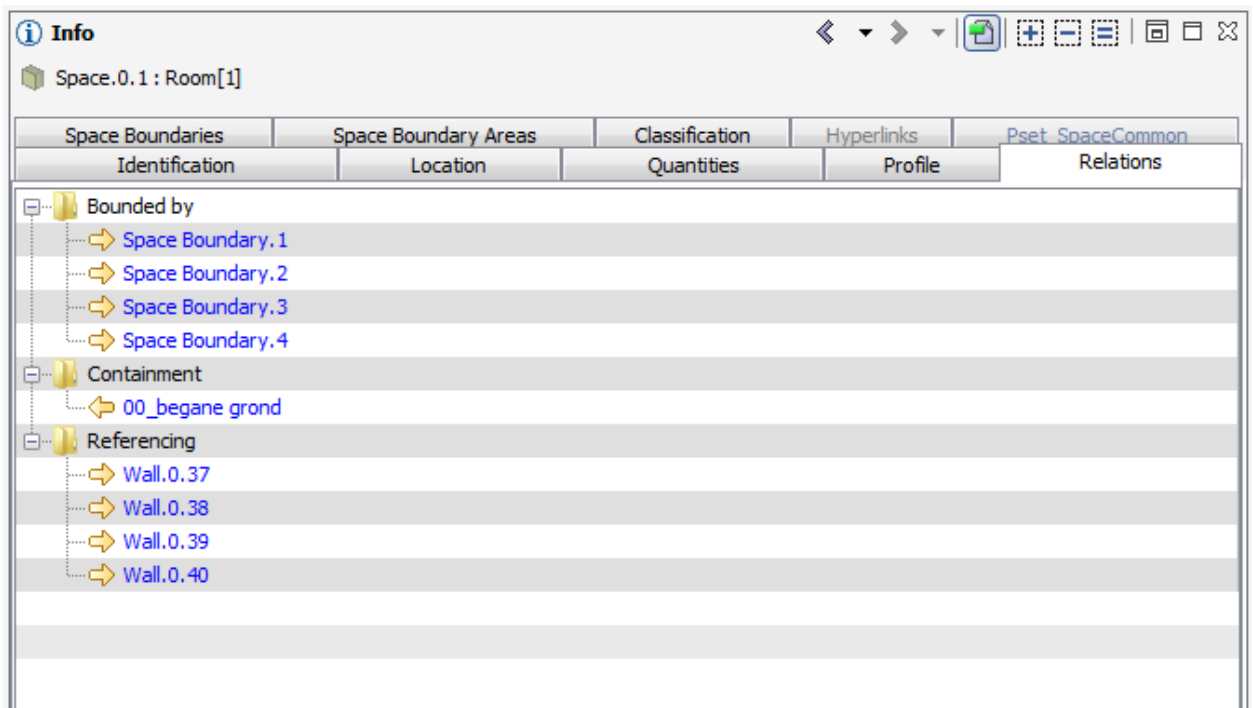


Image 16 1st Level Space Boundaries

Revised UI for the Open Source Exporter

Let's keep looking at the OS Exporter for a while. There have been some massive changes here since last year. A lot of options, checkboxes and stuff has been added and perhaps it's nice to go over the new stuff one by one.

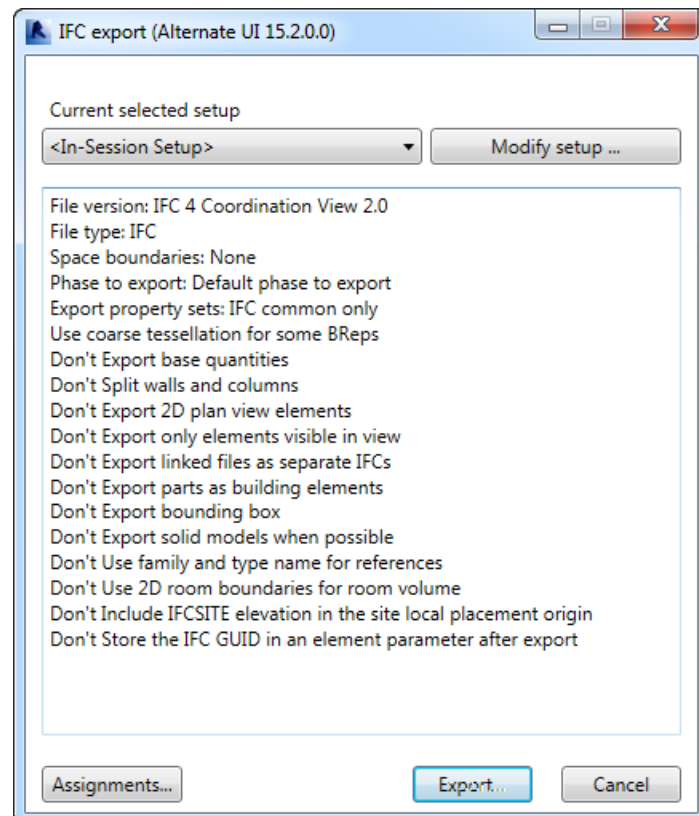


Image 17: Open Source Exporter UI, starting screen

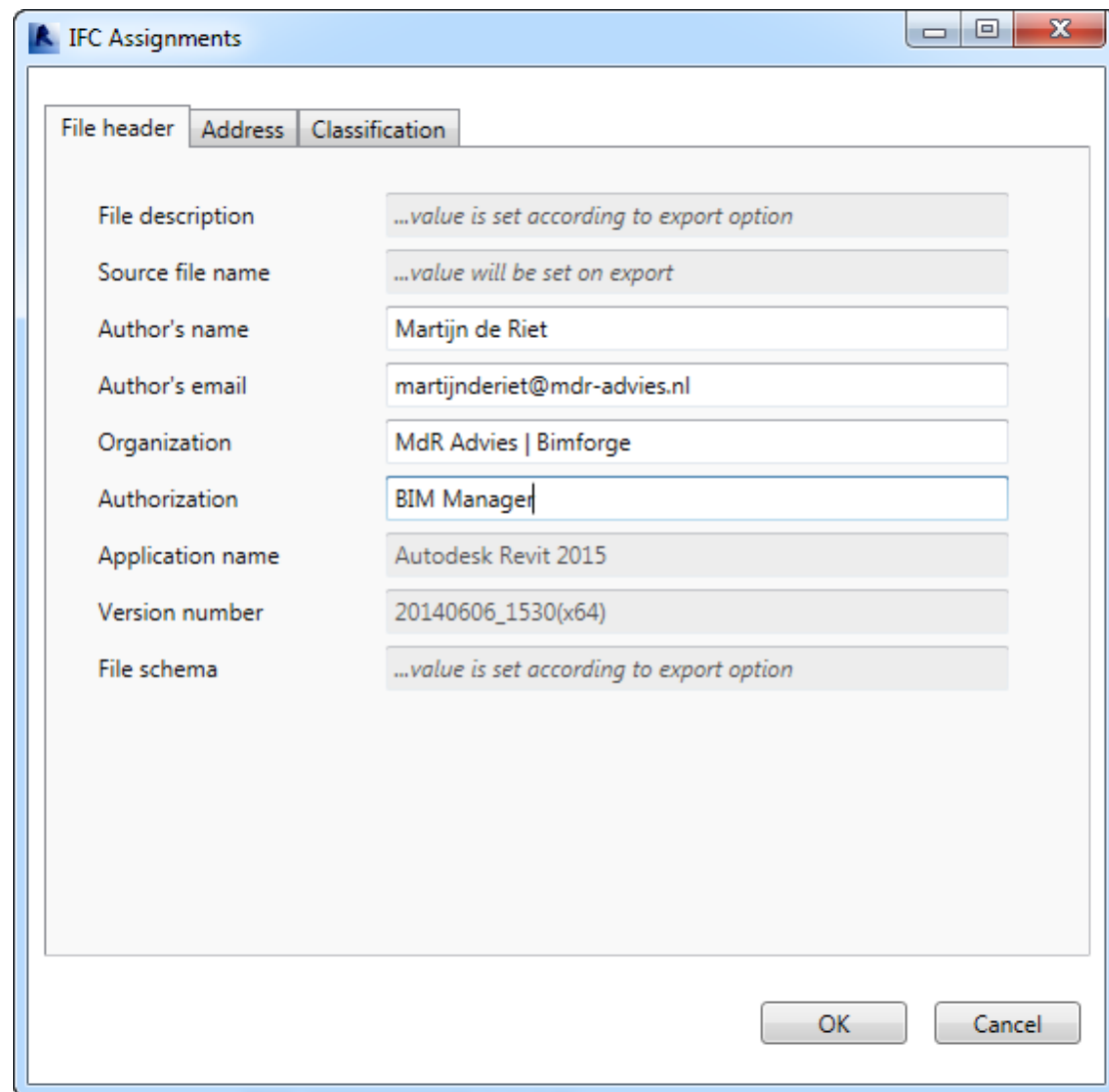
As shown the vanilla User Interface has a very limited amount of options and settings. Besides that, there's no way to keep your settings. All is different when using the Open Source Exporter in combination with the Alternate UI. Most settings won't mean anything to you, but we'll go through them one by one. A major thing to remember is that it's possible to save your settings in a custom selection set.

Assignments window

Let's start by looking at the Assignments button, down to the left. Pushing this opens up a new (tabbed) screen.

File Header

The first tab, File Header, is pretty much self-explanatory. It deals with stuff that goes into the IFC File Header.



The screenshot shows the 'IFC Assignments' dialog box with the 'File header' tab selected. The dialog has three tabs: 'File header', 'Address', and 'Classification'. The 'File header' tab contains the following fields:

Field	Value
File description	...value is set according to export option
Source file name	...value will be set on export
Author's name	Martijn de Riet
Author's email	martijnderiet@mdr-advies.nl
Organization	MdR Advies Bimforge
Authorization	BIM Manager
Application name	Autodesk Revit 2015
Version number	20140606_1530(x64)
File schema	...value is set according to export option

At the bottom right of the dialog are 'OK' and 'Cancel' buttons.

Image 18: Properties to add to the File Header.

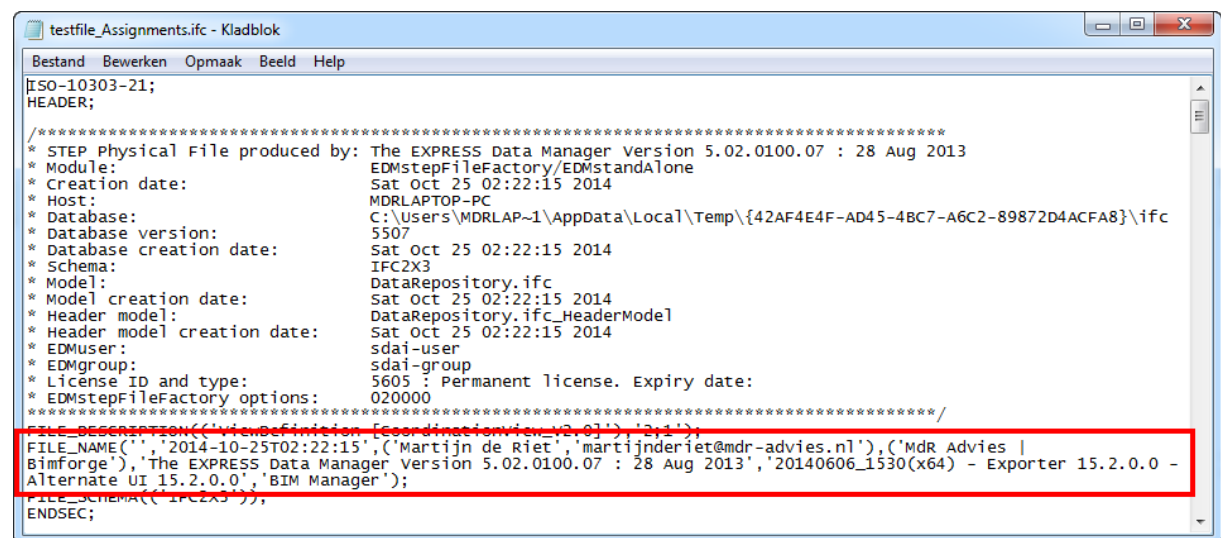


Image 19: Parameters and values added to the IFC

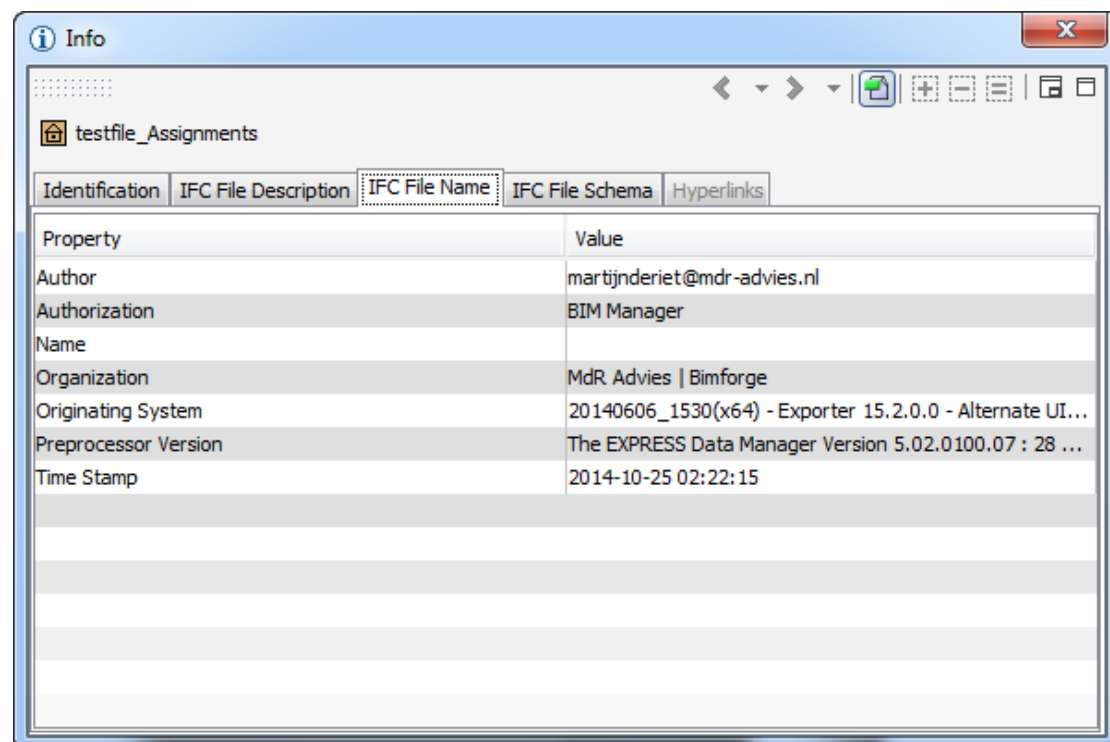


Image 20: And present in the Viewer

Address

In this tab you can enter the necessary Project Information. The checkbox "Update Project Information" copies this data into the Project Information.

The screenshot shows the 'IFC Assignments' dialog box with the 'Address' tab selected. The dialog has three tabs: 'File header', 'Address', and 'Classification'. The 'Address' tab contains the following fields and values:

Field	Value
Purpose	User Defined
Description	Office location
User-defined purpose	Only when "Purpose" is set to "User Defined"
Address line 1	De Papiermaker 22
Address line 2	No Adress Line 2
P. O. Box	Also no P.O. Box
City	Boxtel
State	Noord Brabant
Postal code	5283 ZS
Country	Nederland
Internal location	Leave it at the front door

At the bottom of the dialog, there is a checkbox labeled 'Update project information' which is checked. This checkbox is highlighted with a red rectangle. Below the checkbox is a dashed line. At the bottom right of the dialog are 'OK' and 'Cancel' buttons.

Image 21: Project Information in the Address tab...

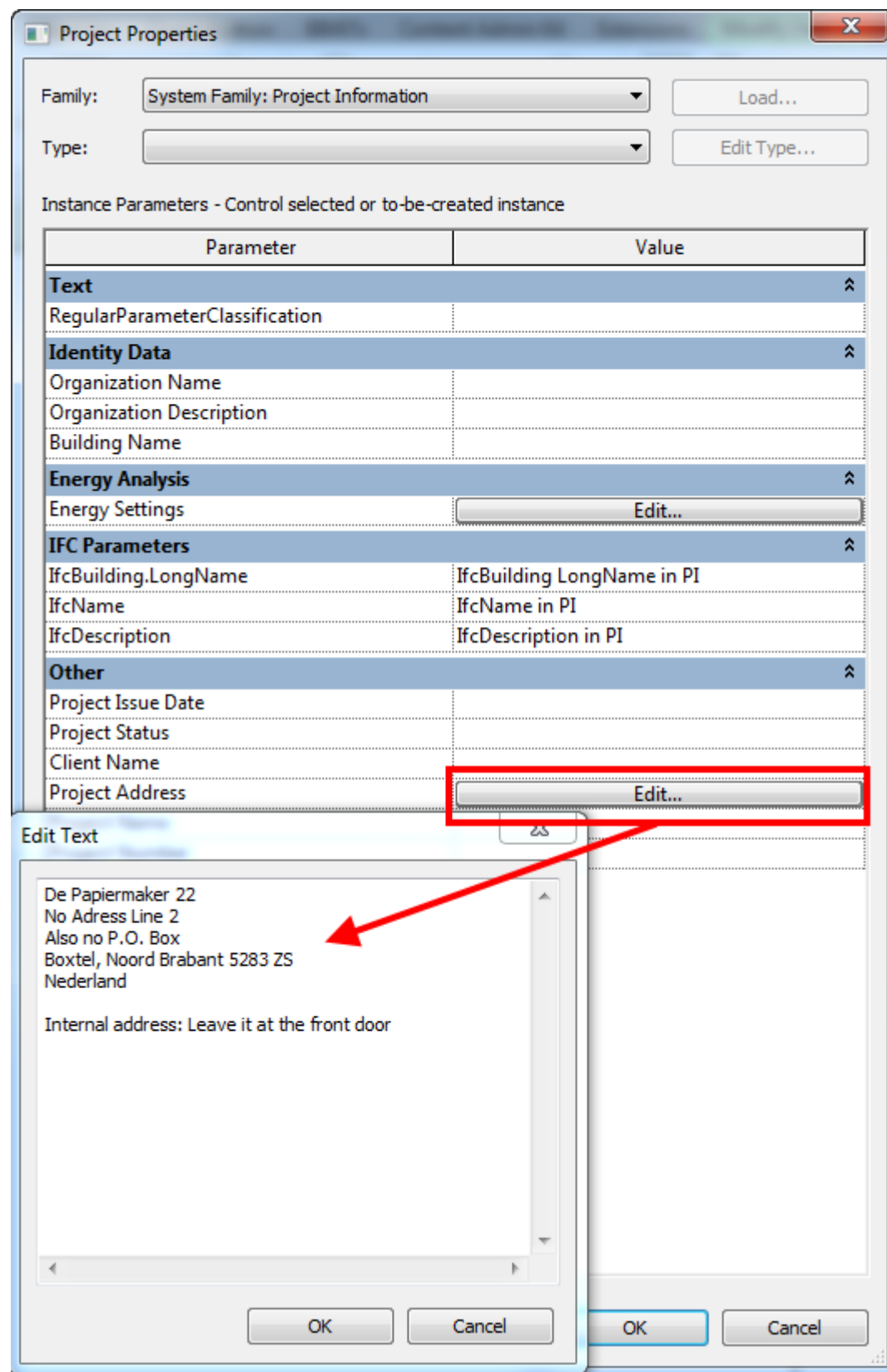


Image 22: Gets copied into the Project Address...

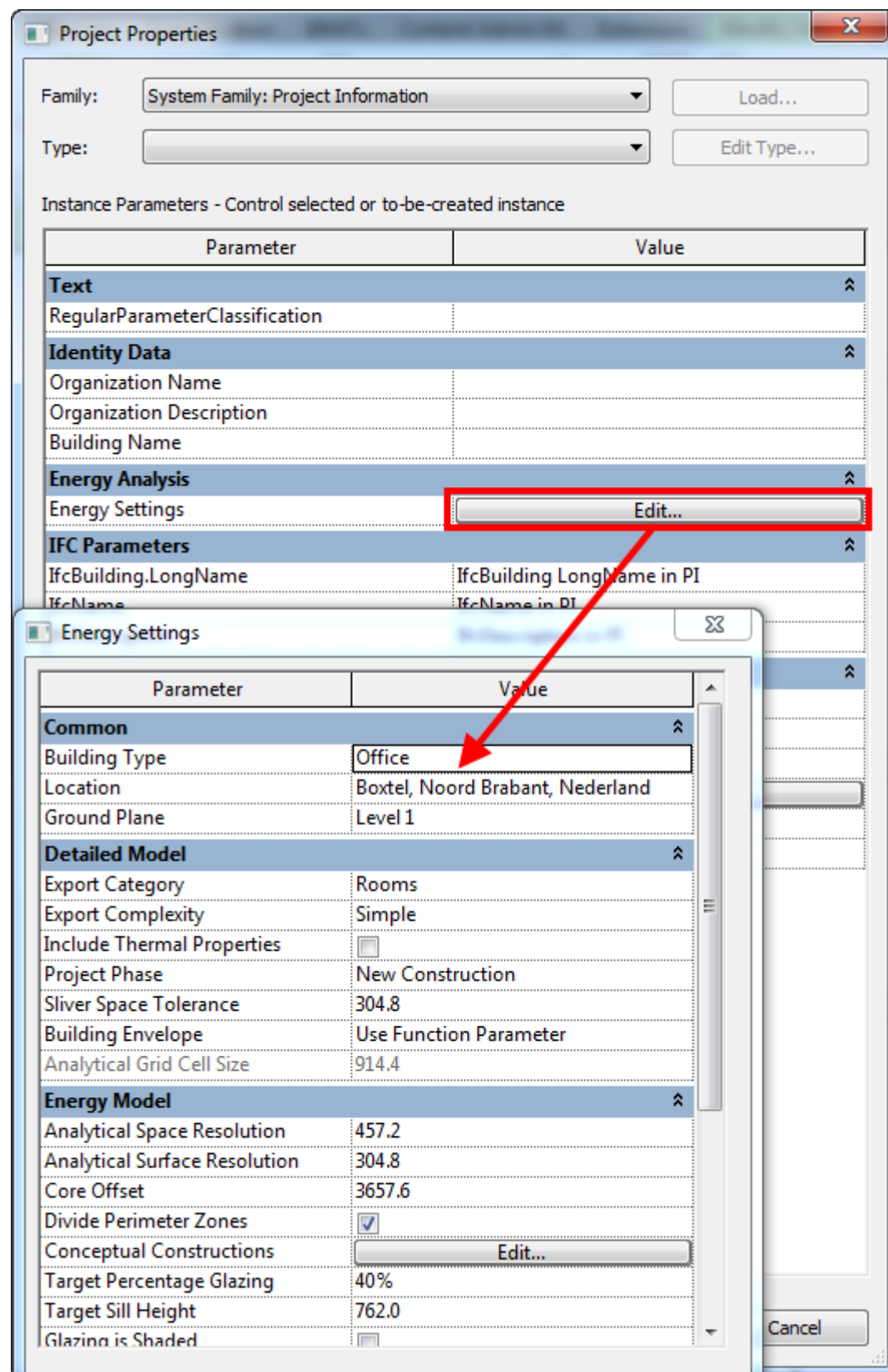


Image 23: ... And into Energy Settings...

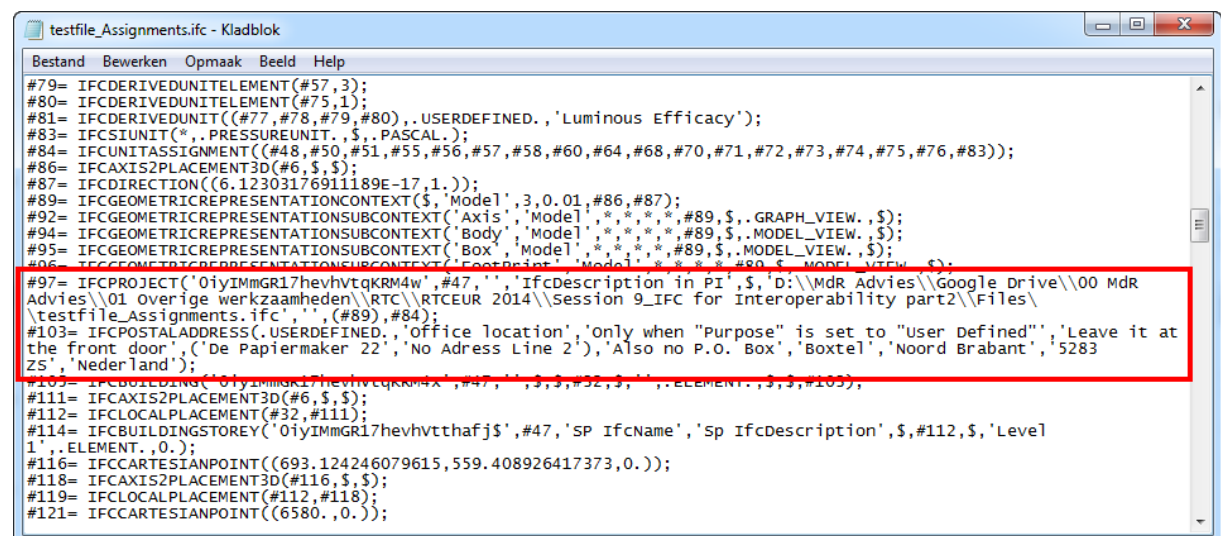


Image 24: ...Last but not least, the IFC itself

Classification

The Classification tab is a nice one. It allows you to specify the classification system used in the project.

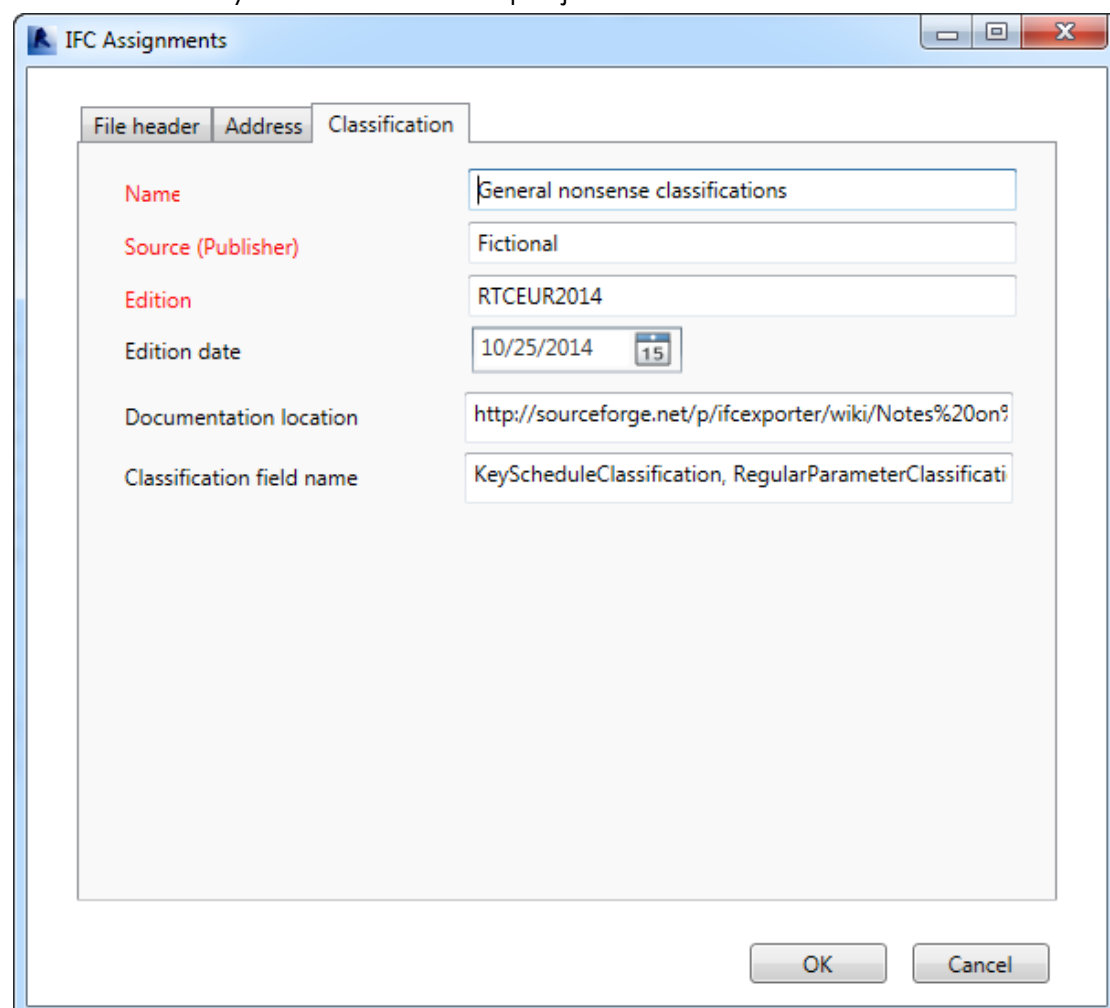


Image 25: Classification tab

Let's say you want to name and define your Rooms through a certain classification system. And you want to export this data with the Rooms. There's two ways to do this: through "regular" parameters (either Project, Family or Shared) or through a Key Schedule.

Both options require 2 things:

1. A (text) parameter
2. A parameter value with the following syntax:
[parameter name]parameter value;parameter description

In this example I have 2 parameter names:

1. KeyScheduleClassification for the Classification parameter defined in the Room Key Schedule.
2. RegularParameterClassification for the Classification parameter defined as a Project Parameter.

Both are assigned to Rooms and filled out for the two Rooms in the project.

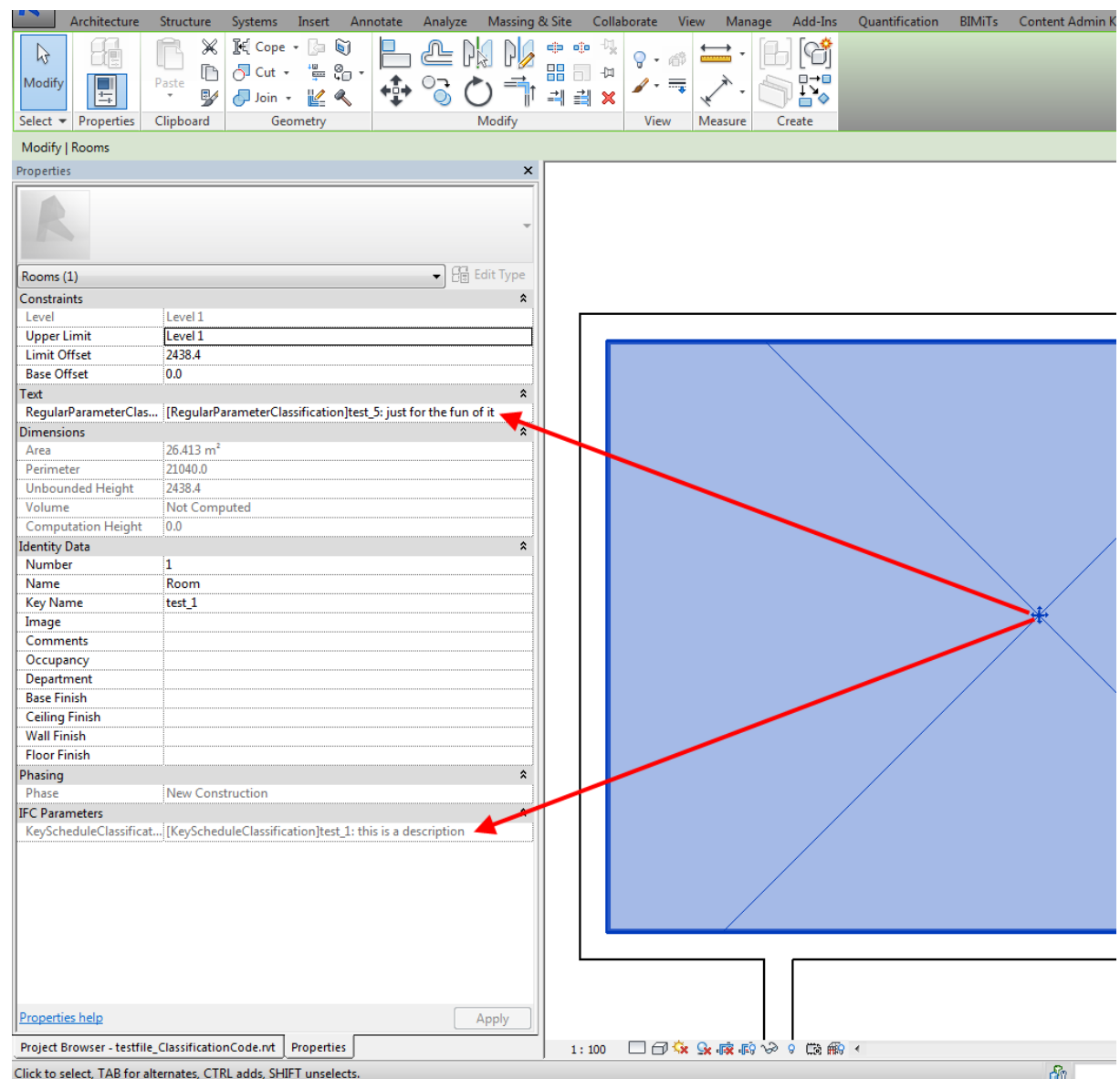


Image 26: Setup in Revit

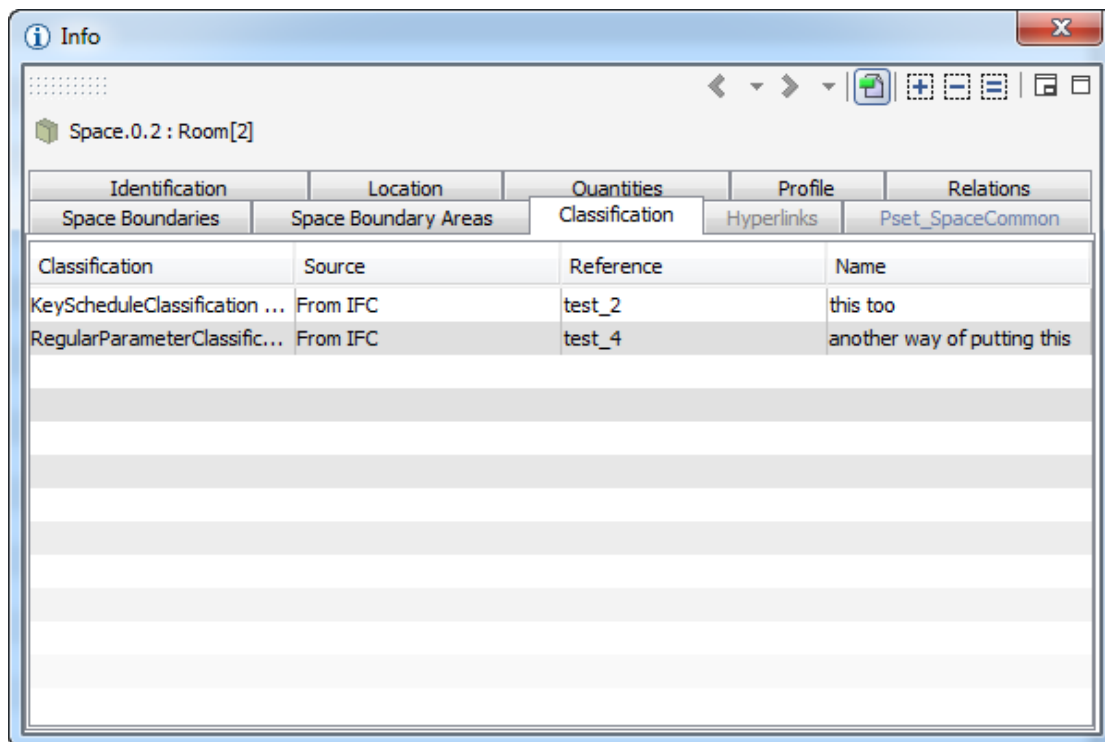


Image 27: Result in IFC Viewer...

As you can see I didn't get the Source to Export, although upon examining the IFC file in Notepad I noticed that the value does get exported from Revit. At this point I'm not sure whether it's a syntax error in the Export, or that this is expected behavior and the "Source" in the IFC Viewer is not the same thing as in Revit.

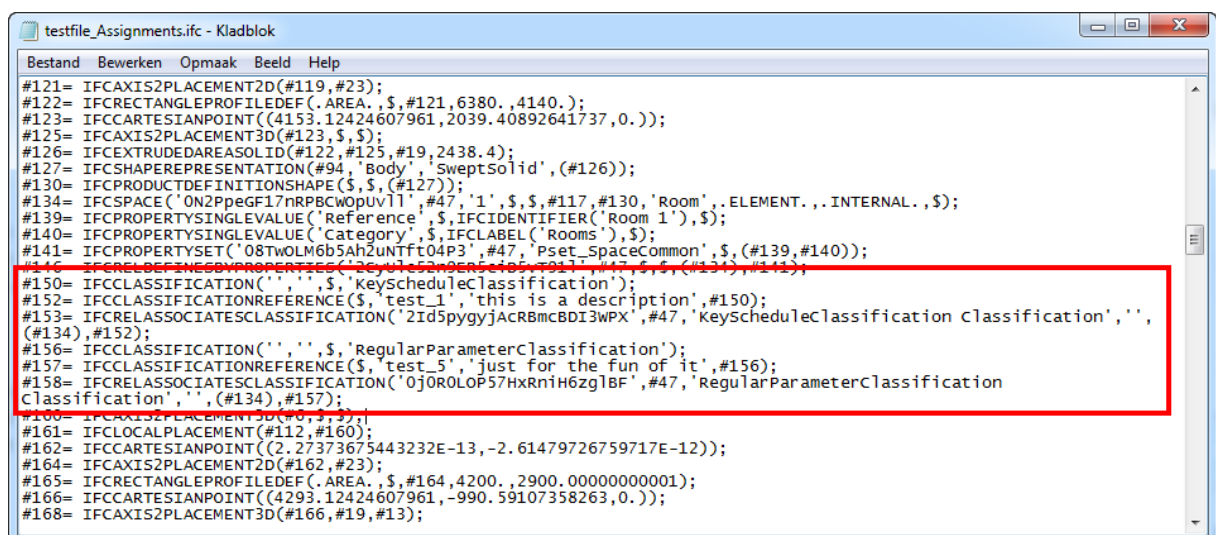


Image 28: ... And in the code itself

Modify Setup window

The second screen that handles the Export setup is the “Modify Setup” screen. In the latest version of the OS Exporter, this screen is separated into 3 tabs:

- General
- Export Property Sets
- Advanced.

General Tab

First and most important option is that you can use that to specify (and save!) your desired Export settings.

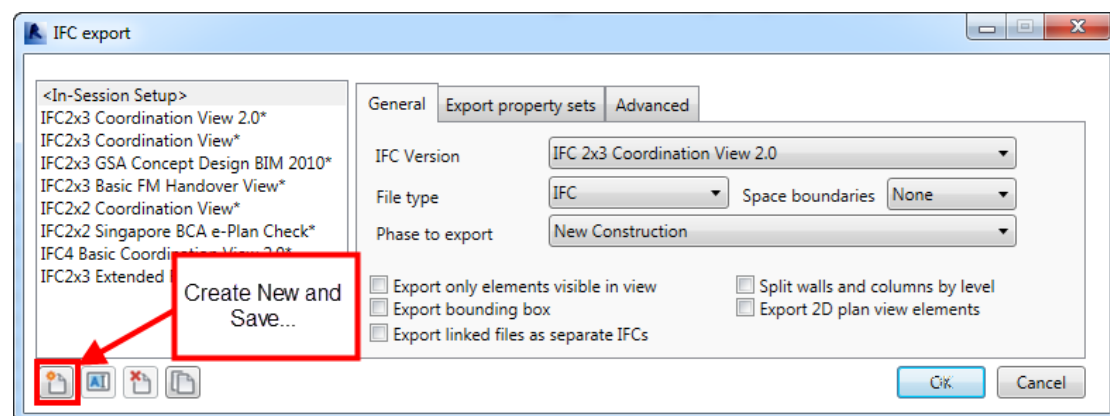


Image 29: The “Modify Setup, General” screen.

Let's look at all the checkboxes one by one... I'll leave out the ones that also exist in the vanilla exporter.

File Type

If I may suggest something: use IFCZip when you're not obliged to a preset Definition. This does exactly as suggested and zips the IFC which cuts down file size drastically. Most viewers don't need you to unzip it anyway...

Phase to export

Pick a Phase you want to export (don't leave it to “Default Phase to Export”, this will cause undesired results)

Export Bounding Box

Think about the Course vs Medium vs Fine detail level in Revit. Lots of people use Coarse representation to model and show a simple box

instead of a complicated window lining profile. This is somewhat the same: the ability to export a simple geometric shape instead of more profiles.

Not quite sure how these are created and if this works for everything though. That would require some testing. And to be honest, my opinion on this is about the same as on the Coarse-Medium-Fine debate: don't really care... So it's not that high on my to-do list.

Export Linked Files as separate IFC's

The name kind of gives it away: checking this box means all Linked Files get exported to separate IFC's. How to re-unite them in a single IFC is an interesting problem we will be exploring later on.

Export 2D plan view elements

This is an obvious one; it exports 2D components from Plan View. Keep in mind though: you'll need to set the Mapping Tables accordingly! (They default to not export 2D elements).

Export property sets

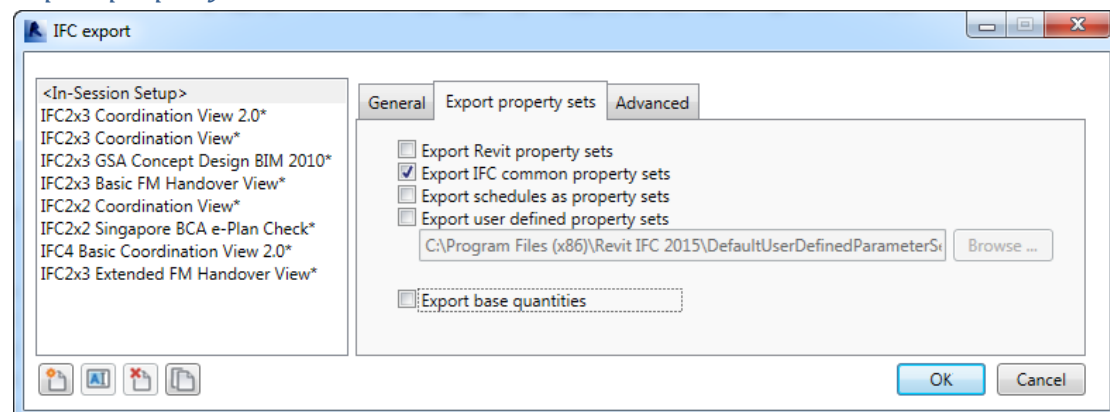


Image 30: Export Property Sets tab

Export internal Revit Property Sets

Seriously people: there's really no reason to use this one.

Export IFC Common property sets

All IFC Classes have common property sets (IFC version of a hardcoded set of parameters). Use this checkbox to export that and map certain hardcoded Revit parameters to them.

Export Schedules as Property Sets

Cool feature: export your schedules as Property Sets. This is sort of an in-between for the “Export Revit Internal Property Sets”-option. While that is a package deal (all or nothing) this option lets you define parameters you want to export based on schedules available. Few things:

- It will export ALL schedules present in the project.
- It will work on every parameter in the schedule, Shared or project, even Calculated Values.

Export user defined Property Sets

Even cooler feature: you can now create a Property Set that exactly defines which parameters should be exported and how they should be named in the resulting IFC.

We'll get to this option in more detail later.

Advanced

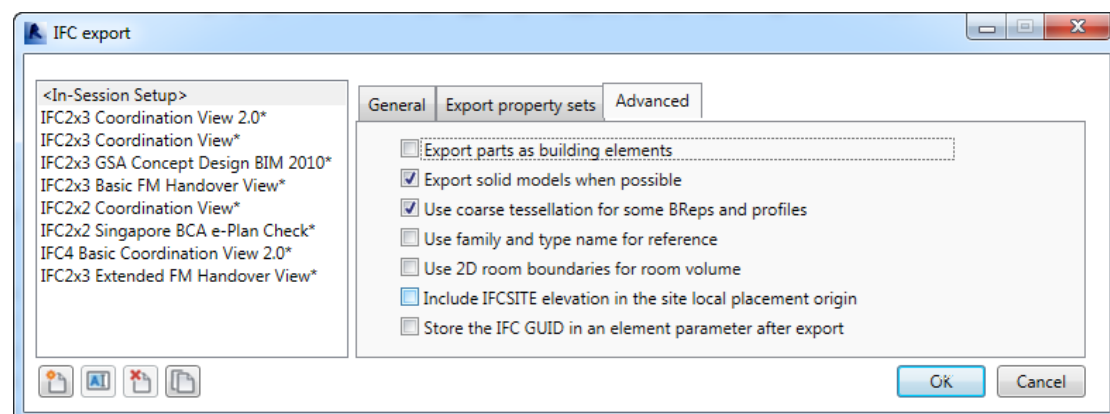


Image 31: Advanced Tab

Export parts as building elements

If you have Parts in your model, this checkbox will export those as separate elements with no relationship to the original object. If you want to export Parts as a sub-element of the encompassing object, check this off.

So if you have a wall split into parts, This is the right IFC:

```
IFCWALL
  IFCBUILDINGELEMENTPART
  IFCBUILDINGELEMENTPART
  IFCBUILDINGELEMENTPART
  ...
```

This is what you'll get with this checkbox on:

```
IFCWALL
IFCWALL
```

IFCWALL

However, for both options you'll need to have Parts turned on in the view you're exporting from (and it only works with "Export only elements visible in view").

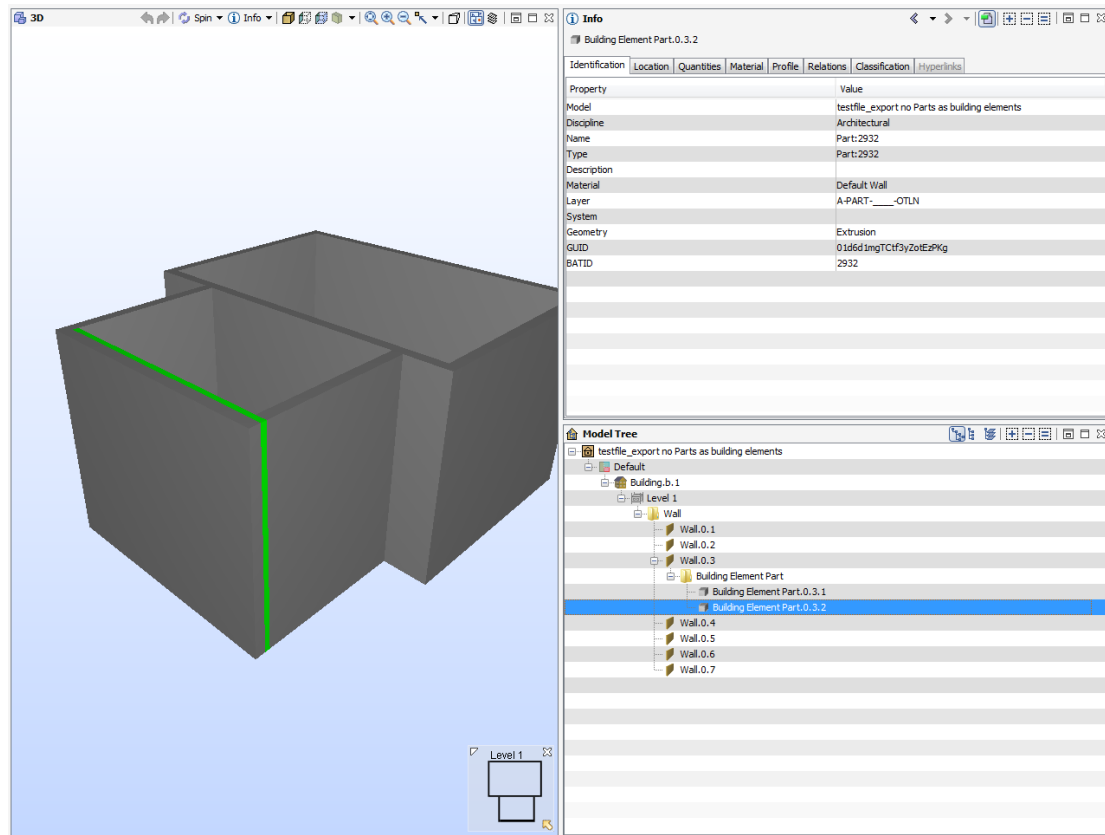


Image 32: Export Parts as Building Elements checked OFF (with Parts visible)

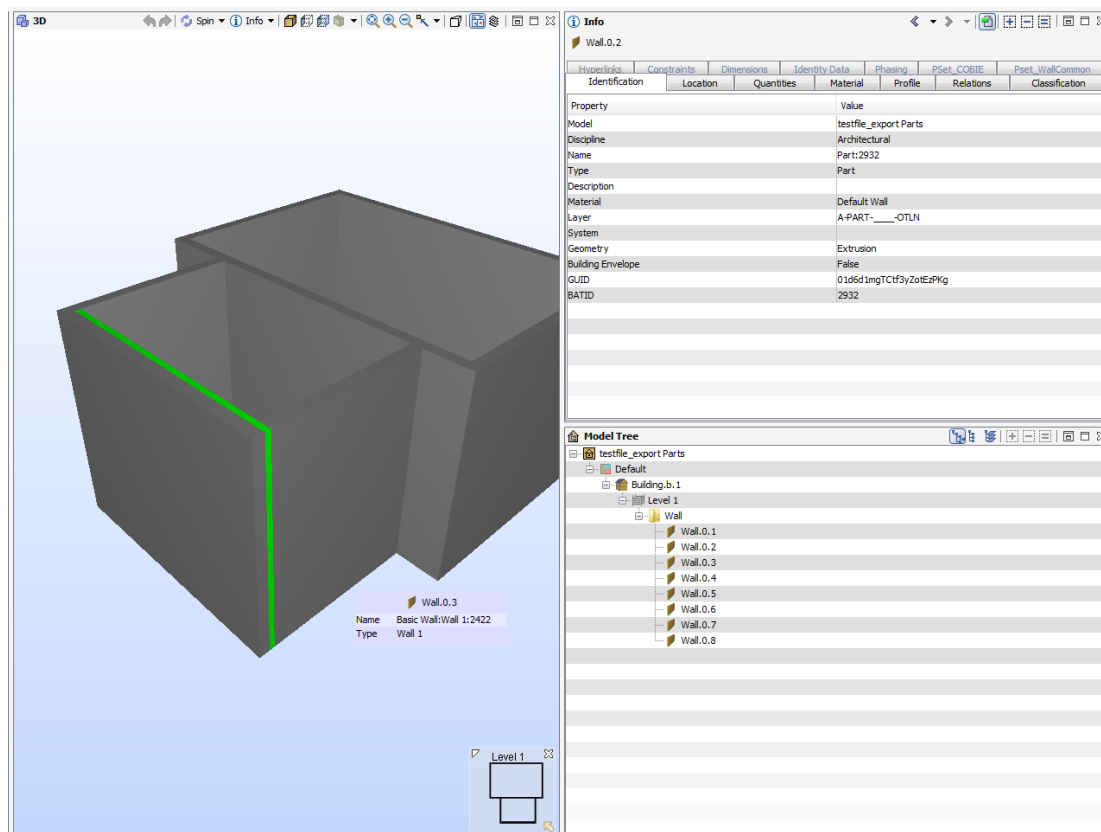


Image 33: Export Parts as Building Elements checked ON (with Parts visible)

Export Solid Models when possible

This requires some background (read: boring) stuff. Basically Revit can export any given geometry in two ways:

1. Extruded solids. Same thing as in Revit. Simply draw the base and extrude to a certain height.
2. Boundary Representation. I'm going to quote Angel Velez, IFC developer for Autodesk, on this one, because he has a great way of explaining this:

For an arbitrary, nicely defined Boundary Representation, you would start with defining 8 vertices (one for each point of the cube). Next, you'd define 12 edges, which would be curves of some sort (in the case above, lines) that would refer to the vertices (so any given vertex would be referenced by 3 edges for the cube). Then, you'd define 6 faces, which would be defined by a series of the 12 edges above, connected nicely. This is called the topological representation of the cube. It isn't bad, in that you have some shared information, but it isn't

as nice as an extrusion or a revolved solid, that has extra parametric information associated with it.

That's not what's in IFC2x3. What is in IFC 2x3 is a dumbed down version of that called a faceted BRep. In this case, all edges are lines, faces are polygons, and while edges can share vertices, they don't have to, and edges aren't shared either. You could make the nice thing I said above, but you have to reconstruct all of the topology. The issues with IFC come from the "faceted" part. A nice sphere can be very easily parametrically defined, but there is no way to exactly create a sphere out of facets. So depending on how hard you try, you might get a cube, or your 100mm radius sphere might be approximated by 1mm x 1mm x 1mm triangles. As you can imagine, the range of possibilities to get this right – or wrong – are staggering. Now, to put it in terms anyone can understand, below is an image of a faceted BRep.

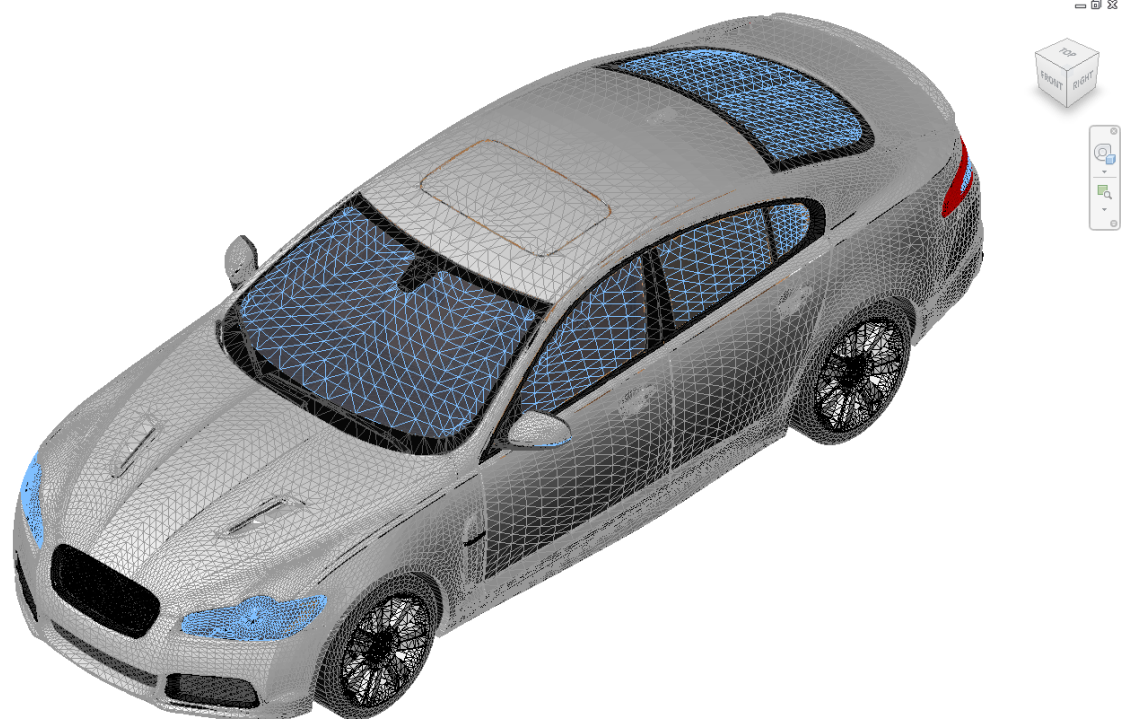


Image 34: Faceted BRep

See all those tiny triangles?

Those are faceted BReps... The default exporter uses either one: Solids OR BReps. If the entire entity cannot be exported as a Solid, it's exported as a BRep. Which, as you can imagine, is often the case (especially with curved forms).

Checking this option means the OS Exporter will always use Solids for those parts that can be defined as such. Only the parts that cannot be defined as a Solid will be exported as a BRep.

Use coarse tessellation for some BReps

The example image on the previous page has a rather coarse tessellation. But creating IFC's with any IFC Exporter using BReps can come with some seriously heavy results.

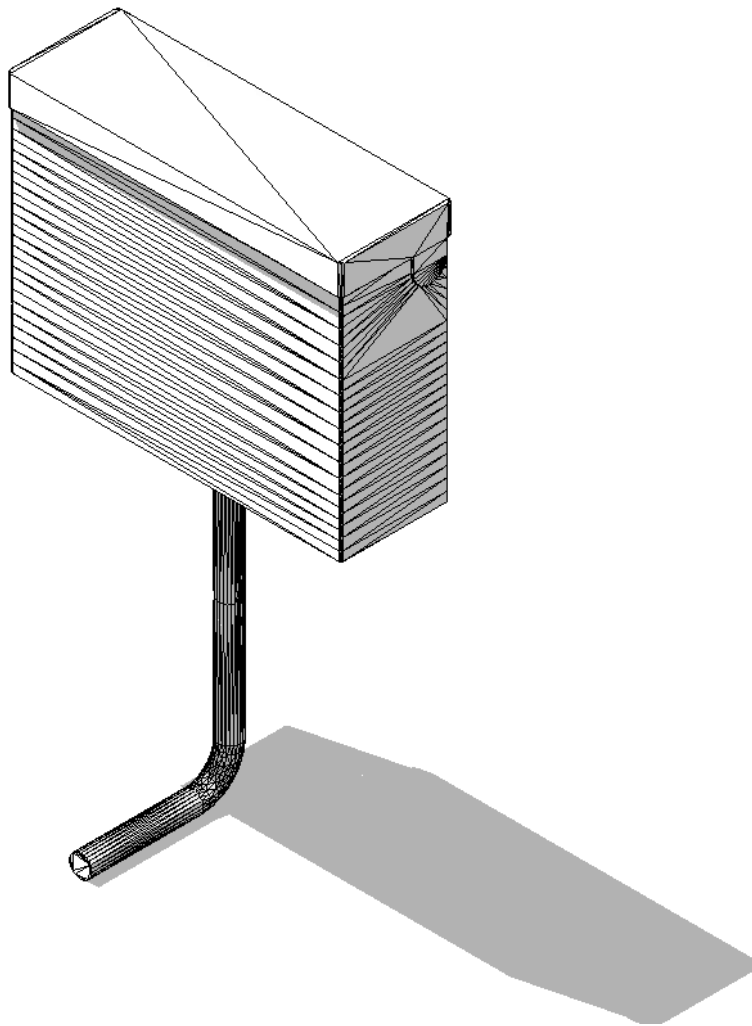


Image 35: Faceted BRep Cistern

The above Cistern in IFC format from the NBL-website is 2640 lines of code. Which actually isn't that bad because the Revit file exported to IFC is a 1000 more lines.

There's two problems here:

1. IFC is different than Revit. Every instance is defined separately. Which means that two of these mean twice the amount of code and twice

the filesize.

2. The frustrating thing about this is that it can be different. Jon Mirtschin from GeometryGym wrote a blog about IFC manufacturer content (find it here: <http://geometrygym.blogspot.nl/2013/10/parametric-ifc.html>) where he proved that with a little smarter implementation, the filesize can be brought down with 80% and the amount of code to 360 lines.

I put the testfiles in the dataset.

Use family and type name for reference

All IFC objects have a "Reference" parameter (sort of a Type Mark and Mark in one). This lets you combine Family and Type Name into that Reference parameter.

Use 2D room boundaries for room volume

Checking this box means Revit will treat a Room Separation Line as a spatial element with a height and area when calculation the Room boundary area.

Include IFCSITE elevation in the site local placement origin

Again, Angel Velez from Autodesk:

A site has a Z offset. By IFC CV 2.0 specification, the Z offset should be included in the specification of the IFCSITE, NOT in its local placement origin. Doing both is "double counting" and will result in the site being 2x as high as it should be. However, most existing applications don't follow the CV2.0 standard, and so doing the right thing generates the wrong picture. So check this if you want bad IFC files that behave correctly.

Store IFC GUID in file after export

Just like all Revit components, their IFC counterparts have a unique identification code, the IFC GUID. Checking this option results in the creation of an instance parameter in all exported objects called <IfcGuid> and writing the value to it.

Why is this important?

Well, let's put it this way: Revit, although it now has basically the same functionality available for export as any other software, has a bad name when it comes to IFC handling. So whenever there is trouble, it's usually condemned to be "Revit's fault". I can tell you right here and now: it's usually not.

From Revit 2015 onwards, the IfcGUID parameter is scheduable. Which means you can create a schedule that filters on IfcGUIDs being non-existent after exporting to IFC. When there's a GUID, the item is created in IFC and you can use the GUID to find it in the text file. If it's there, it's not your fault the other guys software can't see it...

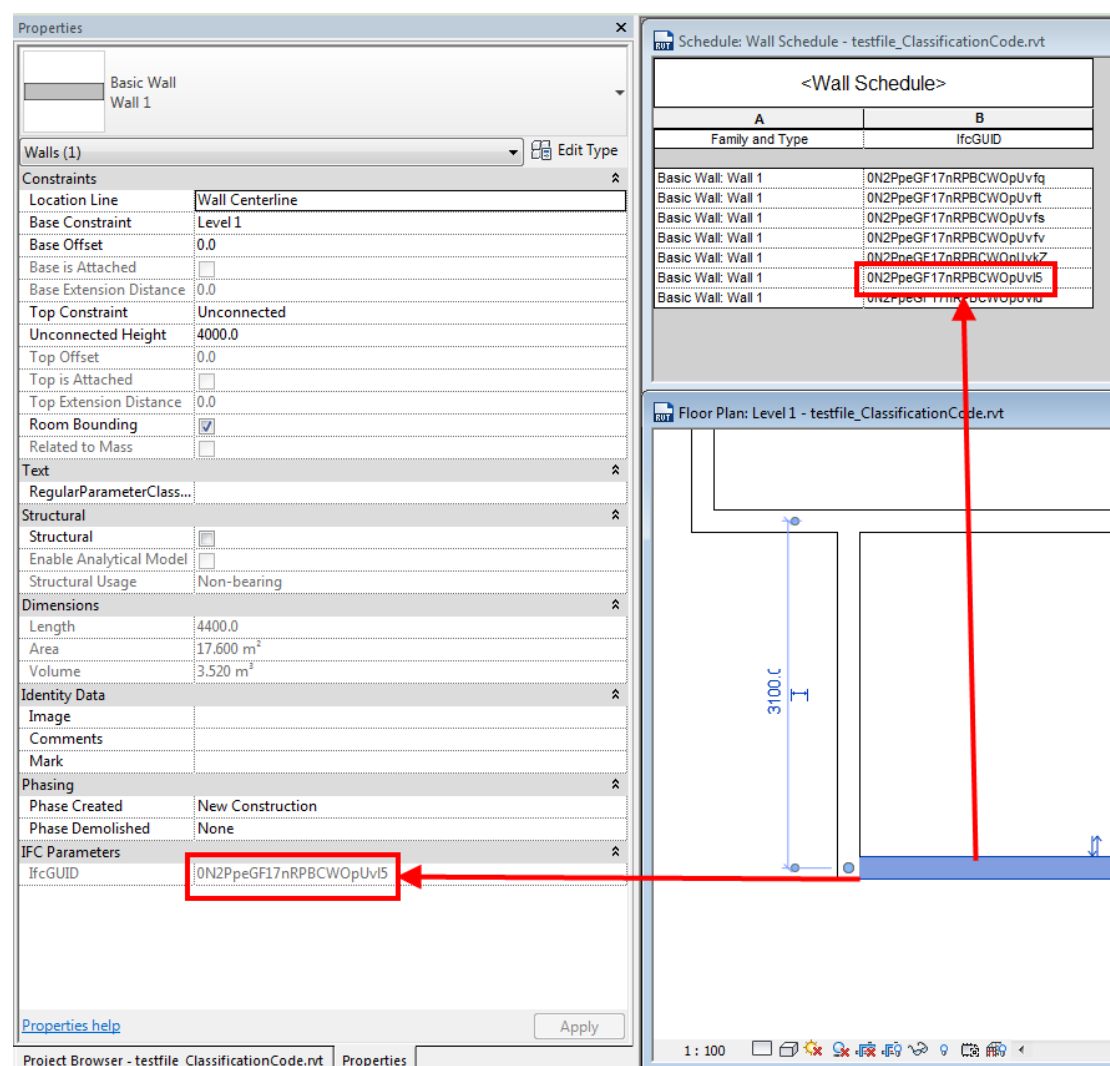


Image 36: IfcGUID in Revit



Custom Property Sets

Like I said, we're not discussing the ParameterMappingTable. That ship has sailed... I did add a downloadlink to last year's handout (and the one from AU2013) that handle that subject thoroughly. In this class we'll be focussing on creating custom Property Sets.

Basic rules

There are a few basic rules to adding custom Properties to an IFC though:

1. You can't add parameters to an "official" Property set.
2. A custom Property Set cannot have the prefix "PSet_". That's a prerogative reserved for the Real Deal. Although, to be honest, nobody is going to yell at you if you do...
3. A parameter will only be added to IFC when it has a value. Empty parameters cannot be added on export from Revit. They can be added in external software but, as far as I can tell, that's not how it's supposed to work.
4. Relating to item nr.3, you can't have an empty Property Set either. An empty Property Set simply won't be created.
5. Property Sets will take into account the IFC model hierarchy. So if you add a Property Set to the "IfcElement" Class, it will get handed down into each and every child of that Class. Which is kind of need, cause that means you can have global PSets.
6. Custom Property Sets are NOT sensitive to Caps when it comes to mapping parameters.
7. The Custom Property Set definition file works top to bottom.

How do you get it to work?

Well, just like the ParameterMappingTable it's a simple textfile. The required syntax is quite simple:

```
#
PropertySet:      <Pset Name>      I[instance]/T[type] <element list separated by ', '>
<Property Name 1>      <Data type>      <[opt] Revit parameter name, if different from IFC>
```

And here's a real life example:

```
PropertySet:      PropertySet_RTCEUR2014_I I      IfcElement
                  TestElementInstance      Text      Test_Element_Instance
#
PropertySet:      PropertySet_RTCEUR2014_T T      IfcElementType
                  TestElementType      Text      Test_Element_Type
#
PropertySet:      PropertySet_RTCEUR2014_WA_I I      IfcWall
                  WallSpecificParameter      Text      Wall_Specific_Parameter
```

So that's it. Feels kind of like an anti-climax after all those time nagging about getting the ability to export any kind of parameter I want and only the ones I want.

Exporting Linked Files

Up until recently it wasn't possible to export the Linked Files contained in a Host File, other than opening the Links one by one and exporting them from there. This posed a few problems. Any of the following situations meant you had to bind all links before export:

1. Links moved around in the Host File
2. Multiple instances of a Link in the Host File
3. Links rotated in the Host File.

The only thing the "export Linked Files" option doesn't take into account is when Links are mirrored in the Host File. But I would state that mirroring a Link is by definition eligible for severe corporal punishment.

As a part of this handout you will find the project shown in the following image: an empty Host File with some links in it.

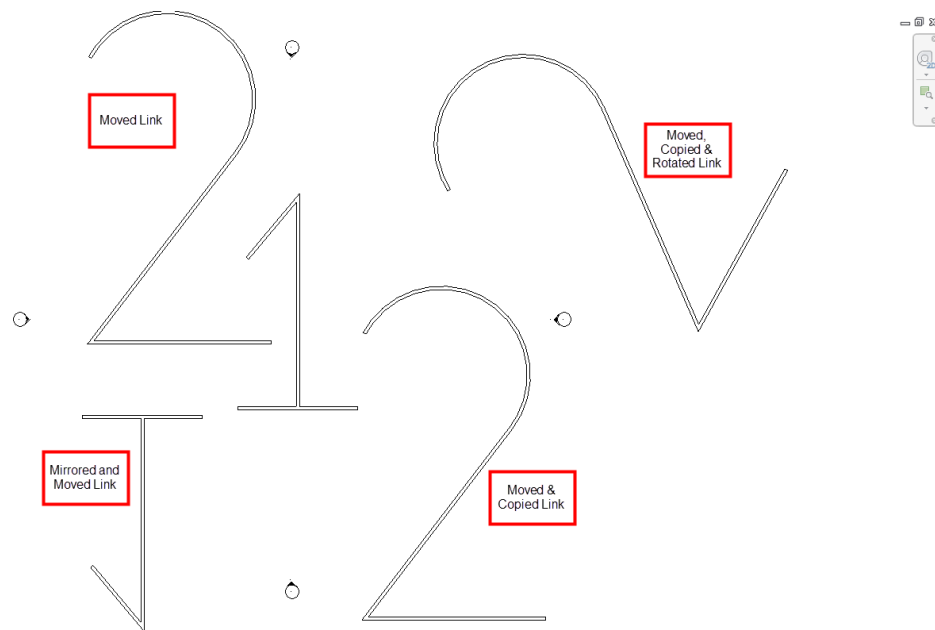


Image 38: Example Links transformed in Host File

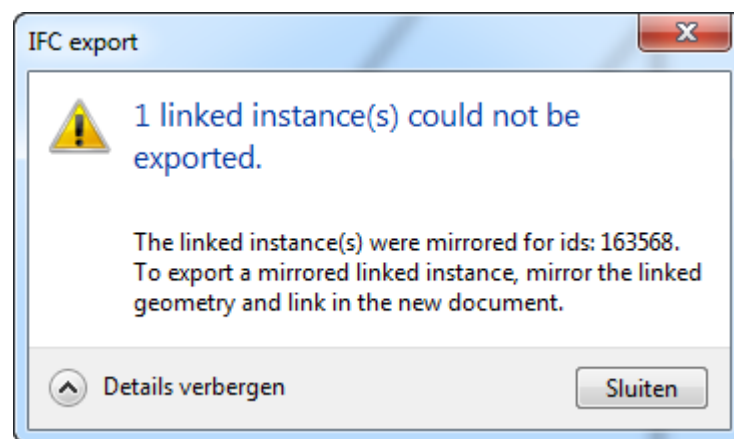


Image 39: Error message when exporting a mirrored Link

The resulting export contains 5 files:

1. The empty Host File (which contained no geometry to begin with)
2. Link 1
3. Link 2, moved
4. Link 2, moved and copied
5. Link 2, moved, copied and rotated.

The only one missing is Link 1, mirrored and moved.

Merging IFCs

These Links are exported and saved as separate files, see below.

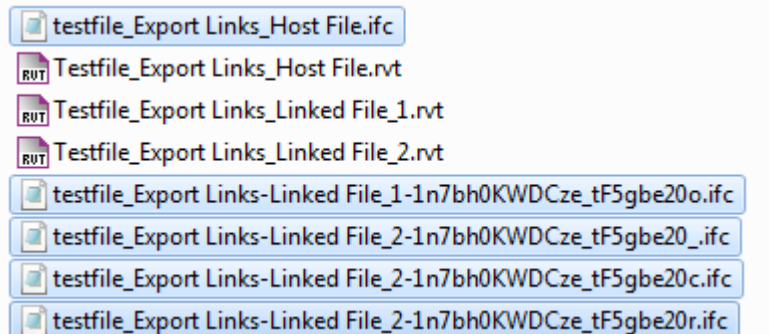


Image 40: Exported Links as separate IFC's

But then what? What if you still need to hand over a single IFC as a deliverable? Fact of the matter is that a lot of software tools can merge IFCs: Navisworks can do it, but so can Solibri, Tekla BIMSight, SimpleBIM and Constructivity. The one problem is: none of these tools can export a single IFC file. They can all merge the IFCs and save them into their native file format. But there is only one tool available that can actually save the merged file into a single IFC.

And the winner is:

TNO BIMServer

This Open Source platform is designed to process IFC files that are uploaded. It is not meant to act as a viewer (although it can), or even have a user interface (which is also an option). It is meant to automate tasks that need to be performed every time an IFC is uploaded. These tasks can vary from something very simple like merging a set of IFCs, to very complex workflows that are automatically run every time an IFC is uploaded.

Even though it's hard to believe, this is the only tool that can successfully import, merge and re-export IFCs.

Because it's an Open Source project (and this seems to be a rule of thumb here) getting BIMServer up and running isn't all that easy. So here's a quick step-by-step on how to do this.

1. Download Java 64 bit from this link: <http://goo.gl/tCRtQP>

Even though you have a 64bit computer, your browser probably is just 32bit (yes, that includes your specific browser, whatever the flavor is). BIMServer won't work on 32 bit Java. And since Java uses the browser to determine which version (32 or 64 bit) needs to be installed, you probably won't be running 64 bit Java. So stop being a wise-ass and install the damn software.

2. Download the BIMServer package from this link: <http://goo.gl/0bAHxt>
I usually pick the second latest version.

Like I said, it's Open Source. Latest versions tend to be buggy. In this case, take version 1.3.3-FINAL-2014-10-12.jar.

Yes, it's supposed to be a jar-file. Heaven knows why. Probably because that too is an Open Source format. Download it and put it where you want to run BIMServer. Keep this in mind:

BIM Server doesn't need installing. It just unpacks and you're good to go. But it saves all its data to this location and there is no way to tell it to go elsewhere. So you have to have write permission and enough space.

Now double click. It should start running and pop up a progress screen. Copy the settings from Image 41 and hit "Start".

When running this for the first time it could take a while to extract and build the server environment. Don't be alarmed if it freezes up for a minute or two. After a while it should give you the resulting screen as shown in Image 42. And, when you look in your Explorer window it should have added some files and folders too, as shown in Image 43. Once this is shown, hit "Stop" to exit and close the screen.

Congratulations, you are now the proud owner of an Open Source Open BIM Model Server.

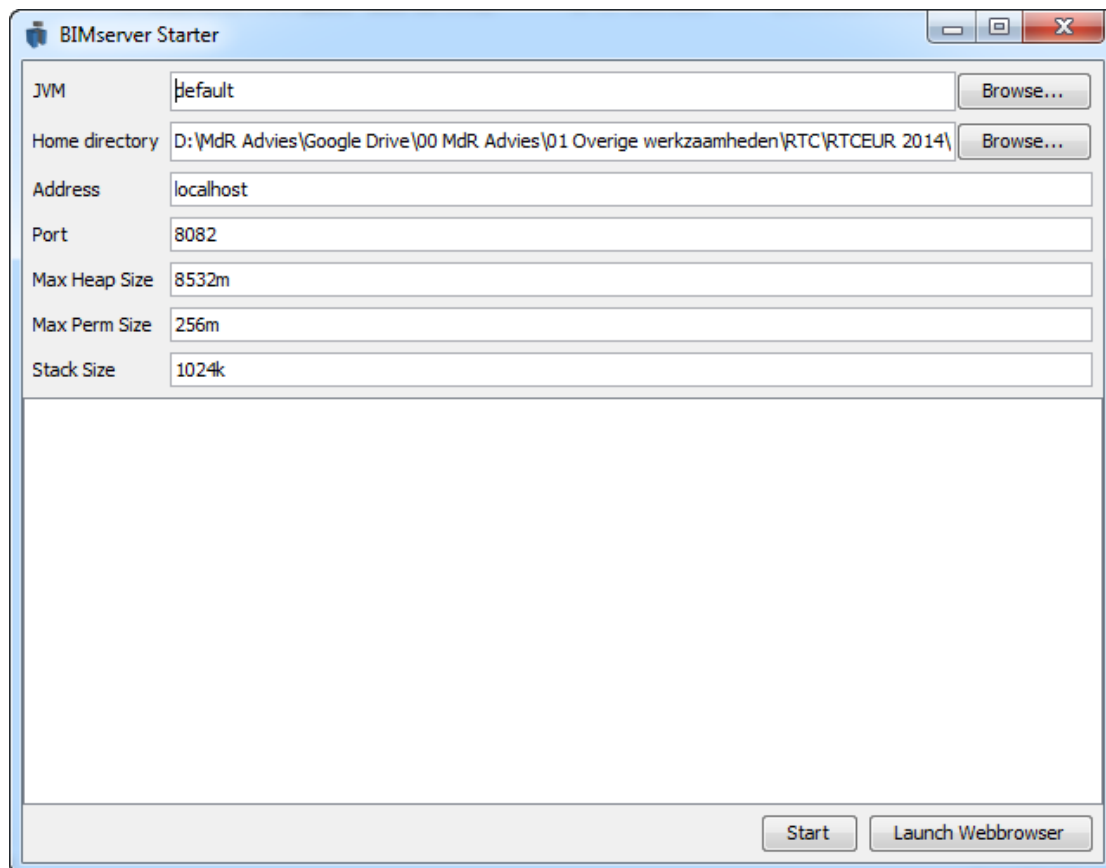
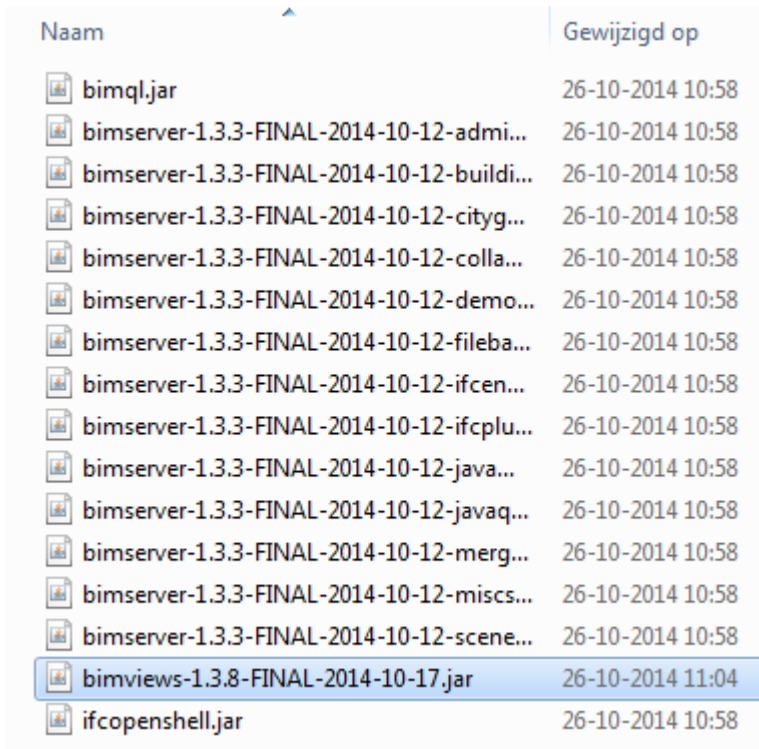


Image 41: BIMserver startup screen



You would think that does the trick, but like I said: BIMServer is not meant to have a User Interface. Plus, it's Open Source and therefore

not very user friendly. What you need to have is a plugin that gives you the desired UI, and this is what the Bimvie.ws plugin does. Download it, and put it in the "Plugins" subfolder of your BIMServer folder as shown in the image.



Naam	Gewijzigd op
bimql.jar	26-10-2014 10:58
bimserver-1.3.3-FINAL-2014-10-12-admi...	26-10-2014 10:58
bimserver-1.3.3-FINAL-2014-10-12-buildi...	26-10-2014 10:58
bimserver-1.3.3-FINAL-2014-10-12-cityg...	26-10-2014 10:58
bimserver-1.3.3-FINAL-2014-10-12-colla...	26-10-2014 10:58
bimserver-1.3.3-FINAL-2014-10-12-demo...	26-10-2014 10:58
bimserver-1.3.3-FINAL-2014-10-12-fileba...	26-10-2014 10:58
bimserver-1.3.3-FINAL-2014-10-12-ifcen...	26-10-2014 10:58
bimserver-1.3.3-FINAL-2014-10-12-ifcplu...	26-10-2014 10:58
bimserver-1.3.3-FINAL-2014-10-12-java...	26-10-2014 10:58
bimserver-1.3.3-FINAL-2014-10-12-javaq...	26-10-2014 10:58
bimserver-1.3.3-FINAL-2014-10-12-merg...	26-10-2014 10:58
bimserver-1.3.3-FINAL-2014-10-12-miscs...	26-10-2014 10:58
bimserver-1.3.3-FINAL-2014-10-12-scene...	26-10-2014 10:58
bimviews-1.3.8-FINAL-2014-10-17.jar	26-10-2014 11:04
ifcopenshell.jar	26-10-2014 10:58

Image 44: BIMviews plugin added to the "Plugins" folder

4. Rerun the Bimserver by double clicking the Bimserver icon in the Explorer screen. It should be powering up significantly faster than the first time. Once it's done starting the server, hit the "Launch in browser" button. This will take you to a screen as shown in the image below.



Setup BiMserver

This BiMserver has not yet been setup. You can setup this BiMserver right now.

Site address

Administrator

Administrator name

Administrator username

Administrator password

[Setup e-mail](#)

Image 45: Setup screen for BiMserver

5. You will need to type in some administrative stuff here. First screen you need to set up a server address. If you're just using it locally to merge IFC's, use the setting as applied in Image 45. Simply add an administrative username (has to be an email address and password). You could also setup an email server, but that's not really necessary. These specific settings only need to be added if you plan on automating stuff and letting Bimserver send out emails when certain (scheduled) tasks are completed. If you have no intention of doing so: don't even worry about clicking the link "Setup e-mail".

And that's it. Once you hit setup, the browser screen will take you to a web-based viewer mode where you can work your magic.

For me, this was the first time I actually configured a server myself. And I had to find out about the Java thing and Bimvie plugin all by myself. Getting it to work was both awesome and frustrating. And has prevented me from starting to dabble with Bimserver. I still use it for one thing only: merging IFCs into a single file.