SEVENTH FRAMEWORK PROGRAMME OF THE EUROPEAN COMMUNITY
(EC GRANT AGREEMENT N° 26088)

ICT PLATFORM FOR HOLISTIC ENERGY EFFICIENCY SIMULATION
AND LIFECYCLE MANAGEMENT OF PUBLIC USE FACILITIES

HESMOS

Virtual Energy Lab

Deliverable D8.2:
Integrated Interoperability Methods

Responsible Authors:

Tuomas Laine, Ken Baumgärtel

Co-Authors:

Peter Katranuschkov, Eino Kukkonen, Francisco Forns-Samso, Raimund Zellner, Timotej Paulik

Due date: 31.08.2012
Issue date: 31.08.2012

Nature: Prototype

Start date of project: 01.09.2010 **Duration: 36 months**

Organisation name of lead contractor for this deliverable:

Granlund OY (OG)

History:

Version	Description	Lead Author	Date
0.1	Initiation & principal structure	T. Laine (OG)	01.06.2012
0.2	Introduction and draft content	K. Baumgärtel (TUD-CIB)	12.06.2012
0.3	IAS Specification	K. Baumgärtel (TUD-CIB)	15.06.2012
0.4	Updated section 1.2	K. Baumgärtel (TUD-CIB)	25.06.2012
0.5	Section 1.2.5 added	T. Laine (OG)	13.08.2012
0.6	Final Revision	K. Baumgärtel (TUD-CIB)	22.08.2012
1.0	**Checked and approved final version**	TUD-CIB	31.08.2012

Citation

Laine T., Baumgärtel K., Katranuschkov, P., Kukkonen E., Forns-Samso F., Zellner R., Paulik T. (2012): *HESMOS Deliverable 8.2: Integrated Interoperability Methods*, © HESMOS Consortium, Brussels.

Acknowledgements

The work presented in this document has been conducted in the context of the seventh framework programme of the European community project HESMOS (n° 26088). HESMOS is a 36-month project that started in September 2010 and is funded by the European Commission as well as by the industrial partners. Their support is gratefully appreciated.
The partners in the project are Technische Universität Dresden (Germany), NEMETSCHEK Slovensko, S.R.O. (Slovakia), Granlund OY (Finland), Royal BAM Group NV (The Netherlands), Obermeyer Planen + Beraten (Germany) and AEC3 LTD. This report was created through the joint effort of the above organizations.

Project of SEVENTH FRAMEWORK PROGRAMME OF THE EUROPEAN COMMUNITY		
	Dissemination Level	
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

TABLE OF CONTENTS:

Executive Summary

The objectives of WP8 are the provision of the central product of HESMOS, which is the Integrated Virtual Energy Laboratory (IVEL) organised as a web-based platform according to the SOA approach. A main task is the integration of the system components developed in the WPs 3-7 so that they shall appear to the end user as one homogenous simulator serving different end user groups with different attitudes, viewpoints and abilities, and to prepare the HESMOS product and component services and tools for exploitation. This includes the development of integrative interoperability methods and configuring them to the particular user needs, development of process scenarios based on the identified use case scenarios from WP1, the orchestration of the implemented services and tools, including integration of all required support/middleware components providing common access to all services via appropriate semantically grounded access rights coordination of integration related activities in WPs 3-7 and the management of all related exploitation and IPR issues.

Deliverable D 8.2 comprises a software prototype and this supporting report, thereby fulfilling the second task of the overall work in WP8, namely:

- T8.2 Integrated interoperability methods

The deliverable report is structured into 5 sections:

In the **first section**, we give a short overview of this deliverable.

In the **second section**, we present the methods of the Intelligent Access Services (IAS) of the IVEL Core. These methods are structured in general methods like registration or login of a user, model specific methods which are mainly based on the IFC model, simulation methods for starting the appropriate energy solvers, visualization methods for the nD navigator and further service methods especially for monitoring the real pilot building.

The **third section** shows the two most important workflows described via UML sequence diagrams.

The **fourth section** provides an overview on the underlying link model which combines information from several models and will be stored on the server so that the user can load it or modify it later on.

The **last section** presents a conclusion enumerating the important results achieved so far.

Three partners were involved in the RTD work and each partner has contributed from their expert viewpoint as follows:

OG:	Overall concept and configuration of the Facility Management tools;
TUD:	Overall concept, IVEL Core configuration and test-bed;
NEM:	Configuration of the IVEL Connector and the nD Navigator.

1 Introduction

This Deliverable Report presents the developed web interface of the HESMOS Integrated Virtual Energy Laboratory supporting the interoperable use of all platform services and tools, the so called **Intelligent Access Services (IAS)**. Its purpose is:

(1) to document how the web service can be accessed, which methods are developed and how they can be used. This provides a starting point for the integration of all services and tools,

(2) to facilitate understanding the technological concepts of the IAS to a broader circle of developers that may wish to adapt the virtual laboratory for other purposes and tasks.

The Intelligent Access Services (IAS) (highlighted in Figure 1) is a service layer in the IVEL Core and creates the connection to the energy computing and monitoring web services to provide the simulated and measured data to other web applications and web services. Furthermore, it combines the workflows of starting simulations, managing simulations (start/cancel and list of running simulations), managing general information (e.g. user management), and managing the required measurement data. While it provides the management functionalities it also is responsible for storing requested data and saving the runtime changes of input parameters. Therefore the workflows are based on a link model where relations to input parameter and results are defined and can be queried easily.

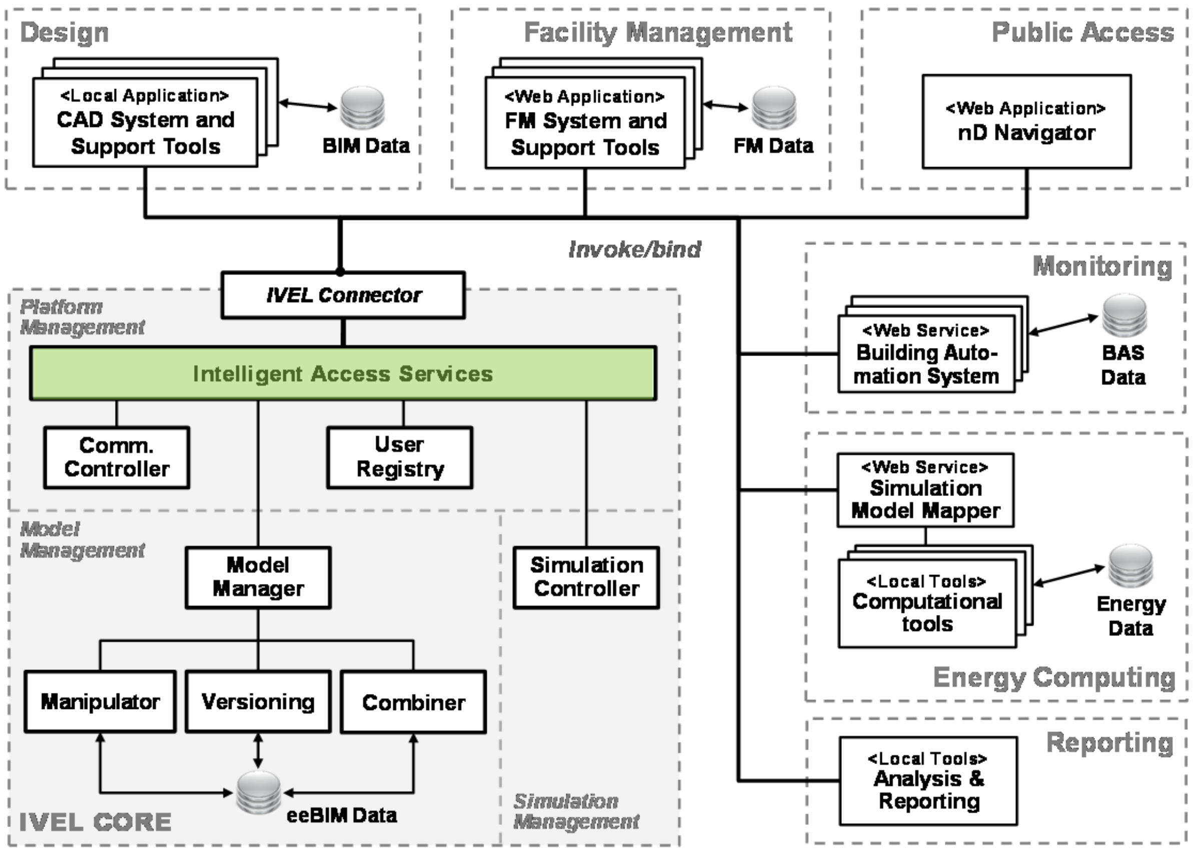


Figure 1: The HESMOS architecture highlighting the focused IAS component

The IAS analyses user requests from the web applications which are connected to it, e.g. the nD navigator. It is possible that a user can compare real measured data with a simulation of one or more locations in a specific time period. When such a user request is given the IAS creates multiple more detailed requests and sends them to the appropriate web services like the energy computing service and the BAS web service. Hence, the IAS call multiple methods until the user request is fulfilled. The analysis of the user requests involves the distinction of the calling applications. In the case of the nD navigator a file (e.g. JSON) for visualizing the results is requested. In the case of facility management it needs a more structured file (e.g. XML). This post-processing is one of the most important services in the IAS.

The following sections describe the multiple methods which are provided by the IAS and can be accessed through web interfaces. While various general methods are important for assigning service calls and guaranteeing a multi-user environment, the model, post-processing and visualization related methods are for the applications which are providing graphical user interfaces and need to call the IVEL Core for working on the energy-enhanced building information model (eeBIM).

This document is not describing the services inside the IVEL Core and also not the other modules around it. It defines the interface methods comprising the IAS. Furthermore it shows how possible workflows will be covered and stored in the eeBIM.

2 IAS Specification

The IAS is a web service which uses REST as architectural style based on web-standards and the HTTP protocol. REST was first described by (Fielding, 2000). In a REST based architecture everything is a resource. A resource is accessed via a common interface based on the HTTP standard methods. In a REST architecture you typically have a REST server which provides access to the resources and a REST client which accesses and modify the REST resources. Every resource should support the HTTP common operations. Resources are identified by global IDs (which are typically URIs). REST allows that resources have different representations, e.g. text, XML, JSON etc. The rest client can ask for specific representation via the HTTP protocol.

In the following we are using a basic URL for the web service and a relative URL for each method. The current (can be changed in future) base URL is:

http://bci53.cib.bau.tu-dresden.de:8080/ivel/ias

A relative URL can be:

/register

The full URL is then:

http://bci53.cib.bau.tu-dresden.de:8080/ivel/ias/register

The WADL of this REST service can be found at:

http://bci53.cib.bau.tu-dresden.de:8080/ivel/ias/application.wadl

The specification for each method is structured into six categories: (1) the method in UML, (2) the relative URL which will be appended to the base URL, (3) the HTTP request which can be for example GET, POST, DELETE or PUT, (4) the input which can be simple text or complex data represented in JSON or XML, (5) the output and the content type of it in simple text or complex data and (6) an example of using the method.

2.1 General Methods

The IVEL Core provides general methods for uploading and downloading files or registration and authentication of users. These methods are guaranteeing the work of the IVEL Core in a multi-user environment. Generally, they are the starting point for working with the other methods (model, simulation, visualization etc.).

2.1.1 User registration

The IVEL Core provides the registration of users which will be authorized to use this web service and all methods that will be described in the following. While much information about a user can be provided, we focused on the essential username and a password.

Method Name	*registerUser (username : string, password : string) : string*
Relative URL	*/register*
HTTP request	POST
Input	application/x-www-form-urlencoded, HTML form tag with the input tag elements namely ‘username’ and ‘password’, like in the example
Output/Content-Type	text/plain, HTTP Status like OK ‘200’ or Bad Request ‘400’ with a message

Example

Following in HTML:

```
<form action="http://bci53.cib.bau.tu-
dresden.de:8080/ivel/ias/register" method="post">
 <p>
  User name : <input type="text" name="username" />
 </p>
 <p>
  Password : <input type="password" name="password" />
 </p>
 <input type="submit" value="Add User" />
</form>
```

Returns ‘200’

2.1.2 Start session

To use all methods except for the *register* method above a user must be logged in. The IVEL Core makes a validation of the given username and password and after a successful check it returns a temporary session identifier. This session identifier has to be included in further method calls to assign HTTP requests to logged in users.

Method Name	*startSession (username : string, password : string) : string*
Relative URL	*/startsession*
HTTP request	POST
Input	application/x-www-form-urlencoded, HTML form tag with the input tag elements namely ‘username’ and ‘password’, like in the example

Output/Content-Type	text/plain, a newly created session identifier ('sessionId') which can be used to access other methods

Example Following in HTML:

```
<form action="http://bci53.cib.bau.tu-
dresden.de:8080/ivel/ias/startsession" method="post">
 <p>
  User name : <input type="text" name="username" />
 </p>
 <p>
  Password : <input type="password" name="password" />
 </p>
 <input type="submit" value="Login" />
</form>
```

Returns e.g. the session identifier 'i43cG48jhDSio23'

2.1.3 End session

This method is called at the end, when the user concludes his work and logs out of the IVEL system. The session will be closed afterwards.

Method Name	*endSession (sessionId : string) : string*
Relative URL	*/endsession/sessionId/{sessionId}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string
Output/Content-Type	text/plain, HTTP Status like OK '200' or Bad Request '400' with a message

Example Following as URL:

```
~/endsession/sessionId/i43cG48jhDSio23
```

Returns '200' (the session is ended in the background)

2.1.4 Start transaction

To use methods that affect the eeBIM process like accessing IFC files, the IVEL Core needs to know which eeBIM container will be used in the user session. This is important because a user can work on multiple eeBIM in parallel and can start many simulations simultaneously. The information which eeBIM will be used is declared via the identifier 'eeBimId'.

Method Name	*startTransaction (sessionId : string, eeBimId : string) : string*
Relative URL	*/starttransaction/sessionId/{sessionId}/eeBimId/{eeBimId}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string, an eeBIM identifier as a xs:string
Output/Content-Type	text/plain, HTTP Status like OK '200' or Bad Request '400' with a message
Example	Following as URL: `~/starttransaction/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423hj` Returns '200' (the eeBimId is passed back in the background)

2.1.5 End transaction

The IVEL Core provides a method for ending a transaction with regard to the active eeBIM process and saving the current eeBIM state.

Method Name	*endTransaction (sessionId : string, eeBimId : string) : string*
Relative URL	*/endtransaction/sessionId/{sessionId}/eeBimId/{eeBimId}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string, an eeBIM identifier as a xs:string
Output/Content-Type	text/plain, HTTP Status like OK '200' or Bad Request '400' with a message
Example	Following as URL: `~/endtransaction/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423hj` Returns '200' (the transaction is ended in the background and the eeBIM status is saved on the IVEL)

2.1.6 Upload file

It is important to upload files to the eeBIM container where it can be inter-linked. It is important to send the eeBIM identifier to assign the file correctly.

Method Name	*uploadFile (sessionId : string, eeBimId : string, uploadFile : object) : string*
Relative URL	*/uploadFile*
HTTP request	POST
Input	multipart/form-data, a session identifier as a xs:string, a eeBim identifier as a xs:string and an uploadFile input type
Output/Content-Type	text/plain, a file identifier for the file or Bad Request '400'

Example Following in HTML:

```
<form action="http://bci53.cib.bau.tu-
dresden.de:8080/ivel/ias/uploadFile" method="post"
enctype="multipart/form-data">
  <input type="hidden" name="sessionId"
value="i43cG48jhDSio23"/>
  <input type="hidden" name="eeBimId" value="jhkjh423hj"/>
 <p>
  File : <input type="file" name="uploadFile" />
 </p>
 <input type="submit" value="Upload it" />
</form>
```

Returns 'file43jhkjhk43' as file identifier

2.1.7 Download file

An application or user can download a file from the eeBIM by a given file identifier.

Method Name	*downloadFile (sessionId : string, eeBimId : string, fileId : string) : string*
Relative URL	*/downloadFile/sessionId/{sessionId}/eeBimId/{eeBimId}/fileId/{fileId}*
HTTP request	GET
Input	a session identifier as a xs:string, a eeBIM identifier as xs:string and a file identifier as xs:string
Output/Content-Type	application/x-zip-compressed, the requested file in the ZIP format

Example	Following as URL:

```
~/downloadFile/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423hj
/fileId/jghjgh4234
```

Returns

‘http://bci53.cib.bau.tu-dresden.de:8080/ivel/ias/Building.zip’

2.2 Model Methods

Model methods are methods providing access to model specific information. They are used to filter or manipulate models like BIM/IFC (buildingSMART International Modeling Support Group, 2009) or the simulation model inside the eeBIM.

2.2.1 Get all building global identifiers

With this method a list of all ‘GlobalIds’ of the *IfcBuilding* entities in an IFC file can be retrieved.

Method Name	*getBuildings (sessionId : string, eeBimId : string) : list*
Relative URL	*/buildings/sessionId/{sessionId}/eeBimId/{eeBimId}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string, an eeBIM identifier as a xs:string
Output/Content-Type	application/xml with xs:string, see example
Example	Following as URL:

```
~/buildings/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423hj
```

Returns:

```
<list>
 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">3Xk4XFer12gg5SHM6E_kg6</item>

 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">2ABQr4K398zQ55DYGCMacf</item>
</list>
```

2.2.2 Get all space names of a building

This is a method for retrieving a list of all names (IFC attribute 'name') of all *IfcSpace* entities in an IFC file.

Method Name	*getSpaceNames (sessionId : string, eeBimId : string, buildingGuid : string) : list*
Relative URL	*/spacenames/sessionId/{sessionId}/eeBimId/{eeBimId}/guid/{buildingGuid}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string, an eeBIM identifier as a xs:string and a building GlobalId as xs:string
Output/Content-Type	application/xml with xs:string, see example
Example	Following as URL:

```
~/spacenames/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423hj/gu
id/3Xk4XFer12gg5SHM6E_kg6
```

Returns:

```
<list>
 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">A1.2.16</item>

 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">A1.3.03</item>
</list>
```

2.2.3 Get all space long names of a building

This method returns a list of all long names (IFC attribute 'LongName') of the *IfcSpace* entities in an IFC file.

Method Name	*getSpaceLongNames (sessionId : string, eeBimId : string, buildingGuid : string) : list*
Relative URL	*/spacelongnames/sessionId/{sessionId}/eeBimId/{eeBimId}/guid/{buildingGuid}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string, an eeBIM identifier as a xs:string and a building GlobalId as xs:string

Output/Content-Type	application/xml with xs:string, see example
Example	Following as URL:

```
~/spacelongnames/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423h
j/guid/3Xk4XFer12gg5SHM6E_kg6
```

Returns:

```
<list>
 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">Kitchen</item>

 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">Office</item>
</list>
```

2.2.4 Get all space global identifiers of a building

This method is for retrieving a list of all 'GlobalIds' of the *IfcSpace* entities in a specific building given by the *IfcBuilding* 'GlobalId' attribute.

Method Name	*getSpaceGuids (sessionId : string, eeBimId : string, buildingGuid : string) : list*
Relative URL	*/spaceguids/sessionId/{sessionId}/eeBimId/{eeBimId}/guid/{buildingGuid}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string, an eeBIM identifier as a xs:string and a building GlobalId as xs:string
Output/Content-Type	application/xml with xs:string, see example
Example	Following as URL:

```
~/spaceguids/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423hj/gu
id/3Xk4XFer12gg5SHM6E_kg6
```

Returns:

```
<list>
 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">1JBup6r7vF8uSdicok6YjV</item>

 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">0rAUSUgi5DNAqHRoXmGhUi</item>
</list>
```

2.2.5 Get materials of a building element

This is a method for retrieving a list of all materials of the *IfcBuildingElement* entities in an IFC file related with *IfcRelAssociatesMaterial*. It returns an ordered list with material names as specified in *IfcMaterialLayerSet*.

Method Name	*getMaterialsOfBuildingElement (sessionId : string, eeBimId : string, buildingElementGuid : string) : list*
Relative URL	*/matofbe/sessionId/{sessionId}/eeBimId/{eeBimId}/guid/{buildingElementGuid}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string, an eeBIM identifier as a xs:string and a building element GlobalId as xs:string
Output/Content-Type	application/xml with xs:string, see example

Example

Following as URL:

```
~/matofbe/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423hj/guid/
1hKtzFYOfDYuCIwFOYDZEP
```

Returns:

```
<list>
 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">Concrete</item>

 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">Air gap</item>

 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">Mineral Wool</item>

 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">Concrete</item>
</list>
```

2.2.6 Get all materials of a building

This is a method for retrieving a list of all materials inside a building. It returns a map with entries distinguished by the 'GlobalId' attribute of the *IfcBuildingElement* as keys, and an ordered list with material names like specified in *IfcMaterialLayerSet*[1] as values.

[1] http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ifcmaterialresource/lexical/ifcmateriallayerset.htm

Method Name	*getMaterialsOfBuilding (sessionId : string, eeBimId : string, buildingGuid : string) : map*
Relative URL	*/matofb/sessionId/{sessionId}/eeBimId/{eeBimId}/guid/{buildingGuid}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string, an eeBIM identifier as a xs:string and a building GlobalId as xs:string
Output/Content-Type	application/xml with xs:string, see example

Example

Following as URL:

```
~/matofb/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423hj/guid/3
Xk4XFer12gg5SHM6E_kg6
```

Returns:

```
<mapdata>
 <map>
  <entry>
   <key xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">0rAUSUgi5DNAqHRoXmGhUi</key>
   <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">Concrete</value>
   <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">Air gap</value>
   <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">Mineral Wool</value>
   <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">Concrete</value>
  </entry>
  <entry>
   <key xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">0nCK85XYPDV8k30cAEDNLp</key>
   <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">Concrete</value>
   <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">Wood</value>
  </entry>
 </map>
</mapdata>
```

2.2.7 Get all outer elements of a building

This method analyses the position of building elements and returns a list with 'GlobalIds' of the building elements which are forming the envelope of the requested building.

Method Name	*getOuterElements (sessionId : string, eeBimId : string, buildingGuid : string, direction : string) : list*
Relative URL	*/outerelements/sessionId/{sessionId}/eeBimId/{eeBimId}/guid/{buildingGuid}/direction/{direction}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string, an eeBIM identifier as a xs:string, a building GlobalId as xs:string and a direction value as xs:int with the following possible values: 0: all building elements with north direction 2: all building elements with east direction 4: all building elements with south direction 6: all building elements with west direction
Output/Content-Type	application/xml with xs:string, see example

Example

Following as URL:

```
~/outerelements/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423hj
/guid/3Xk4XFer12gg5SHM6E_kg6/direction/0
```

Returns:

```
<list>
 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">0nCK85XYPDV8k30cAEDNLp</item>
 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">0YUJCCPpLCO9x_7wXoisKx</item>
<item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">0VDmNjTZz58QOJC4XrO$mL</item>
</list>
```

2.2.8 Get all outer spaces of a building

This method analyses the position of building spaces and returns a list with 'GlobalIds' of the spaces which have related façade elements of the requested building.

Method Name	*getOuterSpaces (sessionId : string, eeBimId : string, buildingGuid : string, direction : string) : list*

Relative URL	*/outerspaces/sessionId/{sessionId}/eeBimId/{eeBimId}/guid/{buildingGuid}/direction/{direction}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string, an eeBIM identifier as a xs:string, a building GlobalId as xs:string and a direction value as xs:int with following possible values: 0: all building elements with north direction 1: all building elements with east direction 2: all building elements with south direction 3: all building elements with west direction
Output/Content-Type	application/xml with xs:string, see example

Example

Following as URL:

```
~/outerspaces/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423hj/g
uid/3Xk4XFer12gg5SHM6E_kg6/direction/0
```

Returns:

```
<list>
 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">0rAUSUgi5DNAqHRoXmGhUi</item>
 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">1JBup6r7vF8uSdicok6YjV</item>
</list>
```

2.2.9 Get all building elements of a specific type

This method returns all building element identifiers (IFC GlobalId) of a requested type.

Method Name	*getElementsForType (sessionId : string, eeBimId : string, buildingGuid : string, type : string) : list*
Relative URL	*/elementsfortype/sessionId/{sessionId}/eeBimId/{eeBimId}/guid/{buildingGuid}/type/{type}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string, an eeBIM identifier as a xs:string, a building GlobalId as xs:string and a type value as xs:string.
Output/Content-Type	application/xml with xs:string, see example

Example	Following as URL: `~/elementsfortype/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423hj/guid/3Xk4XFer12gg5SHM6E_kg6/type/wall`

Returns:

```
<list>
 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">0rAUSUgi5XCOsHRoXmGhPa</item>
 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">1JBup6r7vF8uSdicok6YqA</item>
</list>
```

2.2.10 Get the type of a building element

This method returns the type of the requested building element.

Method Name	*getTypeForElement (sessionId : string, eeBimId : string, buildingElementGuid : string) : list*
Relative URL	*/typeforelement/sessionId/{sessionId}/eeBimId/{eeBimId}/guid/{buildingElementGuid}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string, an eeBIM identifier as a xs:string, a building element GlobalId as xs:string
Output/Content-Type	text/plain
Example	Following as URL: `~/typeforelement/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423hj/guid/3Xk4XFer12gg5SHM6E_kg6`

Returns:

The type ‘facadewall’.

2.3 Simulation Methods

Simulation methods are providing managing capabilities for starting or listing current running simulations and assigning simulation parameters.

2.3.1 Assign climate data

In the eeBIM generation process the building model has to be linked with climate data for the location of the building. To make an explicit assignment this method can be called. Via the ‘RefLatitude’ attribute and ‘RefLongitude’ attribute of *IfcSite* in the BIM model it will search the corresponding climate data for the region.

Method Name	*assignClimateData (sessionId : string, eeBimId : string, climateId : string) : string*
Relative URL	*/assignclimate*
HTTP request	POST
Input	Multipart/form-data, a session identifier as a xs:string, an eeBim identifier as xs:string and a climateId as xs:string
Output/Content-Type	text/plain, HTTP Status like OK ‘200’ or Bad Request ‘400’ with a message
Example	Following as URL: `~/assignclimate/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423hj/climateId/5` Returns ‘200’ (assigns/links the climate data reference to the BIM in the background)

2.3.2 Assign materials to a building element

In the eeBIM generation process assignment of materials to building elements has to be done. To make an explicit assignment of materials this method can be called.

Method Name	*assignMaterials (sessionId : string, eeBimId : string, buildingGuid : string, buildingElementGuid : string, materialId : string, order : int) : string*
Relative URL	*/assignmaterials*
HTTP request	POST
Input	Multipart/form-data, a session identifier as a xs:string, an eeBIM identifier as a xs:string, a building GlobalId as xs:string, a building element GlobalId as xs:string, a material identifier as xs:string and the order of the material layer as xs:int

Output/Content-Type	text/plain, HTTP Status like OK '200' or Bad Request '400' with a message
Example	Following as URL: `~/assignmaterials` `With form parameters sessionId = i43cG48jhDSio23, eeBimId = jhkjh423hj` Returns '200' (assigns/links the material data reference (files or database) to the BIM in the background)

2.3.3 Convert space boundaries

Before starting a simulation it is necessary that second level space boundaries are provided in the IFC file. This method should be called before starting a simulation to guarantee this model quality. It will generate a new IFC file from the original file including none or only first level space boundaries with this additional information, and link it inside the eeBIM. The method will do nothing if the file already has this model quality.

Method Name	*convertSpaceBoundaries (sessionId : string, eeBimId : string) : string*
Relative URL	*/convertsb/sessionId/{sessionId}/eeBimId/{eeBimId}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string and an eeBim identifier as xs:string
Output/Content-Type	text/plain, HTTP Status like OK '200' or Bad Request '400' with a message
Example	Following as URL: `~/convertsb/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423hj` Returns '200' (space boundaris are generated and a new BIM/IFC version is respect-tively created in the background)

2.3.4 Start energy simulation

This method will start the energy simulation. It needs a valid session and an eeBim identifier and a list with GlobalIds of the building, rooms, storeys or building elements which shall be simulated.

Method Name	*startEnergySimulation (sessionId : string, eeBimId : string, locationIds : list) : string*

Relative URL *\/energysim*

HTTP request POST

Input Multipart/form-data, a session identifier as a xs:string, an eeBim identifier as xs:string and a list with GlobalId of *IfcBuildingElements* (some building elements), *IfcSpace* (some rooms), *IfcBuildingStorey* (a storey) or *IfcBuilding* (whole building) as xs:string

Output/Content-Type text/plain, a newly created simulation identifier

Example Following as URL:

```
~/energysim
with parameters: sessionId = i43cG48jhDSio23, eeBimId = jhkjh423hj
```

and the list (locationIds) of GlobalIds:

```
<list>
 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">0nCK85XYPDV8k30cAEDNLp</item>
 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">0YUJCCPpLCO9x_7wXoisKx</item>
 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">0VDmNjTZz58QOJC4XrO$mL</item>
</list>
```

Returns simulation identifier ‘sim1jhk453’.

2.3.5 Cancel energy simulation

This method will cancel a running energy simulation.

Method Name *cancelEnergySimulation (sessionId : string, simId : string) : string*

Relative URL */cancelsim/sessionId/{sessionId}/simId/{simId}*

HTTP request GET

Input text/plain, a session identifier as a xs:string and an simulation identifier as xs:string

Output/Content-Type text/plain, HTTP Status like OK ‘200’ or Bad Request ‘400’ with a message

Example Following as URL:

```
~/cancelsim/sessionId/i43cG48jhDSio23/simId/sim1jhk453
```

Returns ‘200’ (the simulation run is stopped in the background)

2.3.6 List simulations for eeBIM

This method will return a list of simulation identifiers for a specific eeBIM.

Method Name	*listSimulationsForEeBim (sessionId : string, eeBimId : string) : list*
Relative URL	*/listsimeebim/sessionId/{sessionId}/eeBimId/{eeBimId}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string and an eeBIM identifier as xs:string
Output/Content-Type	text/plain, list with simulation identifiers as xs:string
Example	Following as URL: `~/listsimeebim/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423hj`

Returns e.g. the following list:

```
<list>
 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">sim1jhk453</item>

 <item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">sim2hk3$mL</item>
</list>
```

2.3.7 List all simulations

This method will return a list of all simulation identifiers separated by their related eeBIM.

Method Name	*listAllSimulations (sessionId : string) : map*
Relative URL	*/listsim/sessionId/{sessionId}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string
Output/Content-Type	text/plain, map with eeBIM identifiers as xs:string as keys and list with simulation identifiers as xs:string
Example	Following as URL: `~/listsim/sessionId/i43cG48jhDSio23`

Returns e.g. the following:

```
<mapdata>
 <map>
  <entry>
   <key xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">jhkjh423hj</key>
   <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">sim1jhk453</value>
   <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">sim2jhk3$mL</value>
  </entry>
  <entry>
   <key xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">fg2Ydfewrq</key>
   <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">sim1ffqw!3</value>
  </entry>
 </map>
</mapdata>
```

2.3.8 Download BCF file

This method is provided to download a BCF file[1] generated after the simulation so that the architect knows which elements should be changed in the building model in order to provide the respective energy performance related result.

Method Name	*downloadBCMFile (sessionId : string, eeBimId : string) : string*
Relative URL	*/downloadbcm/sessionId/{sessionId}/eeBimId/{eeBimId}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string and an eeBim identifier as xs:string
Output/Content-Type	application/x-zip-compressed, a ZIP file with the BCM file
Example	Following as URL: `~/downloadbcm/sessionId/i43cG48jhDSio23/eeBimId/jhkjh423hj` Returns: A file: 'bcmFile.zip'

[1] A file in the BIM Collaboration Format (BCF) as specified by BuildingSMART.

2.3.9 Get the status of a running energy simulation

After starting a simulation it is necessary to check if the simulation is completed before asking for results. This method checks and returns that.

Method Name	*getEnergySimulationStatus (sessionId : string, simId : string) : string*
Relative URL	*/simstatus/sessionId/{sessionId}/simId/{simId}*
HTTP request	GET
Input	text/plain, a session identifier as a xs:string and an simulation identifier as xs:string
Output/Content-Type	text/plain, simulation status “Completed”, “Running” or “N/A”.
Example	Following as URL: `~/simstatus/sessionId/i43cG48jhDSio23/simId/sim1jhk453` Returns ‘Running’

2.4 BAS Support Methods

These IAS methods are supported through the developed new building automation system (BAS) web service. The IVEL Core provides the connection to it and prepares the calls for requesting measurement data. The complete list of monitoring methods which are used in the IVEL Core is provided in (Ploennigs et al. 2012).

2.4.1 Get Devices for Room

This method provides the device identifiers in a room.

Method Name	*getDeviceIds (sessionId : string, eeBimId : string, buildingName : string, roomNames : List) : map*
Relative URL	*/devices*
HTTP request	POST
Input	multipart/form-data, application/xml, a session identifier as a xs:string, an eeBim identifier as xs:string, a building name as xs:string and room names of the building as xs:list
Output/Content-Type	application/xml, XML data map with all room names as keys and the device identifiers in a room as values

Example	Following as URL: `~/measurement`
	Requesting the “Room Temperature” of the room “A1.2.16” in the building “A1” on the 1st January 2012 from 2 a.m. to 6 a.m.
	Returns:

```
<mapdata>
 <map>
  <entry>
   <key xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">A.1.2.16</key>
   <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">131_.A1_2_16_MW_RT_TL Archive</value>
   <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">132_.A1_2_16_MW_RT_TL Archive </value>
  </entry>
  <entry>
   <key xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">A.1.1.10</key>
   <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">16_.A1_1_10_MW_RT_TL</value>
  </entry>
 </map>
</mapdata>
```

2.4.2 Get Measurement Data

This method is for retrieving measurement data like “Room Temperature” of a list of devices in a specific building by a given time range.

Method Name	*getMeasurementData (sessionId : string, eeBimId : string, buildingName : string, deviceIds : List, startTime : string, endTime : string, function : string) : string*
Relative URL	*/measurement*
HTTP request	POST
Input	multipart/form-data, application/xml, a session identifier as a xs:string, an eeBim identifier as xs:string, a building name as xs:string, device identifiers of the rooms as xs:list, a start and end time as xs:string according to the ISO8601 and a function (e.g. “Room Humidity”, “Room Temperature”, “CO2 Level”) as xs:string
Output/Content-Type	application/xml, XML data with all measurements in file format (cf. Laine et al. 2012)

Example	Following as URL:
	`~/measurement`
	thereby requesting the “Room Temperature” for the sensor “131_.A1_2_01_MW_RT_TL Archive” in the building “A1” on the 1st January 2012 from 2 a.m. to 6 a.m.
	returns, e.g.:

```
<CONDITIONFILE HostID="localhost" OverwriteSenderHostID="no">
    <LogPoint LogPointID="131_.A1_2_01_MW_RT_TL Archive">
        <LogResult year="2012" month="1" day="1" hour="0" minute="2" second="38" type="TEMPERATURE_C">22,3</LogResult>
        <LogResult year="2012" month="1" day="1" hour="0" minute="17" second="38" type="TEMPERATURE_C">22,3</LogResult>
        <LogResult year="2012" month="1" day="1" hour="0" minute="32" second="38" type="TEMPERATURE_C">22,3</LogResult>
        <LogResult year="2012" month="1" day="1" hour="0" minute="47" second="38" type="TEMPERATURE_C">22,3</LogResult>
        <LogResult year="2012" month="1" day="1" hour="1" minute="2" second="38" type="TEMPERATURE_C">22,3</LogResult>
        <LogResult year="2012" month="1" day="1" hour="1" minute="17" second="38" type="TEMPERATURE_C">22,3</LogResult>
        <LogResult year="2012" month="1" day="1" hour="1" minute="32" second="38" type="TEMPERATURE_C">22,3</LogResult>
        <LogResult year="2012" month="1" day="1" hour="1" minute="47" second="38" type="TEMPERATURE_C">22,3</LogResult>
        <LogResult year="2012" month="1" day="1" hour="2" minute="2" second="38" type="TEMPERATURE_C">22,3</LogResult>
        <LogResult year="2012" month="1" day="1" hour="2" minute="17" second="38" type="TEMPERATURE_C">22,3</LogResult>
        <LogResult year="2012" month="1" day="1" hour="2" minute="32" second="38" type="TEMPERATURE_C">22,3</LogResult>
        <LogResult year="2012" month="1" day="1" hour="2" minute="47" second="38" type="TEMPERATURE_C">22,3</LogResult>
        <LogResult year="2012" month="1" day="1" hour="3" minute="2" second="38" type="TEMPERATURE_C">22,3</LogResult>
        <LogResult year="2012" month="1" day="1" hour="3" minute="17" second="38" type="TEMPERATURE_C">22,3</LogResult>
        <LogResult year="2012" month="1" day="1" hour="3" minute="32" second="38" type="TEMPERATURE_C">22,2</LogResult>
        <LogResult year="2012" month="1" day="1" hour="3" minute="47" second="38" type="TEMPERATURE_C">22,2</LogResult>
        <LogResult year="2012" month="1" day="1" hour="4" minute="2" second="38" type="TEMPERATURE_C">22,2</LogResult>
        <LogResult year="2012" month="1" day="1" hour="4" minute="17" second="38" type="TEMPERATURE_C">22,2</LogResult>
        <LogResult year="2012" month="1" day="1" hour="4" minute="32" second="39" type="TEMPERATURE_C">22,2</LogResult>
        <LogResult year="2012" month="1" day="1" hour="4" minute="47" second="39" type="TEMPERATURE_C">22,2</LogResult>
        <LogResult year="2012" month="1" day="1" hour="5" minute="2" second="39" type="TEMPERATURE_C">22,2</LogResult>
        <LogResult year="2012" month="1" day="1" hour="5" minute="17" second="39" type="TEMPERATURE_C">22,2</LogResult>
        <LogResult year="2012" month="1" day="1" hour="5" minute="32" second="39" type="TEMPERATURE_C">22,2</LogResult>
        <LogResult year="2012" month="1" day="1" hour="5" minute="47" second="39" type="TEMPERATURE_C">22,2</LogResult>
        <LogResult year="2012" month="1" day="1" hour="6" minute="2" second="39" type="TEMPERATURE_C">22,2</LogResult>
        <LogResult year="2012" month="1" day="1" hour="6" minute="17" second="39" type="TEMPERATURE_C">22,2</LogResult>
        <LogResult year="2012" month="1" day="1" hour="6" minute="32" second="39" type="TEMPERATURE_C">22,2</LogResult>
        <LogResult year="2012" month="1" day="1" hour="6" minute="47" second="39" type="TEMPERATURE_C">22,2</LogResult>
    </LogPoint>
</CONDITIONFILE>
```

The result can be returned in three different formats (as above, i.e. in the BAS native format, as required by the FM system integrated in the IVEL, and in the same form as the simulation results, thereby enabling the interoperability between simulation and monitoring post-processing tools.

2.4.3 Get average Measurement Data

This method is for retrieving the average measurement data like “Room Temperature” of a list of rooms in a specific building by a given time range. This method is needed because it is possible that multiple devices of a same type exist in a room. If this is the case the average of the requested data is calculated.

Method Name	*getAverageMeasurementData (sessionId : string, eeBimId : string, buildingName : string, roomNames : List, startTime : string, endTime : string, function : string) : string*
Relative URL	*/averagemeasurement*
HTTP request	POST

Input	multipart/form-data, application/xml, a session identifier as a xs:string, an eeBim identifier as xs:string, a building name as xs:string, room names of the building as xs:list, a start and end time as xs:string according to the ISO8601 and a function (e.g. “Room Humidity”, “Room Temperature”, “CO2 Level”) as xs:string
Output/Content-Type	application/xml, XML data with all measurements (calculated as average if multiple devices are in the same room) in file format (Laine T., 2012)
Example	See the method getMeasurementData().

2.4.4 Get the Minimum/Maximum values of measurements

Not only can the measurement data be accessed, it is also possible to get the assigned schedule of the measurement data.

Method Name	*getMeasurementSchedule (sessionId : string, eeBimId : string, buildingName : string, roomNames : List, startTime : string, endTime : string, function : string) : string*
Relative URL	*/measurementschedule*
HTTP request	POST
Input	text/plain, a session identifier as a xs:string, an eeBim identifier as xs:string, a building name as xs:string, room names of the building as xs:list, a start and end time as xs:string according to the ISO8601 and a function (e.g. “Room Humidity”, “Room Temperature”, “CO2 Level”) as xs:string
Output/Content-Type	application/xml, XML data with minimum/maximum schedules ordered by room names
Example	Following as URL:

```
~/measurementschedule
```

thereby requesting the “Room Temperature” of the room “A1.2.16” in the building “A1”

returns:

```
<mapdata>
 <map>
  <entry>
   <key xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">A1.2.16</key>
   <value>
    <mapdata>
     <map>
      <entry>
       <key xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">roomtemperature_minimum </key>
```

```
       <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string"> 20.0 </value>
      </entry>
      <entry>
       <key xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">roomtemperature_maximum </key>
       <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string"> 25.0 </value>
      </entry>
     </map>
    </mapdata>
   </value>
  </entry>
 </map>
</mapdata>
```

2.5 Post-Processing Methods

The query results from performed energy simulations have to be visualised in the HESMOS nD Navigator developed in WP 7 of the project. Basis for these visualisations are:

- The developed and/or adapted energy solvers (WP5)
- The monitored energy data from FM (WP4, WP6)
- The eKPIs specified in WP5 and WP9
- The result/visualisation scenarios matrix developed in conjunction with Deliverable D9.2 after intensive end-user/developed discussions and enhanced within the work in WP5 / WP7.

Basically, the following IT approach is considered for the implementation:

1) **Visualisation of various kinds of diagrams**, based on one or more sets of simulation results returned by the IAS method **getResultSet**. These diagrams can show (a) one result set, (b) two or more result sets with the same x-y axes, or (c) two or more result sets with two different y-axes (one on the left side and one on the right side of the shown diagram). Depending on the user input in the nD Navigator, such diagrams can present aggregate plots for all selected locations/spaces, one or more plots for a single location, or one and the same plot for several selected locations for comparison.

2) **Visualisation of snapshot values** in the BIM view of the nD Navigator and/or in a separate window for focused (snapshot) values obtained via the IAS method **getResultSnapshot**.

For each of the above two visualisation types, a single harmonised web service interface for simulation results post-processing is specified. In this way, the nD Navigator can address the IVEL IAS Services in comfortable and flexible manner. For example, the nD Navigator can first collect and store multiple user requests for visualisation concerning different energy performance aspects and then address the IAS Services in a single web service request (or in a single transaction packaging a set of services that provide all data requested by the end user). Alternatively, it is also possible to process user requests step by step. The actual implemented logic is thereby depending only on the nD Navigator GUI and the envisaged user activities, and not on the IAS Services and the Energy Solvers. It should be kept in mind that values/quantities visualised in the BIM view of the Navigator independently for each space (room) have to be requested separate from the values visualised in the different diagram plots.

2.5.1 Get simulation results for diagram visualisation

This method provides uniform access to any simulation results that can be plotted in a diagram. The different diagram types are:

- For single result sets: XYPlot, BarChart, PieChart, FreqDist
- For multiple result sets: XYPlotMulti, XYPlotStacked, BarChartMulti, BarChartStacked, BarChartByType, PieChartByType, ResultTable (not a diagram, but a value table), FreqDistMulti, whereby XYPlotMulti and BarChartMulti can have one or two different Y-Axes.

Multiple plots can be accumulated also by successive calls to this method and intermediate storage of the returned results. It is also possible to use the method for snapshot values, but the more specific method **getResultSnapshot** is more convenient and should therefore be preferred for that purpose.

Method Name	*getResultSet (sessionId : string, eeBimId : string, dataSource : string, resultTypes : list, locationIds : list, startTime : string, endTime : string, unitTime : int, timeMeasure : string, detailLevel : string)*
Relative URL	*/vizplot*
HTTP request	POST
Input	Multipart/form-data,

- *sessionId* and *eeBimId* are of type xs:string and have the same meaning as in all other IAS methods. Their purpose is to uniquely identify where to look for the requested results on the IVEL;
- *dataSource* is of type xs:string; it denotes which of the stored simulation (or monitoring) results is addressed. For simulations, the value should be the same as the one returned from the service which started the respective simulation – this value should be best kept within the Navigator but can also be requested from the IAS with another service request
- *resultType* is a list of xs:string denoting the type(s) of energy result(s), i.e. the specific aspect(s) to be plotted. The number of result sets that will be returned depends on the number of values in this list. If only one value is provided, then a single result set will be returned.

  resultType(s) will be denoted shortly by a number code, e.g.:
  22010 – "Heating"
  22020 – "Cooling"

  etc.

  The full list of possibilities is provided in a separate table.

- *locationIds* is a list of xs:string denoting the GUIDs of the elements for which the result should be returned. This can be the GUID of the whole building (*IfcBuilding*), of one storey (*IfcBuildingStorey*), one or more rooms (*IfcSpace*) or one or more building elements (subclasses of *IfcBuildingElement* –see IFC2x3)
- *startTime*, *endTime* are both of type xs:string; they denote the start / end of the period that should be examined. For time format the ISO 10303-21 specification should be used, i. e. yyyy-mm-ddThh:mm:ss, e. g. 2012-06-26T14:00:00. Seconds will always be ignored and can therefore be set to zeros.
- *unitTime* is of type xs:integer; it denotes the requested time step by the user for which results should be computed. This can be equal or larger than the actual time step used in the simulations. If smaller than the latter, the simulation unit time will be taken.
- *timeMeasure* is of *type* xs:string and denotes the units in which unitTime is measured. Possible values are MIN, HOUR, DAY, WORKDAY, WEEK, MONTH, YEAR
- *detailLevel* can be “separate”, “aggregated” or “both”; this denotes if the result is provided separately for each location/space, aggregated for all locations or both. Aggregated values can be sum/integral or average over the given time step and locations, depending on the selected physical aspect.

Output/Content-Type application/json

A JSON file that contains a JSONArray and multiple JSONObjects each representing a result. A result has the corresponding location identifier “locId”, the start time “startYear”, optionally the energy key performance indicator value “eKPI”, “aggregated” whether the data is aggregated or not, and a JSONObject “cols” defining meta data needed to visualize a chart, like e.g. the time step and time unit. The JSONObject “rows” contains the values per one time step (“x” is the time step and “y” the value). This result file can be relatively simple, as in the example shown below, or fairly complex, depending on the particular request issued. The method can provide output for one or more result types (energy aspects), one or more locations (storeys, rooms, elements), and one or more time steps. Regarding the specific energy aspects (given via the result type codes) the data can be aggregated along two axes: (1) time, and (2) locations, or min/max/average values can be computed. The simplest usage implies one result type, one location and one time interval, which returns a series of Y values for the time steps $X_1 \dots X_N$.

Example

(ouput trimmed for conciseness)

```
[{
    "locId": ["1ueFMxRuX8VOi3xnkXut4D"],
    "startYear": "2012-01-01T00:00:00",
    "eKPI": 0.5,
    "aggregated": false,
    "cols": [{
        "id": "TIME",
        "timeUnit": "h",
        "type": "datetime"
    },
    {
        "id": "VALUE",
        "quantity": "Room Temperature",
        "label": "Room Temperature [degC]",
        "valueUnit": "degC",
        "resultCode": 15010,
        "type": "number"
    }],
    "rows": [{
        "x": 0,
        "y": 0
    },
    {
        "x": 1,
        "y": -1
    },
    {
        "x": 2,
        "y": -1.1
    },
    {
        "x": 3,
        "y": -1
    },
    {
        "x": 4,
        "y": -0.9
    },
    {
        "x": 5,
        "y": -0.9
    },
    {
        "x": 6,
        "y": -0.9
    },
    ,
    …
]
```

2.5.2 Get simulation results for BIM visualisation

This method provides access to simulation results that can be plotted in a BIM view (one or more results per room/space). Numerical output can be additionally provided in a separate view panel. The method provides a specific utility function that enables easier BIM-related requests than the more general *getResultSet* above.

Method Name *getResultSnapshot*
(sessionId : string, eeBimId : string,
dataSource : string, resultTypes : list,
locationIds : list, startTime : string, endTime : string)

Relative URL	*/vizsnapshot*
HTTP request	POST
Input	Multipart/form-data All input values have the same meaning as for getResultPlot. If endTime is equal to startTime, sinlge non-aggregated snapshot values are returned. If endTime is greater than startTime, aggregated values for the given time span are returned, depending on the value of processingId.
Output/Content-Type	application/json Principally, the output of this method is quite similar to *getResultSet* above. The difference to the results obtained from getResultSet is basically in (1) the number of datasets that may be returned, and (2) in the structuring of the result file with regard to the selected location(s), which is simpler and more straightforward to interpret here.

3 Basic Sequence Diagrams

3.1 IVEL Core initialization

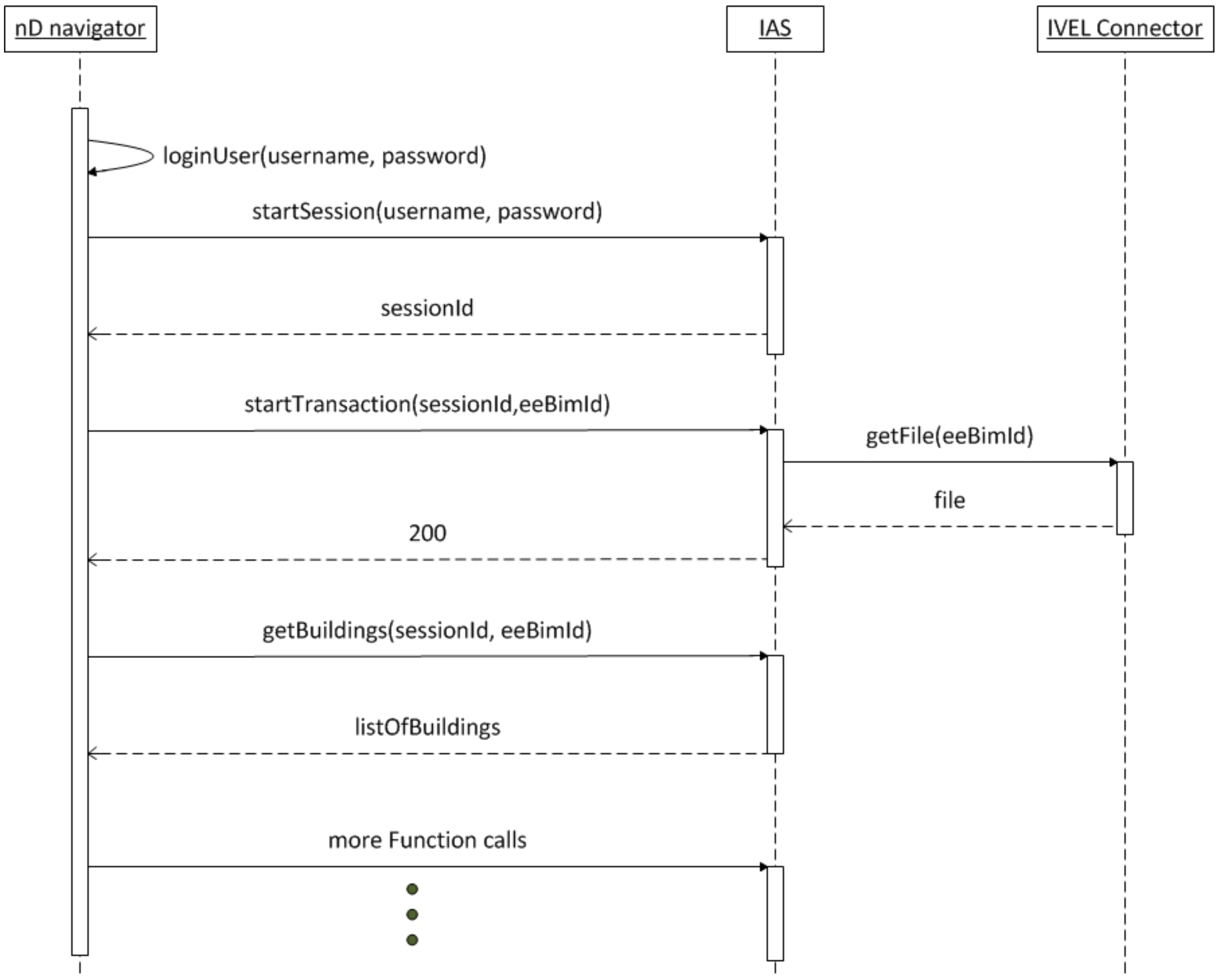

nD navigator
IAS
IVEL Connector
loginUser(username, password)
startSession(username, password)
sessionId
startTransaction(sessionId,eeBimId)
getFile(eeBimId)
file
200
getBuildings(sessionId, eeBimId)
listOfBuildings
more Function calls

Figure 2: Initialization of the IVEL Core

3.2 Start Energy Simulation and get visualisation data

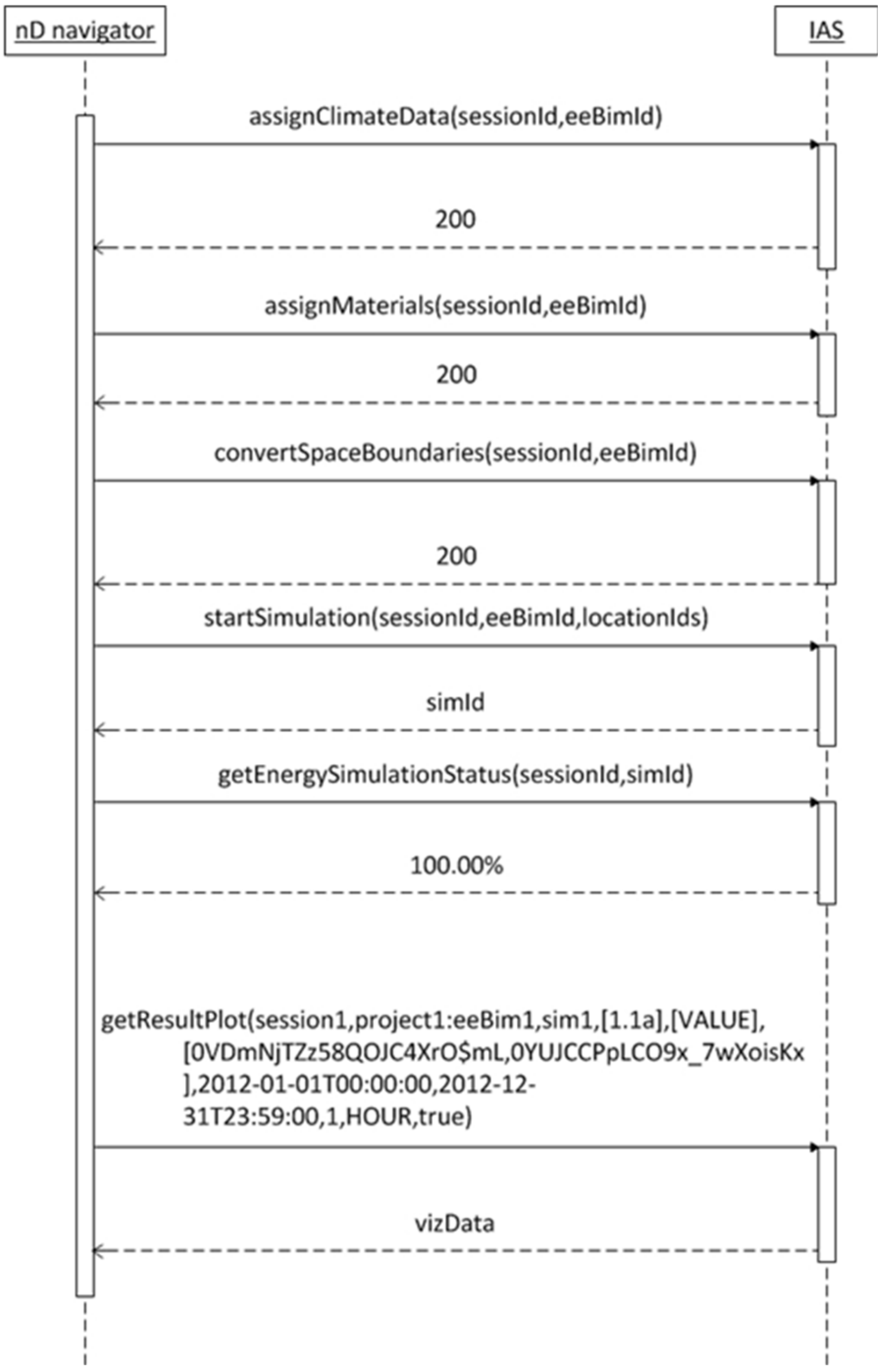

nD navigator
IAS
assignClimateData(sessionId,eeBimId)
200
assignMaterials(sessionId,eeBimId)
200
convertSpaceBoundaries(sessionId,eeBimId)
200
startSimulation(sessionId,eeBimId,locationIds)
simId
getEnergySimulationStatus(sessionId,simId)
100.00%
getResultPlot(session1,project1:eeBim1,sim1,[1.1a],[VALUE],
[0VDmNjTZz58QOJC4XrO$mL,0YUJCCPpLCO9x_7wXoisKx
],2012-01-01T00:00:00,2012-12-
31T23:59:00,1,HOUR,true)
vizData

Figure 3: Start of the energy simulation and return of visualisation data

4 IVEL Core Link Model

The purpose of the IVEL Core Link Model is to provide for loose integration of the multi-model information resources needed to support lifecycle energy performance simulations, monitoring and efficiency assessment. It provides an approach that allows avoiding rigid integration into a single, difficult to maintain model, and takes into account that information/model evolution is spread among different parties, each with its own interests, responsibilities, preferences and permissions. The link model of the IVEL Core contains the inter-linked information that is needed during runtime to satisfy the actual user needs. It involves some information contained in the building model (IFC file) which is necessary for the simulation. This information is energy enhanced through space usage templates, construction templates and material templates, as well as detailed climate and material property data.

4.1 Overview

The principal concept of the link model is given in (Liebich et al. 2011). Therefore, the implementation has started closely to this concept. We implemented links from the IFC entities to templates (climate, materials, space usages etc.) where requirements and parameters are stored. Furthermore we implement links to different versions of one building model e.g. the IFC file with the converted 2nd level space boundaries (Laine et al. 2012) and simulation results and monitoring data.

The link model is based on OWL so that we can model the link types using ontology techniques in a tool called *Protégé* (http://protege.stanford.edu/). In the scope of HESMOS no semantic rules or constraints are implemented, but in the future this can be easily extended and queried e.g. via SPARQL. Such more comprehensive ontology support is planned as part of the work in the EU project ISES (project no. 288819, http://ises.eu-project.info) on the basis of the HESMOS findings.

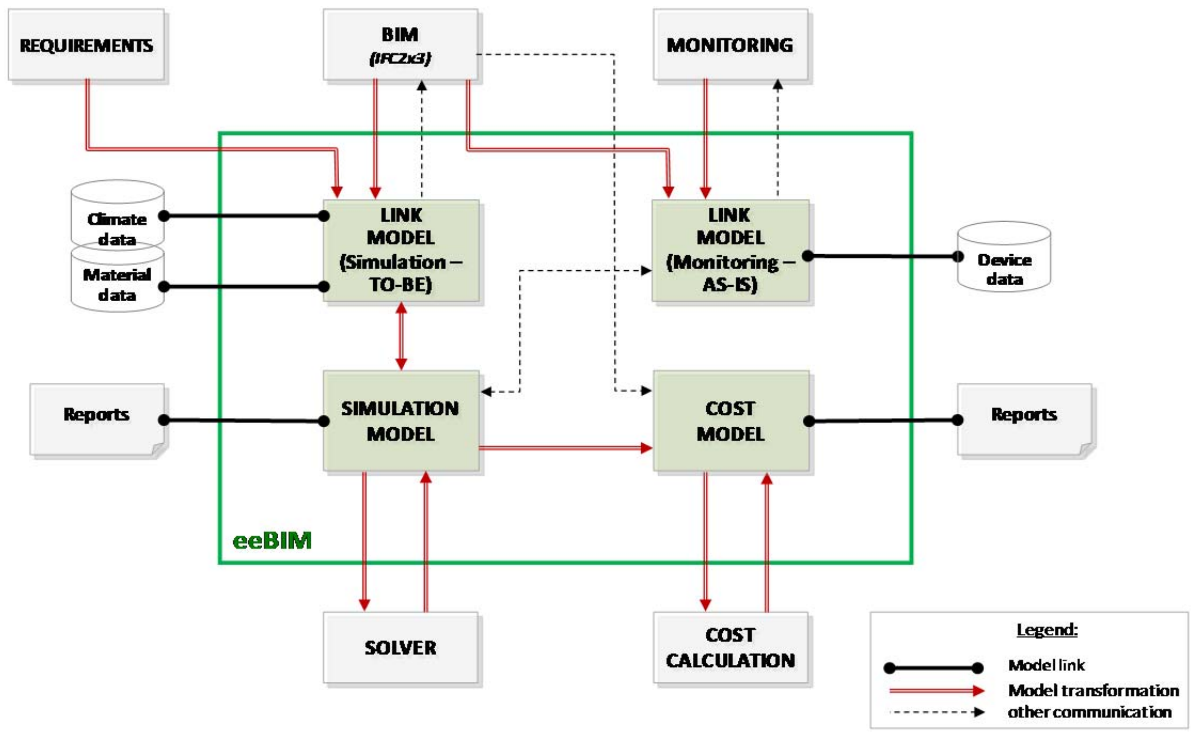


Figure 4: Principal model links (Liebich et al., 2011)

The link model is structured as a meta model which imports multiple other models, called sub models, based on OWL such as ifcOWL (Beetz et al. 2009). It contains link types to refer to each such model. These properties are of the type *owl:ObjectProperty* for linking other elements in the ontology, or *owl:DatatypeProperty* for defining primitive attributes of the elements.

4.2 Meta Model

The Meta model contains classes for importing other ontologies. Currently there exist the following five base concepts defined as OWL classes:

- *Model :* general class for all models;
- *File* : general class for defining a link to a file which does not represent a model but provides some additional input;
- *BIM* : this class is the general class of all models that belongs to the building information modelling process like IFC or eeBIM;
- *IFC* : a subclass of BIM representing the building model in IFC; it links the IFC entities available through ifcOWL that are needed for the simulation (e.g. IfcSpace, IfcBuilding, IfcWall, etc.);
- *eeBIM* : the class which links eeBIM entities (available through a native eeBIM ontology), like templates for climate, materials, wall/floor construction etc.

The following object properties are defined:

- *evolutesTo*; this relation links newly created elements in the eeBim workflow. For example, new created elements are 2nd level space boundaries which are the outcome of a geometrical analysis of one 1st level space boundary;
- *isEvolutedFrom*; is the inverse relation of *evolutesTo*;
- *hasIfc2x3Elements*; this relation inter-links all needed IFC entities like rooms, buildings, storeys, building elements etc. with the *IFC* class;
- *hasEeBimElements*; this relation inter-links all eeBim entities with the *eeBim* class so that IFC entities, linked with the *IFC* class, can be connected;
- *hasLinkTo*; defines a link of a model to an additional file;
- *linkedFrom*; inverse relation of *hasLinkTo*;
- *isRelatedBy*; defines a link to another model;
- *isRelatedFrom*; inverse relation of *isRelatedBy*.

The following data properties are defined:

- *modelId*; defines a unique identifier for models;
- *filePath; defines* the file path to a model or a file.

Figure 5 presents the link model in a graphical way. The top class of all classes is *owl:Thing*. The inheritance of the classes is defined through the *is-a* relationship (solid lines in magenta). Individuals which are representing elements in the link model are connected with the *instance-of* relationship (dashed lines in blue colour). In Figure 5 two individuals are shown (there exists much more individuals but this view was selected to give a brief overview), one is the *IFC* model and one the *eeBIM* individual. The *IFC* class is related to the *IfcRoot* class in the ifcOWL and therefore the relationship to all subclasses (*IfcWallStandardCase*, *IfcSpace*, *IfcWindow, etc.*) can be covered.

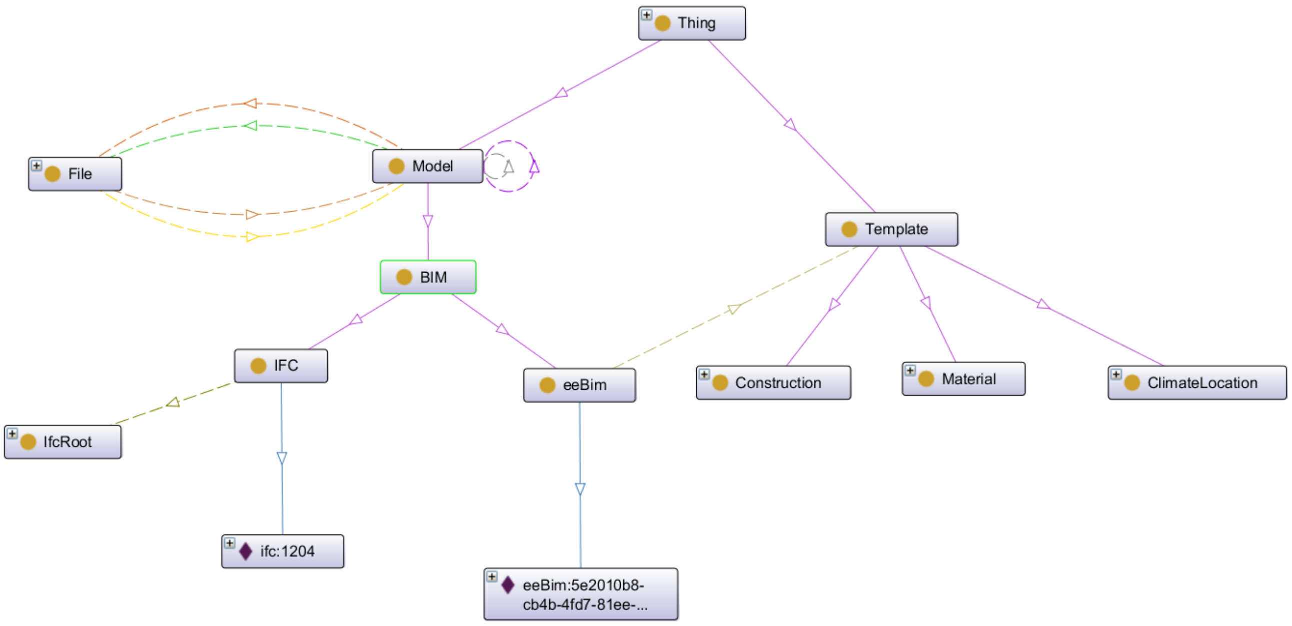
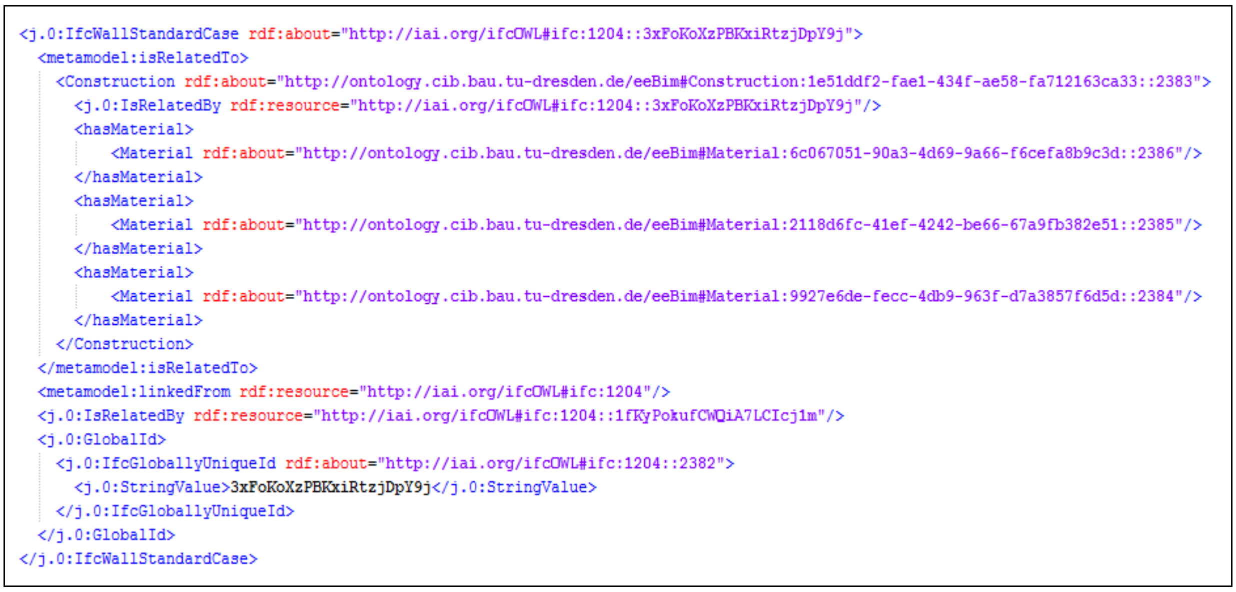


Figure 5: Structure of the Meta model

Figure 6 presents an example of a wall entity in the link model. This wall has the GlobalId "3xFoKoXzPBKxiRtzjDpY9j" and is provided through the ifcOWL model with the id "1204". The URI is a concatenation of these identifiers. It is a IfcWallStandardCase entity and is *linkedFrom* the *IFC* model individual in the meta model. The *isRelatedBy* link specifies that it has a relationship to an IfcSpace entity with the GlobalId "1fKyPokufCWQiA7LCIcj1m". A link to the construction template with the identifier "2208" is provided through *isRelatedTo* and inter-links to the eeBim model with the identifier "1e51ddf2-fae1-434f-ae58-fa712163ca33".



```
<j.0:IfcWallStandardCase rdf:about="http://iai.org/ifcOWL#ifc:1204::3xFoKoXzPBKxiRtzjDpY9j">
  <metamodel:isRelatedTo>
    <Construction rdf:about="http://ontology.cib.bau.tu-dresden.de/eeBim#Construction:1e51ddf2-fae1-434f-ae58-fa712163ca33::2383">
      <j.0:IsRelatedBy rdf:resource="http://iai.org/ifcOWL#ifc:1204::3xFoKoXzPBKxiRtzjDpY9j"/>
      <hasMaterial>
          <Material rdf:about="http://ontology.cib.bau.tu-dresden.de/eeBim#Material:6c067051-90a3-4d69-9a66-f6cefa8b9c3d::2386"/>
      </hasMaterial>
      <hasMaterial>
          <Material rdf:about="http://ontology.cib.bau.tu-dresden.de/eeBim#Material:2118d6fc-41ef-4242-be66-67a9fb382e51::2385"/>
      </hasMaterial>
      <hasMaterial>
          <Material rdf:about="http://ontology.cib.bau.tu-dresden.de/eeBim#Material:9927e6de-fecc-4db9-963f-d7a3857f6d5d::2384"/>
      </hasMaterial>
    </Construction>
  </metamodel:isRelatedTo>
  <metamodel:linkedFrom rdf:resource="http://iai.org/ifcOWL#ifc:1204"/>
  <j.0:IsRelatedBy rdf:resource="http://iai.org/ifcOWL#ifc:1204::1fKyPokufCWQiA7LCIcj1m"/>
  <j.0:GlobalId>
    <j.0:IfcGloballyUniqueId rdf:about="http://iai.org/ifcOWL#ifc:1204::2382">
      <j.0:StringValue>3xFoKoXzPBKxiRtzjDpY9j</j.0:StringValue>
    </j.0:IfcGloballyUniqueId>
  </j.0:GlobalId>
</j.0:IfcWallStandardCase>
```

Figure 6: Example of a wall represented in the link model

4.3 Submodels

There currently exist two sub models, the ifcOWL ontology and the newly developed eeBIM ontology. While the ifcOWL represents the IFC2x3 model, the eeBIM ontology provides templates and other simulation-related information.

4.3.1 ifcOWL

The ifcOWL ontology is described in (Beetz et al., 2009). We are currently using the whole IFC2x3 scheme represented in OWL. However, in the future we will use only a part of the whole schema because the energy simulation doesn't need all information in an IFC file. For instance, the geometry provides most of the entities in a file, because it is represented in very complex manner in IFC. The simulation solvers need explicit geometry information like height, area, width etc. of building elements or spatial elements, which can be created through the computational analysis of the given implicit coordinates and representation profiles. Such information will overload the link model and make it very slow. We are only using sub classes of *IfcRoot* and will create links from such entities to energy enhanced information like eeTemplates. If a user will change a template or select a new one, a new link to the selected template is respectively created.

4.3.2 eeBIM Model

The eeBim model (see Appendix V) is a light-weight model which can easily be extended to overcome the requirements of the energy performance simulations.

The eeBim model contains four classes:

- *Template*; the general class for all templates;
- *ClimateLocation*; provides the information about climate of a location;
- *Construction*; provides the information about a construction. A construction consists of multiple materials. It is possible to create links between a construction and the material templates.
- *Material*; represents information about a material. This material is taken out of the material catalogue of the energy solvers.

One object property is defined:

- *hasMaterial*; describes a relationship of a construction and one or many material templates.

Two data properties are defined:

- *id*; describes the identifier of a template;
- *pathToTemplate*; defines the file path to a template.

Figure 7 presents an excerpt of a construction template in an example Link Model. It is the template which is linked from the wall (central in the figure) of the example in Figure 6. The construction provides a layered wall of three materials. The link isRelatedBy represents the relationship to the wall. The hasMaterial link interlinks the material templates. Furthermore, the link to the IfcSpace objects and the overall IFC model is provided.

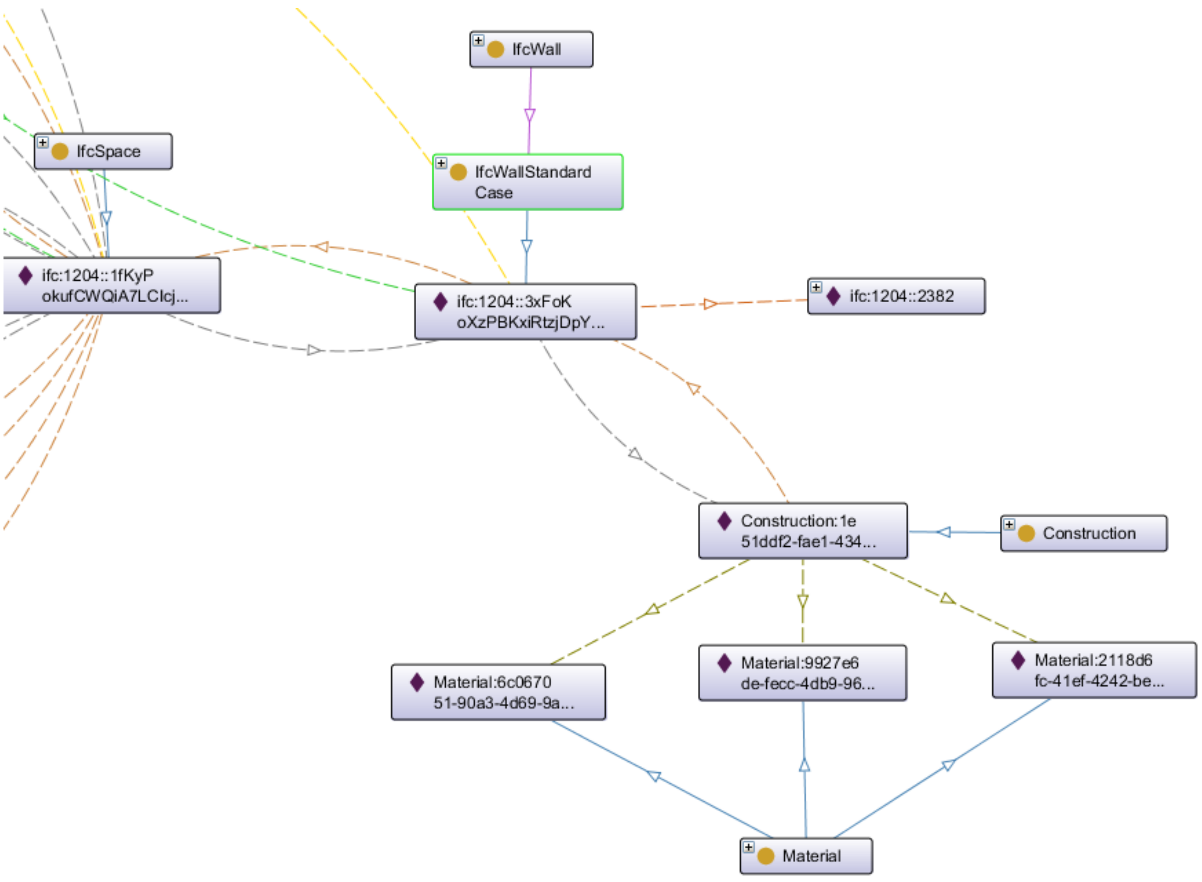

IfcWall
IfcSpace
IfcWallStandard Case
ifc:1204::1fKyP okufCWQiA7LClcj...
ifc:1204::3xFoK oXzPBKxiRtzjDpY...
ifc:1204::2382
Construction:1e 51ddf2-fae1-434...
Construction
Material:6c0670 51-90a3-4d69-9a...
Material:9927e6 de-fecc-4db9-96...
Material:2118d6 fc-41ef-4242-be...
Material

Figure 7: Graphical representation of an example link model with eeBim information

5 Conclusions

At this stage, we have implemented most of the required interoperability methods and the link model as basis of the targeted overall lifecycle integration on the HESMOS platform. The implementation of the IVEL Core made very good progress and the access to the used models is fully provided via the developed web services called IAS. It is possible to upload building models and get the involved information like rooms, façade elements (also of a specific direction), and more. With that information the user can select the locations he is interested in and start an energy performance simulation, list currently running simulations and request various post-processing actions on the simulation results to take a look on them in suitable synthetic form on the IVEL nD navigator. It is also possible to ask for real measured data, as e.g. for the HESMOS pilot project "Pforzheim School", for comparing simulation results and real data. The measured data is requested from the Building Automation System web service developed in WP 4. The IVEL Core is preparing this raw data by analysing the user queries and returns a structured answer to the tools of the end users. The basis for tackling such large distributed amounts of data is the developed service-oriented architecture and the link model defined as native ontology in OWL.

Thus, the following progress towards the objectives of WP 8 was made. The IVEL is based on a SOA and currently integrates several components and applications like the nD Navigator and the monitoring and energy computing services. The link model based on OWL is the major integrative interoperability method inside in the IVEL Core. To the external components it provides the IAS as a REST-based web service and thus can be used in every application which needs its functionality.

Since the link model is developed on the basis of OWL, we have the possibility to check if an ontology-based simulation model can be developed and used within it. This can be a very good approach for varying simulation parameters and for checking the selected parameters before simulation. Therefore it has to be enhanced semantically and can be used in a more general and not application-dependent way. This is work planned in the related EU project ISES inaugurated in December 2011 (ISES Consortium, 2012).

Literature Sources

Beetz, J., van Leeuwen, J. and de Vries, B. 2009. IfcOWL: A case of transforming EXPRESS schemas into ontologies. *Artificial Intelligence for Engineering Design, analysis and Manufacturing,* 23, 2009.

buildingSMART International Modeling Support Group. 2009. *IFC 2x Edition 3 Model Implementation Guide*.

Fielding, R. T. 2000. *Architectural Styles and the Design of Network-based Software Architectures.* University of California, Irvine, USA.

http://protege.stanford.edu/. protégé. [Online] Stanford University School of Medicine, USA [Cited: August 23, 2012.]

ISES Consortium. 2012. http://ises.eu-project.info/. [Online] © ISES Consortium, Brussels, EU [Cited: February 27, 2012.] http://ises.eu-project.info/.

Katranuschkov P., Baumgärtel K., Guruz R., Kaiser J., Grunewald J., Hensel B., Steinmann R., Zellner R., Laine T., Hänninnen R. 2011. *HESMOS Deliverable D2.2: HESMOS Architecture.* © HESMOS Consortium, Brussels., EU.

Laine T., Forns-Samso F., Kukkonen E., Geißler M. 2012. *HESMOS Deliverable D6.2: Web service and interface client for interoperable energy management support.* © HESMOS Consortium, Brussels, EU.

Laine, T. et al. 2012. *HESMOS Deliverable D8.1: Configuration and Deployment of the Developed Basic SOA System.* © HESMOS Consortium, Brussels, EU.

Liebich, T. et al. 2011. *D2.1: BIM Enhancement Specification.* © HESMOS Consortium, Brussels., EU.

Ploennigs J., Dibowski H., Röder A., Kabitzsch K., Hensel B., Baumgärtel K. 2012. *HESMOS Deliverable D4.2: Full prototype of ICT system integration and intelligent access services.* HESMOS Consortium, Brussels, EU.

Richardson L., Ruby S. 2007. *RESTful Web Services*. O'Reilly Media, CA, USA.

Appendix I: Acronyms

BCF	**B**IM **C**ollaboration **F**ormat
BIM	**B**uilding **I**nformation **M**odel
HTML	**H**yper**T**ext **M**arkup **L**anguage
HTML	**H**yper**T**ext **M**arkup **L**anguage
HTTP:	**H**yper**T**ext **T**ransfer **P**rotocol
IAS:	**I**ntelligent **A**ccess **S**ervices
IFC:	**I**ndustry **F**oundation **C**lasses
IVEL:	**I**ntegrated **V**irtual **E**nergy **L**aboratory
JSON	**J**ava**S**cript **O**bject **N**otation
REST	**Re**presentational **S**tate **T**ransfer
SOAP	**S**imple **O**bject **A**ccess **P**rotocol
UML	**U**nified **M**arkup **L**anguage
URI	**U**niform **R**esource **I**dentifier
URI	**U**niform **R**esource **L**ocator
WADL	**W**eb **A**pplication **D**escription **L**anguage
XML	E**x**tended **M**arkup **L**anguage

Appendix II: Used Technologies in the IVEL Core

Technology	Role	Reference (URL)
OSGi	Module and service platform	http://www.osgi.org/
SOAP	Framework for exchanging web service data	http://www.w3.org/TR/soap/
REST	Web service architectural style providing an alternative framework to SOAP	--
Axis 2	Engine for SOAP and WSDL	http://axis.apache.org/axis2/java/core/
Apache Jersey	Engine for REST	http://jersey.java.net/
Apache Camel	Rule-based routing engine for linking and parsing of data formats	http://camel.apache.org/
Spring	Business application framework providing simple and flexible integration of other technologies	http://www.springsource.org/
Spring dm	Extension of Spring for OSGi	http://www.springsource.org/osgi
Hibernate	Object-relational mapping	http://www.hibernate.org/
HSQLDB	Relational database management system	http://hsqldb.org/
Log4j	Application logging in the development and production phase	http://logging.apache.org/log4j/
Eclipse RAP	GUI framework for rich internet application	http://www.eclipse.org/rap/
JENA	Ontology implementation for Java	http://incubator.apache.org/jena/
SPARQL	Query language based on RDF	http://jena.apache.org
Underlying Models and Specifications		
OWL	Knowledge representation language for ontologies	http://www.w3.org/TR/owl-ref/
RDF	Metadata model for enriched description of web resources	http://www.w3.org/TR/rdf-primer/
CSV	Data format to store data in plain text	--
IFC	The ISO standardized BIM data model	http://buildingsmart-tech.org/specifications/ifc-overview
JSON	Data format	http://www.json.org/
WSDL	Web Service Definition Language	http://www.w3.org/TR/wsdl