.15926 Editor

Version 1.5beta

Sample Mapping and Adapter Prototyping

Walk-through Guide

13 July 2014

This document will guide you through the process of data mapping and adapter prototyping for the sample plant process data using .15926 Editor. To follow it step-by-step you have to download the Editor from http://techinvestlab.ru/dot15926Editor (this guide is valid for versions starting from **1.5beta**).

Initial data set and all information required to reproduce described transformations are included in the folder **dot15926Editor15beta\samples\ProcessDiagram** and its subfolders. All folder references below are given related to it.

Prototyping process described here depends on the MS Excel data transformation capabilities. Data are preprocessed in spreadsheets and imported into the ISO 15926 RDF format using the Editor's built-in spreadsheet import. Fully functional adapter independent of the inherent restrictions of this approach can be implemented at a later stage when mapping and data transformation are prototyped and debugged.

1. Source data

We will be working with a single high-level process diagram. It is prepared with the software from one of the major engineering software vendors. The data is exported using the standard export functionality of the tool. Look in **\Source** folder to see the diagram in **PF-PFB-Plant.pdf** file and two spreadsheets with exported data **EquipmentWithBaseObjectAndAttributeHeight.xls** and **ProcessUnits-Connectors.xls**.

2. Modelling conventions and project setup

To make this example short and easily understandable even for novice data modellers we will keep some major modelling choices very simple or probable oversimplified.

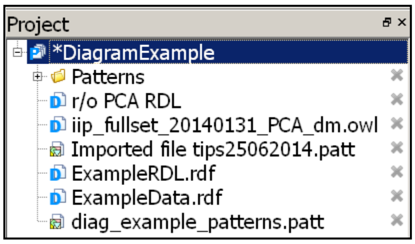
All objects on the diagram will be modeled as individual physical objects. The most rigorous modeling at the initial stages of the plant lifecycle can require seeing them as classes of activities.

The model will not include temporal parts of modeled individuals, temporal nature and temporal boundaries of objects will not be modeled.

The data prepared with such modeling conventions can be used for one-time data exchange between engineering tools, but can not be used for lifecycle data storage.

According to the choices made above we will use templates for individuals from the IIP template set (local copy of templates available from http://posccaesar.org/sandbox/p8iwg/) and IIP project Template Information Patterns (project page http://iringug.org/wiki/index.php?title=ISO_15926_Information_Patterns_%28IIP%29, TIP Manager http://iringsandbox.org:8080/tip/tipmanager, patterns imported into the Editor from database backup http://www.iringsandbox.org/bak/tips.mdb).

Of course we will use PCA Reference Data Library (file for local use available from http://rds.posccaesar.org/downloads/PCA-RDL.owl.zip). Please download the file, unzip it and register its location in the Editor settings.



Project

*DiagramExample
- Patterns
- r/o PCA RDL
- iip_fullset_20140131_PCA_dm.owl
- Imported file tips25062014.patt
- ExampleRDL.rdf
- ExampleData.rdf
- diag_example_patterns.patt

Go to the folder **\ProjectData** and open project file **DiagramExample.15926** in the Editor. You will see the project composed from the following data sources:

PCA RDL – POSCCaesar RDL (opened as read-only from the Editor settings);

iip_fullset_20140131_PCA_dm.owl – IIP template set (initial set and specialized templates);

tips25062014.patt – IIP TIPs file;

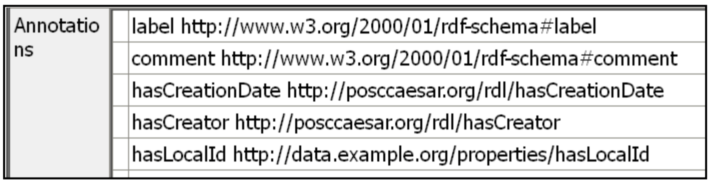
diag_example_patterns.patt – project-specific patterns;

ExampleRDL.rdf – project-specific reference data library (now empty);

ExampleData.rdf – project data file (now empty).

We will use the following two namespaces: http://data.example.org/rdl/ for project-specific reference data and http://data.example.org/project/ for project data. You can find them registered in the Properties of respective data sources.

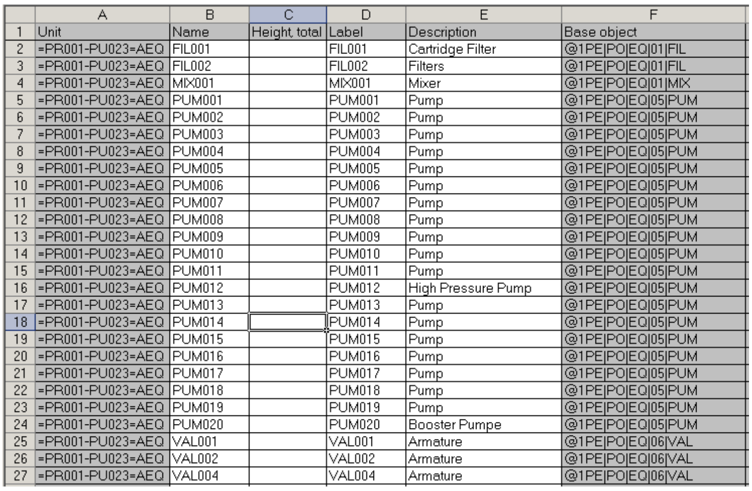
One project specific annotation *hasLocalId* is registered in the Properties of the project to record IDs of reference data entities and project objects used in the native application. Other annotation properties in the project are standard RDF/RDFS properties and properties used in PCA RDL.



Annotations	
	label http://www.w3.org/2000/01/rdf-schema#label
	comment http://www.w3.org/2000/01/rdf-schema#comment
	hasCreationDate http://posccaesar.org/rdl/hasCreationDate
	hasCreator http://posccaesar.org/rdl/hasCreator
	hasLocalId http://data.example.org/properties/hasLocalId

3. Preparing RDL from Equipment list

Looking at exported project data file **EquipmentWithBaseObjectAndAttributeHeight.xls** we can see that equipment item types can be derived from *Base object* column (F) where some type of internal ID of authoring system is located. Column *Decscription* (E) allows us to deduce the names of base objects from the descriptions of project objects.

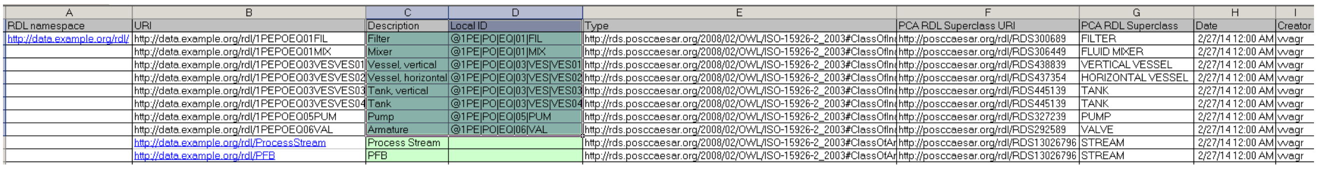


	A	B	C	D	E	F
1	Unit	Name	Height, total	Label	Description	Base object
2	=PR001-PU023=AEQ	FIL001		FIL001	Cartridge Filter	@1PE|PO|EQ|01|FIL
3	=PR001-PU023=AEQ	FIL002		FIL002	Filters	@1PE|PO|EQ|01|FIL
4	=PR001-PU023=AEQ	MIX001		MIX001	Mixer	@1PE|PO|EQ|01|MIX
5	=PR001-PU023=AEQ	PUM001		PUM001	Pump	@1PE|PO|EQ|05|PUM
6	=PR001-PU023=AEQ	PUM002		PUM002	Pump	@1PE|PO|EQ|05|PUM
7	=PR001-PU023=AEQ	PUM003		PUM003	Pump	@1PE|PO|EQ|05|PUM
8	=PR001-PU023=AEQ	PUM004		PUM004	Pump	@1PE|PO|EQ|05|PUM
9	=PR001-PU023=AEQ	PUM005		PUM005	Pump	@1PE|PO|EQ|05|PUM
10	=PR001-PU023=AEQ	PUM006		PUM006	Pump	@1PE|PO|EQ|05|PUM
11	=PR001-PU023=AEQ	PUM007		PUM007	Pump	@1PE|PO|EQ|05|PUM
12	=PR001-PU023=AEQ	PUM008		PUM008	Pump	@1PE|PO|EQ|05|PUM
13	=PR001-PU023=AEQ	PUM009		PUM009	Pump	@1PE|PO|EQ|05|PUM
14	=PR001-PU023=AEQ	PUM010		PUM010	Pump	@1PE|PO|EQ|05|PUM
15	=PR001-PU023=AEQ	PUM011		PUM011	Pump	@1PE|PO|EQ|05|PUM
16	=PR001-PU023=AEQ	PUM012		PUM012	High Pressure Pump	@1PE|PO|EQ|05|PUM
17	=PR001-PU023=AEQ	PUM013		PUM013	Pump	@1PE|PO|EQ|05|PUM
18	=PR001-PU023=AEQ	PUM014		PUM014	Pump	@1PE|PO|EQ|05|PUM
19	=PR001-PU023=AEQ	PUM015		PUM015	Pump	@1PE|PO|EQ|05|PUM
20	=PR001-PU023=AEQ	PUM016		PUM016	Pump	@1PE|PO|EQ|05|PUM
21	=PR001-PU023=AEQ	PUM017		PUM017	Pump	@1PE|PO|EQ|05|PUM
22	=PR001-PU023=AEQ	PUM018		PUM018	Pump	@1PE|PO|EQ|05|PUM
23	=PR001-PU023=AEQ	PUM019		PUM019	Pump	@1PE|PO|EQ|05|PUM
24	=PR001-PU023=AEQ	PUM020		PUM020	Booster Pumpe	@1PE|PO|EQ|05|PUM
25	=PR001-PU023=AEQ	VAL001		VAL001	Armature	@1PE|PO|EQ|06|VAL
26	=PR001-PU023=AEQ	VAL002		VAL002	Armature	@1PE|PO|EQ|06|VAL
27	=PR001-PU023=AEQ	VAL004		VAL004	Armature	@1PE|PO|EQ|06|VAL

Only 8 base objects with different IDs are used in this file:

Description	Local ID
Filter	@1PE|PO|EQ|01|FIL
Mixer	@1PE|PO|EQ|01|MIX
Vessel, vertical	@1PE|PO|EQ|03|VES|VES01
Vessel, horizontal	@1PE|PO|EQ|03|VES|VES02
Tank, vertical	@1PE|PO|EQ|03|VES|VES03
Tank	@1PE|PO|EQ|03|VES|VES04
Pump	@1PE|PO|EQ|05|PUM
Armature	@1PE|PO|EQ|06|VAL

We'll record them to the project-specific reference data library and link them to appropriate PCA RDL reference data classes. To do this we prepare a spreadsheet and import it using the Editor spreadsheet adapter. Go to the folder **\ForImport** and open **ProjectRD.xls** file.



A	B	C	D	E	F	G	H	I
RDL namespace	URI	Description	Local ID	Type	PCA RDL Superclass URI	PCA RDL Superclass	Date	Creator
http://data.example.org/rdl/	http://data.example.org/rdl/1PEPOEQ01FIL	Filter	@1PE|PO|EQ|01|FIL	http://rds.posccaesar.org/2008/02/OWL/ISO-15926-2_2003#ClassOfIn	http://posccaesar.org/rdl/RDS300689	FILTER	2/27/14 12:00 AM	wagr
	http://data.example.org/rdl/1PEPOEQ01MIX	Mixer	@1PE|PO|EQ|01|MIX	http://rds.posccaesar.org/2008/02/OWL/ISO-15926-2_2003#ClassOfIn	http://posccaesar.org/rdl/RDS306449	FLUID MIXER	2/27/14 12:00 AM	wagr
	http://data.example.org/rdl/1PEPOEQ03VESVES01	Vessel, vertical	@1PE|PO|EQ|03|VES|VES01	http://rds.posccaesar.org/2008/02/OWL/ISO-15926-2_2003#ClassOfIn	http://posccaesar.org/rdl/RDS438839	VERTICAL VESSEL	2/27/14 12:00 AM	wagr
	http://data.example.org/rdl/1PEPOEQ03VESVES02	Vessel, horizontal	@1PE|PO|EQ|03|VES|VES02	http://rds.posccaesar.org/2008/02/OWL/ISO-15926-2_2003#ClassOfIn	http://posccaesar.org/rdl/RDS437354	HORIZONTAL VESSEL	2/27/14 12:00 AM	wagr
	http://data.example.org/rdl/1PEPOEQ03VESVES03	Tank, vertical	@1PE|PO|EQ|03|VES|VES03	http://rds.posccaesar.org/2008/02/OWL/ISO-15926-2_2003#ClassOfIn	http://posccaesar.org/rdl/RDS445139	TANK	2/27/14 12:00 AM	wagr
	http://data.example.org/rdl/1PEPOEQ03VESVES04	Tank	@1PE|PO|EQ|03|VES|VES04	http://rds.posccaesar.org/2008/02/OWL/ISO-15926-2_2003#ClassOfIn	http://posccaesar.org/rdl/RDS445139	TANK	2/27/14 12:00 AM	wagr
	http://data.example.org/rdl/1PEPOEQ05PUM	Pump	@1PE|PO|EQ|05|PUM	http://rds.posccaesar.org/2008/02/OWL/ISO-15926-2_2003#ClassOfIn	http://posccaesar.org/rdl/RDS327239	PUMP	2/27/14 12:00 AM	wagr
	http://data.example.org/rdl/1PEPOEQ06VAL	Armature	@1PE|PO|EQ|06|VAL	http://rds.posccaesar.org/2008/02/OWL/ISO-15926-2_2003#ClassOfIn	http://posccaesar.org/rdl/RDS292589	VALVE	2/27/14 12:00 AM	wagr
	http://data.example.org/rdl/ProcessStream	Process Stream		http://rds.posccaesar.org/2008/02/OWL/ISO-15926-2_2003#ClassOfAr	http://posccaesar.org/rdl/RDS13026796	STREAM	2/27/14 12:00 AM	wagr
	http://data.example.org/rdl/PFB	PFB		http://rds.posccaesar.org/2008/02/OWL/ISO-15926-2_2003#ClassOfAr	http://posccaesar.org/rdl/RDS13026796	STREAM	2/27/14 12:00 AM	wagr

Column A contains project RDL namespace we will use to form URIs for new entities.

We will use unique IDs (fragment IDs) for reference data entities built from their internal IDs – this will allow us to use *Base object* field in the exported spreadsheet to determine the type of the object using only Excel data processing. Obviously it will be better to use UUID generator to guarantee global uniqueness of fragment IDs. When spreadsheet adapter is prototyped and debugged, UUID generation can be implemented in a dedicated adapter code free from restrictions of the Editor's built-in spreadsheet import.

Column B contains Excel formula designed to remove all symbols not allowed in URI from Local ID and concatenate resulting fragment ID with the namespace:

=SUBSTITUTE(SUBSTITUTE(CONCATENATE(A$2;D2);"|";"");"@";"")

We preserve Description and Local ID in columns C and D to import them as annotation properties.

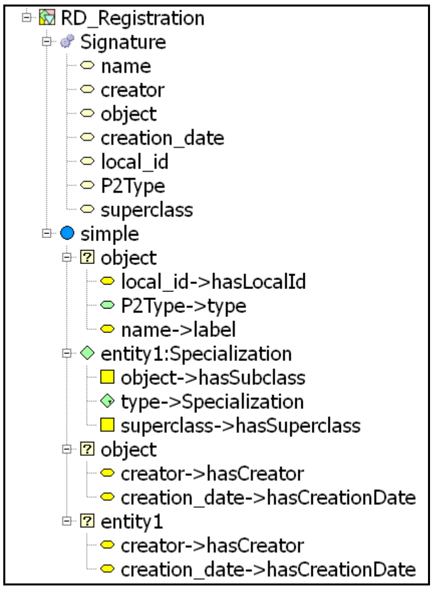
Column E contains Part 2 type of the RD entity.

Column F contains URI of PCA RDL superclass and column G contains its name for easy reference.

Columns H and I contain metadata we'll use in our project RDL – for the demonstration purposes we'll import only the date of entity creation and ID of the creator.

<u>4. Defining pattern for project RDL</u>

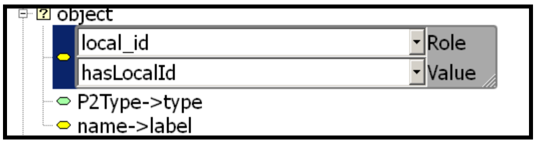
To import RDL spreadsheet content into the project reference data library we need a pattern which will describe the structure of the spreadsheet. Open **diag_example_patterns.patt** (project-specific pattern data source) in the Editor, find *RD_Registration* pattern and fully unfold all its nodes.



```
RD_Registration
  Signature
    name
    creator
    object
    creation_date
    local_id
    P2Type
    superclass
  simple
    object
      local_id->hasLocalId
      P2Type->type
      name->label
    entity1:Specialization
      object->hasSubclass
      type->Specialization
      superclass->hasSuperclass
    object
      creator->hasCreator
      creation_date->hasCreationDate
    entity1
      creator->hasCreator
      creation_date->hasCreationDate
```

The pattern has a signature that corresponds to the columns of the imported spreadsheet. A single mapping to templates and properties (named *simple*) is defined for this pattern.

It maps *local_id*, *P2Type* and *name* to the appropriate annotation and object properties of an *object* (double click each mapping node to see property used).



```
object
  local_id    Role
  hasLocalId  Value
  P2Type->type
  name->label
```

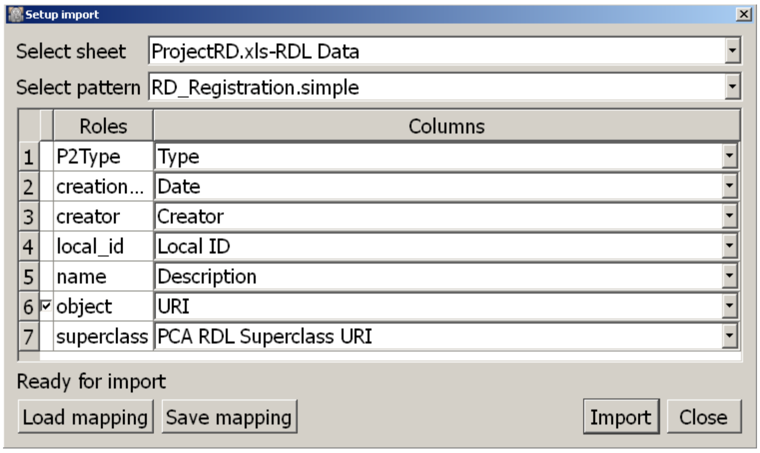
The pattern also describes one additional Specialization entity *entity1* (an instance of Part 2 Specialization type) which describes relation between new entity and its PCA RDL superclass. We will create project RDL using Part 2 type instances, according to the current PCA RDL modelling rules.

The pattern contains two more parts, assigning our metadata properties to the same entities *object* and *entity1*. Separate parts are required to allow repeated imports of the same spreadsheet, please refer to the documentation for more detail on the work of the Editor's built-in spreadsheet import.

5. Importing RDL

Check that **ProjectRD.xls** file is open in Excel on your computer and **ExampleRDL** panel is an active panel in the Editor. Call pattern import extension (*Build patterns from MS Excel* in *Extensions* menu).

Select sheet *RDL Data* and load mapping **rd_mapping.json** from **\Scripts** folder. Correspondence between pattern roles described above and spreadsheet columns is established. The check mark at the *object* role indicates that entity in this role should be created with URI recorded in the corresponding column (all other URIs for new entities will be generated by the Editor).



Select sheet ProjectRD.xls-RDL Data

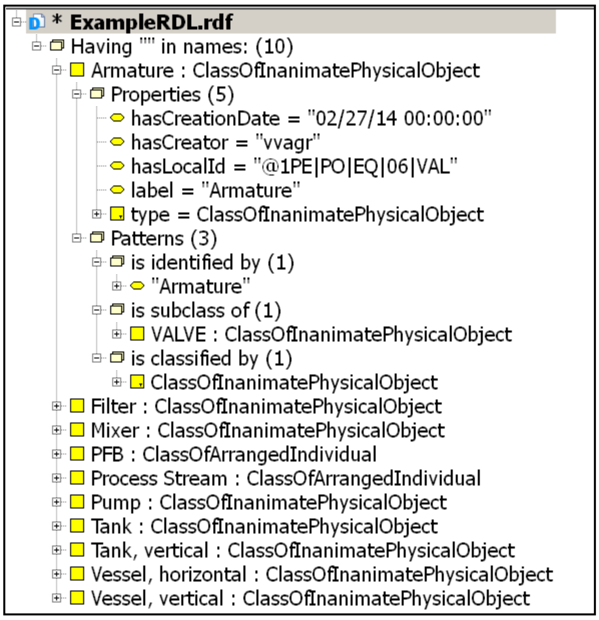
Select pattern RD_Registration.simple

	Roles	Columns
1	P2Type	Type
2	creation...	Date
3	creator	Creator
4	local_id	Local ID
5	name	Description
6	☑ object	URI
7	superclass	PCA RDL Superclass URI

Ready for import

Import data. Sometimes the adapter will return an error code in the console indicating problems with Excel ODBC connection. Please make sure that the cell selected in the spreadsheet is an empty cell out of the range of data prepared for import.

Check the content of the **ExampleRDL**. Look also at the spreadsheet. It now contains URIs of all entities created during the import (to the right of the main data block), which allows incremental import – you can add more entities to it and repeat the process.



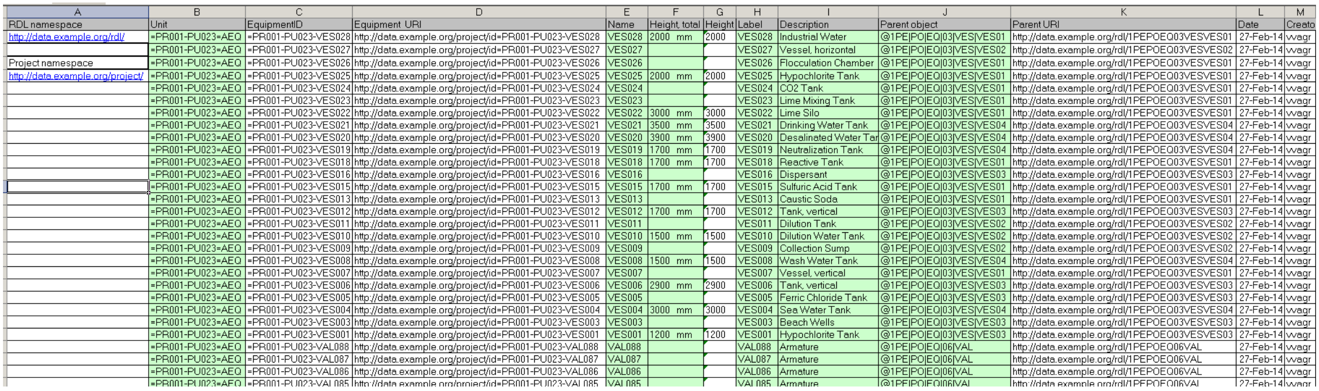
- * ExampleRDL.rdf
  - Having "" in names: (10)
    - Armature : ClassOfInanimatePhysicalObject
      - Properties (5)
        - hasCreationDate = "02/27/14 00:00:00"
        - hasCreator = "vvagr"
        - hasLocalId = "@1PE|PO|EQ|06|VAL"
        - label = "Armature"
        - type = ClassOfInanimatePhysicalObject
      - Patterns (3)
        - is identified by (1)
          - "Armature"
        - is subclass of (1)
          - VALVE : ClassOfInanimatePhysicalObject
        - is classified by (1)
          - ClassOfInanimatePhysicalObject
    - Filter : ClassOfInanimatePhysicalObject
    - Mixer : ClassOfInanimatePhysicalObject
    - PFB : ClassOfArrangedIndividual
    - Process Stream : ClassOfArrangedIndividual
    - Pump : ClassOfInanimatePhysicalObject
    - Tank : ClassOfInanimatePhysicalObject
    - Tank, vertical : ClassOfInanimatePhysicalObject
    - Vessel, horizontal : ClassOfInanimatePhysicalObject
    - Vessel, vertical : ClassOfInanimatePhysicalObject

6. Preparing Equipment list

Now we will prepare for import an exported project data file **EquipmentWithBaseObjectAndAttributeHeight.xls**. We have to record URIs for entities and preprocess information about the one property we have.

Go to the folder **\ForImport** and open **Equipment_for_Import.xls** file.

We will again use internal IDs as unique IDs (fragment IDs) in project data item URIs – this will allow us to connect to the second project export file at the next stage. Again it will be better to use UUID generator to guarantee global uniqueness of fragment IDs, and UUID generation can be implemented at a later stage in an adapter which is not dependant on standard Excel capabilities or on the built-in spreadsheet adapter



A	B	C	D	E	F	G	H	I	J	K	L	M
RDL namespace	Unit	EquipmentID	Equipment URI	Name	Height, total	Height	Label	Description	Parent object	Parent URI	Date	Creato
http://data.example.org/rdl/	=PR001-PU023=AEQ	=PR001-PU023-VES028	http://data.example.org/project/id=PR001-PU023-VES028	VES028	2000 mm	2000	VES028	Industrial Water	@1PE|PO|EQ|03|VES|VES01	http://data.example.org/rdl/1PEPOEQ03VESVES01	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES027	http://data.example.org/project/id=PR001-PU023-VES027	VES027			VES027	Vessel, horizontal	@1PE|PO|EQ|03|VES|VES02	http://data.example.org/rdl/1PEPOEQ03VESVES02	27-Feb-14	vvagr
Project namespace	=PR001-PU023=AEQ	=PR001-PU023-VES026	http://data.example.org/project/id=PR001-PU023-VES026	VES026			VES026	Flocculation Chamber	@1PE|PO|EQ|03|VES|VES01	http://data.example.org/rdl/1PEPOEQ03VESVES01	27-Feb-14	vvagr
http://data.example.org/project/	=PR001-PU023=AEQ	=PR001-PU023-VES025	http://data.example.org/project/id=PR001-PU023-VES025	VES025	2000 mm	2000	VES025	Hypochlorite Tank	@1PE|PO|EQ|03|VES|VES01	http://data.example.org/rdl/1PEPOEQ03VESVES01	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES024	http://data.example.org/project/id=PR001-PU023-VES024	VES024			VES024	CO2 Tank	@1PE|PO|EQ|03|VES|VES01	http://data.example.org/rdl/1PEPOEQ03VESVES01	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES023	http://data.example.org/project/id=PR001-PU023-VES023	VES023			VES023	Lime Mixing Tank	@1PE|PO|EQ|03|VES|VES01	http://data.example.org/rdl/1PEPOEQ03VESVES01	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES022	http://data.example.org/project/id=PR001-PU023-VES022	VES022	3000 mm	3000	VES022	Lime Silo	@1PE|PO|EQ|03|VES|VES01	http://data.example.org/rdl/1PEPOEQ03VESVES01	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES021	http://data.example.org/project/id=PR001-PU023-VES021	VES021	3500 mm	3500	VES021	Drinking Water Tank	@1PE|PO|EQ|03|VES|VES04	http://data.example.org/rdl/1PEPOEQ03VESVES04	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES020	http://data.example.org/project/id=PR001-PU023-VES020	VES020	3900 mm	3900	VES020	Desalinated Water Tar	@1PE|PO|EQ|03|VES|VES04	http://data.example.org/rdl/1PEPOEQ03VESVES04	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES019	http://data.example.org/project/id=PR001-PU023-VES019	VES019	1700 mm	1700	VES019	Neutralization Tank	@1PE|PO|EQ|03|VES|VES04	http://data.example.org/rdl/1PEPOEQ03VESVES04	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES018	http://data.example.org/project/id=PR001-PU023-VES018	VES018	1700 mm	1700	VES018	Reactive Tank	@1PE|PO|EQ|03|VES|VES01	http://data.example.org/rdl/1PEPOEQ03VESVES01	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES016	http://data.example.org/project/id=PR001-PU023-VES016	VES016			VES016	Dispersant	@1PE|PO|EQ|03|VES|VES03	http://data.example.org/rdl/1PEPOEQ03VESVES03	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES015	http://data.example.org/project/id=PR001-PU023-VES015	VES015	1700 mm	1700	VES015	Sulfuric Acid Tank	@1PE|PO|EQ|03|VES|VES01	http://data.example.org/rdl/1PEPOEQ03VESVES01	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES013	http://data.example.org/project/id=PR001-PU023-VES013	VES013			VES013	Caustic Soda	@1PE|PO|EQ|03|VES|VES01	http://data.example.org/rdl/1PEPOEQ03VESVES01	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES012	http://data.example.org/project/id=PR001-PU023-VES012	VES012	1700 mm	1700	VES012	Tank, vertical	@1PE|PO|EQ|03|VES|VES03	http://data.example.org/rdl/1PEPOEQ03VESVES03	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES011	http://data.example.org/project/id=PR001-PU023-VES011	VES011			VES011	Dilution Tank	@1PE|PO|EQ|03|VES|VES02	http://data.example.org/rdl/1PEPOEQ03VESVES02	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES010	http://data.example.org/project/id=PR001-PU023-VES010	VES010	1500 mm	1500	VES010	Dilution Water Tank	@1PE|PO|EQ|03|VES|VES02	http://data.example.org/rdl/1PEPOEQ03VESVES02	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES009	http://data.example.org/project/id=PR001-PU023-VES009	VES009			VES009	Collection Sump	@1PE|PO|EQ|03|VES|VES02	http://data.example.org/rdl/1PEPOEQ03VESVES02	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES008	http://data.example.org/project/id=PR001-PU023-VES008	VES008	1500 mm	1500	VES008	Wash Water Tank	@1PE|PO|EQ|03|VES|VES04	http://data.example.org/rdl/1PEPOEQ03VESVES04	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES007	http://data.example.org/project/id=PR001-PU023-VES007	VES007			VES007	Vessel, vertical	@1PE|PO|EQ|03|VES|VES01	http://data.example.org/rdl/1PEPOEQ03VESVES01	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES006	http://data.example.org/project/id=PR001-PU023-VES006	VES006	2900 mm	2900	VES006	Tank, vertical	@1PE|PO|EQ|03|VES|VES03	http://data.example.org/rdl/1PEPOEQ03VESVES03	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES005	http://data.example.org/project/id=PR001-PU023-VES005	VES005			VES005	Ferric Chloride Tank	@1PE|PO|EQ|03|VES|VES03	http://data.example.org/rdl/1PEPOEQ03VESVES03	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES004	http://data.example.org/project/id=PR001-PU023-VES004	VES004	3000 mm	3000	VES004	Sea Water Tank	@1PE|PO|EQ|03|VES|VES04	http://data.example.org/rdl/1PEPOEQ03VESVES04	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES003	http://data.example.org/project/id=PR001-PU023-VES003	VES003			VES003	Beach Wells	@1PE|PO|EQ|03|VES|VES03	http://data.example.org/rdl/1PEPOEQ03VESVES03	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VES001	http://data.example.org/project/id=PR001-PU023-VES001	VES001	1200 mm	1200	VES001	Hypochlorite Tank	@1PE|PO|EQ|03|VES|VES03	http://data.example.org/rdl/1PEPOEQ03VESVES03	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VAL088	http://data.example.org/project/id=PR001-PU023-VAL088	VAL088			VAL088	Armature	@1PE|PO|EQ|06|VAL	http://data.example.org/rdl/1PEPOEQ06VAL	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VAL087	http://data.example.org/project/id=PR001-PU023-VAL087	VAL087			VAL087	Armature	@1PE|PO|EQ|06|VAL	http://data.example.org/rdl/1PEPOEQ06VAL	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VAL086	http://data.example.org/project/id=PR001-PU023-VAL086	VAL086			VAL086	Armature	@1PE|PO|EQ|06|VAL	http://data.example.org/rdl/1PEPOEQ06VAL	27-Feb-14	vvagr
	=PR001-PU023=AEQ	=PR001-PU023-VAL085	http://data.example.org/project/id=PR001-PU023-VAL085	VAL085			VAL085	Armature	@1PE|PO|EQ|06|VAL	http://data.example.org/rdl/1PEPOEQ06VAL	27-Feb-14	vvagr

Column A contains project RDL and project data namespaces to form URIs for entities.

We will construct Equipment IDs in column C using Unit ID (column B) and equipment label (column H) using Excel formula:

=CONCATENATE(LEFT(B2;12);"-";H2)

The same schema is used for equipment IDs in the second project export file.

Equipment IDs we will use to form equipment URIs in column D by concatenating them with project namespace and prefix *"id"* using formula:

=CONCATENATE(A5; "id"; C2)

To classify project data items we'll reconstruct project RDL URIs in column K from parent object IDs in column J. The schema used to build these URIs is the one used in project RDL import:

=SUBSTITUTE(SUBSTITUTE(CONCATENATE(A$2;J2);"|";"");"@";"")

To import the Height attribute we have to separate value from the UOM. It is done by the Excel formula:

=IF(ISBLANK(F2); ""; (LEFT(F2;FIND("mm";F2)-4)))

This formula accounts for the fact that not all items have an attribute recorded, and we have lo leave blank cells blank.

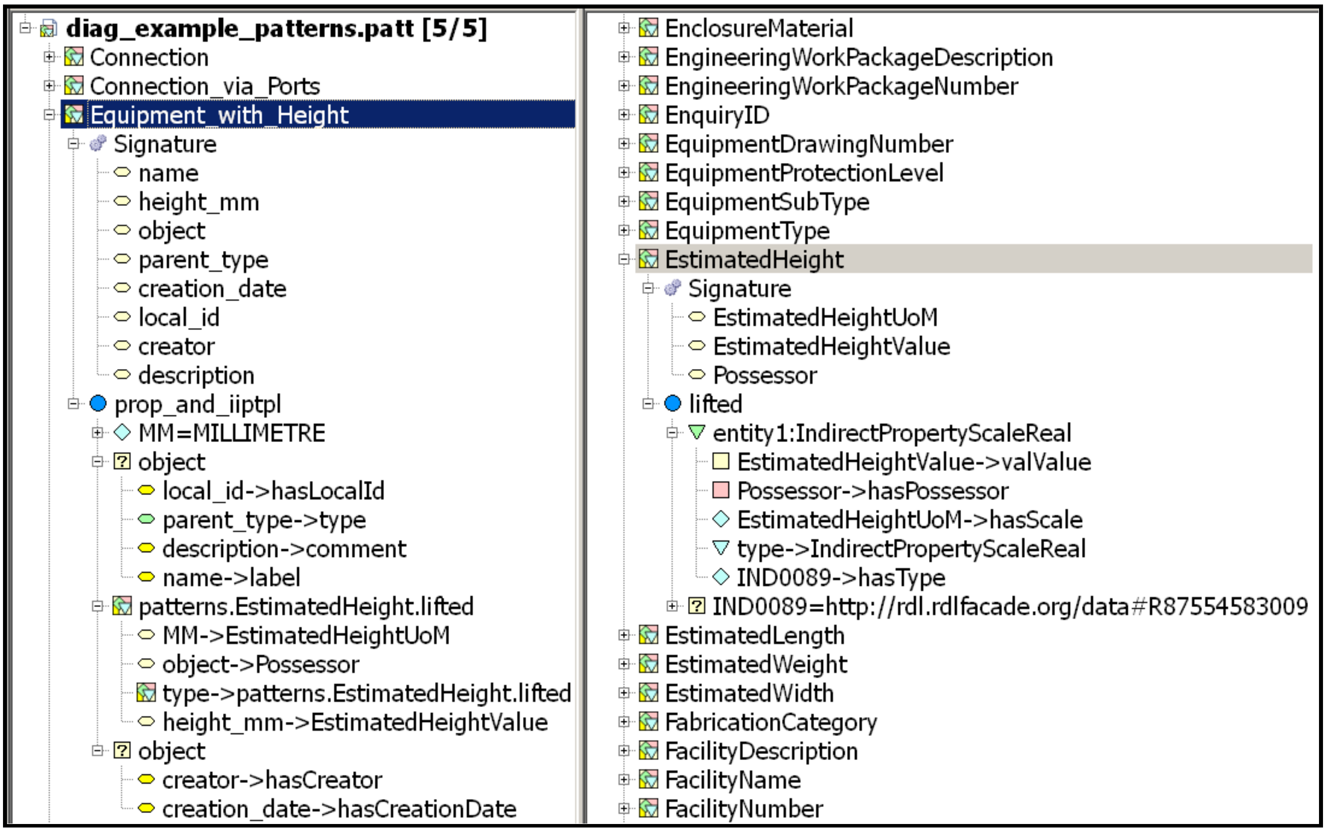
As all UOMs are the same (millimetres) we will not put them in a separate column, just record them in the mapping.

Columns L and M contain the same metadata we've used in the project RDL.

7. Defining pattern for Equipment.

To import equipment spreadsheet content into the project data source we need a pattern which will describe the structure of the spreadsheet. Open the panel with the **diag_example_patterns.patt** again (project-specific pattern data source), find *Equipment_with_Height* pattern and fully unfold all its nodes.

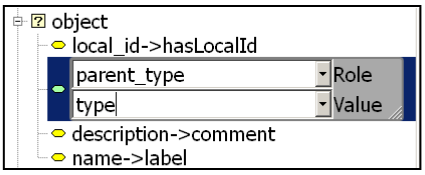
The pattern has a signature that corresponds to the columns of the imported spreadsheet. One single mapping to templates and properties is defined for this pattern. It is named *prop_and_iiptpl* to reflect the fact that it contains mapping to properties and to the templates from IIP template set.



```
diag_example_patterns.patt [5/5]
  Connection
  Connection_via_Ports
  Equipment_with_Height
    Signature
      name
      height_mm
      object
      parent_type
      creation_date
      local_id
      creator
      description
    prop_and_iiptpl
      MM=MILLIMETRE
      object
        local_id->hasLocalId
        parent_type->type
        description->comment
        name->label
      patterns.EstimatedHeight.lifted
        MM->EstimatedHeightUoM
        object->Possessor
        type->patterns.EstimatedHeight.lifted
        height_mm->EstimatedHeightValue
      object
        creator->hasCreator
        creation_date->hasCreationDate
```

```
EnclosureMaterial
EngineeringWorkPackageDescription
EngineeringWorkPackageNumber
EnquiryID
EquipmentDrawingNumber
EquipmentProtectionLevel
EquipmentSubType
EquipmentType
EstimatedHeight
  Signature
    EstimatedHeightUoM
    EstimatedHeightValue
    Possessor
  lifted
    entity1:IndirectPropertyScaleReal
      EstimatedHeightValue->valValue
      Possessor->hasPossessor
      EstimatedHeightUoM->hasScale
      type->IndirectPropertyScaleReal
      IND0089->hasType
    IND0089=http://rdl.rdlfacade.org/data#R87554583009
EstimatedLength
EstimatedWeight
EstimatedWidth
FabricationCategory
FacilityDescription
FacilityName
FacilityNumber
```

It maps three roles *local_id, comment* and *name* to the appropriate annotation properties of an *object* and also maps a *parent_type* role to the *rdf:type* property (this role of the pattern should be occupied by an URI of parent object).



```
object
  local_id->hasLocalId
  parent_type   Role
  type          Value
  description->comment
  name->label
```

We’ll not assign Part 2 types to the entities in the project data, using their classifiers from project RDL instead. It is difficult to add Part 2 types to the big project data spreadsheet manually, but very easy to do it after the import by inferring appropriate types from RDL classifiers, if required.

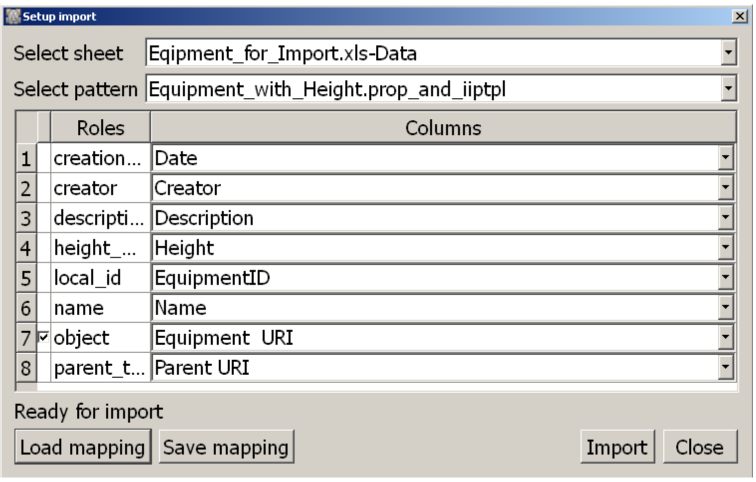
To map the Height property we'll use *EstimatedHeight* TIP from imported database of the TIP Manager. To do it we create a part *MM* which contains millimetre UOM (all RD entities to be referred in patterns require separate parts in the pattern description).

The next part corresponds to the *EstimatedHeight* TIP with *MM* part occupying the *EstimatedHeightUoM* role, *oblect* mapped to the *Possessor* role and *height* value mapped to the *EstimatedHeightValue* role.

The pattern also contains the separate part assigning to the same *object* our metadata properties.

8. Importing Equipment

Check that **Equipment_for_Import.xls** file is open on your computer and **ExampleData** panel is an active panel in the Editor. Call pattern import extension (*Build patterns from MS Excel* in *Extensions* menu).



Setup import

Select sheet: Eqipment_for_Import.xls-Data

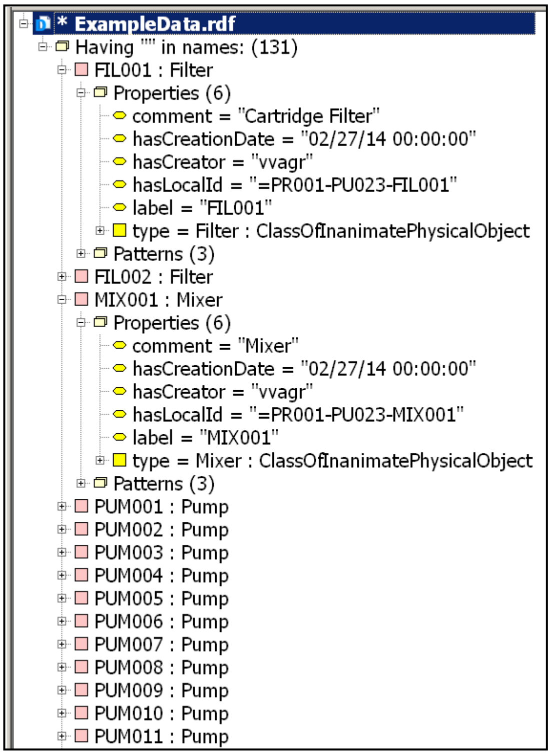
Select pattern: Equipment_with_Height.prop_and_iiptpl

	Roles	Columns
1	creation...	Date
2	creator	Creator
3	descripti...	Description
4	height_...	Height
5	local_id	EquipmentID
6	name	Name
7	☑ object	Equipment URI
8	parent_t...	Parent URI

Ready for import

Select sheet *Data* and load mapping **equipment_height.json** from **\Scripts** folder. Correspondence between pattern roles described above and spreadsheet columns is established. The check mark at the *object* role indicates that entity in this role should be created with URI recorded in the corresponding column (all other URIs for new entities will be generated by the Editor).

Import data and check the content of the **ExampleData**.

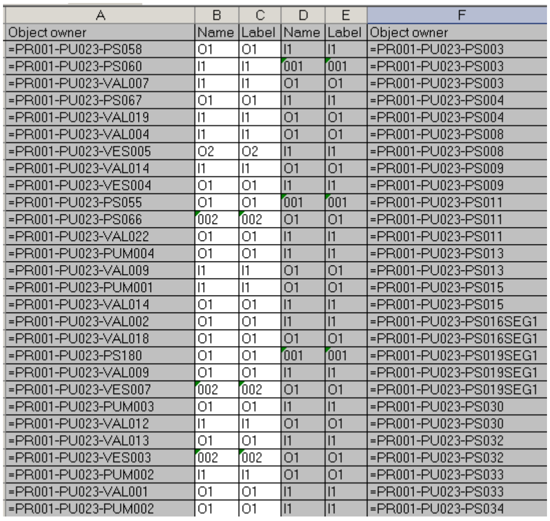


```
* ExampleData.rdf
  Having "" in names: (131)
    FIL001 : Filter
      Properties (6)
        comment = "Cartridge Filter"
        hasCreationDate = "02/27/14 00:00:00"
        hasCreator = "vvagr"
        hasLocalId = "=PR001-PU023-FIL001"
        label = "FIL001"
        type = Filter : ClassOfInanimatePhysicalObject
      Patterns (3)
    FIL002 : Filter
    MIX001 : Mixer
      Properties (6)
        comment = "Mixer"
        hasCreationDate = "02/27/14 00:00:00"
        hasCreator = "vvagr"
        hasLocalId = "=PR001-PU023-MIX001"
        label = "MIX001"
        type = Mixer : ClassOfInanimatePhysicalObject
      Patterns (3)
    PUM001 : Pump
    PUM002 : Pump
    PUM003 : Pump
    PUM004 : Pump
    PUM005 : Pump
    PUM006 : Pump
    PUM007 : Pump
    PUM008 : Pump
    PUM009 : Pump
    PUM010 : Pump
    PUM011 : Pump
```

9. Preparing Connectivity data

Now we will prepare for import a second exported project data file **ProcessUnits-Connectors.xls**.

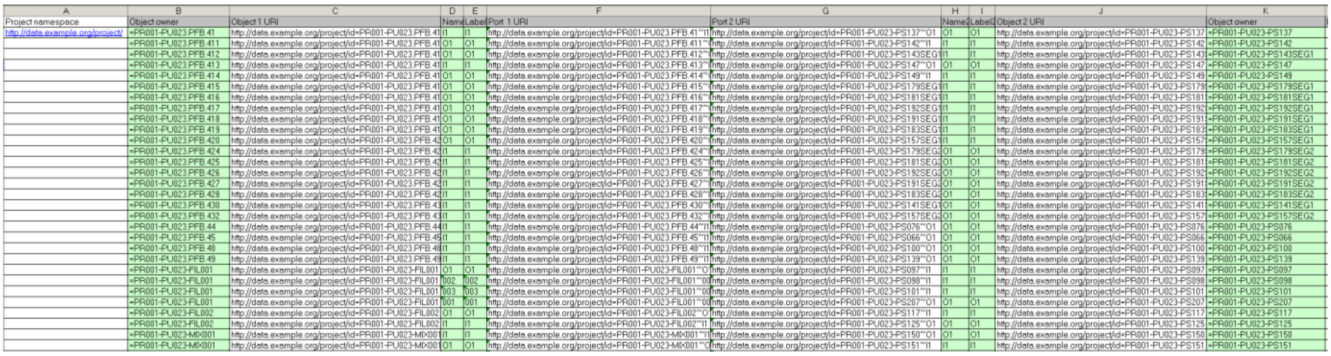
Looking at the file we can notice several important points:



A	B	C	D	E	F
Object owner	Name	Label	Name	Label	Object owner
=PR001-PU023-PS058	O1	O1	I1	I1	=PR001-PU023-PS003
=PR001-PU023-PS060	I1	I1	001	001	=PR001-PU023-PS003
=PR001-PU023-VAL007	I1	I1	O1	O1	=PR001-PU023-PS003
=PR001-PU023-PS067	O1	O1	I1	I1	=PR001-PU023-PS004
=PR001-PU023-VAL019	I1	I1	O1	O1	=PR001-PU023-PS004
=PR001-PU023-VAL004	I1	I1	O1	O1	=PR001-PU023-PS008
=PR001-PU023-VES005	O2	O2	I1	I1	=PR001-PU023-PS008
=PR001-PU023-VAL014	I1	I1	O1	O1	=PR001-PU023-PS009
=PR001-PU023-VES004	O1	O1	I1	I1	=PR001-PU023-PS009
=PR001-PU023-PS055	O1	O1	001	001	=PR001-PU023-PS011
=PR001-PU023-PS066	002	002	O1	O1	=PR001-PU023-PS011
=PR001-PU023-VAL022	O1	O1	I1	I1	=PR001-PU023-PS011
=PR001-PU023-PUM004	O1	O1	I1	I1	=PR001-PU023-PS013
=PR001-PU023-VAL009	I1	I1	O1	O1	=PR001-PU023-PS013
=PR001-PU023-PUM001	I1	I1	O1	O1	=PR001-PU023-PS015
=PR001-PU023-VAL014	O1	O1	I1	I1	=PR001-PU023-PS015
=PR001-PU023-VAL002	O1	O1	I1	I1	=PR001-PU023-PS016SEG1
=PR001-PU023-VAL018	O1	O1	O1	O1	=PR001-PU023-PS016SEG1
=PR001-PU023-PS180	O1	O1	001	001	=PR001-PU023-PS019SEG1
=PR001-PU023-VAL009	O1	O1	I1	I1	=PR001-PU023-PS019SEG1
=PR001-PU023-VES007	002	002	O1	O1	=PR001-PU023-PS019SEG1
=PR001-PU023-PUM003	O1	O1	I1	I1	=PR001-PU023-PS030
=PR001-PU023-VAL012	I1	I1	O1	O1	=PR001-PU023-PS030
=PR001-PU023-VAL013	O1	O1	I1	I1	=PR001-PU023-PS032
=PR001-PU023-VES003	002	002	O1	O1	=PR001-PU023-PS032
=PR001-PU023-PUM002	I1	I1	O1	O1	=PR001-PU023-PS033
=PR001-PU023-VAL001	O1	O1	I1	I1	=PR001-PU023-PS033
=PR001-PU023-PUM002	O1	O1	I1	I1	=PR001-PU023-PS034

1. Equipment items are identified by internal IDs we've already learned to reconstruct during equipment import.
2. There are more objects, identified as PS and PFB, representing process streams connecting equipment items on the diagram or leading to other equipment beyond this diagram. Two more entities should be added to the project RDL to classify such objects in the project data (they were already added in our file).
3. Connection of objects is recorded via ports. Some ports have identifiers with letters *O* or *I*, signifying that they are either *output* or *input* ports. We should create such ports in the project data as separate entities then.
4. Each connection is recorded twice – from A to B and from B to A. It is very difficult to clean this out is Excel, so we'll deal with it later.

Let's prepare these data for import. Go to the folder **\ForImport** and open **Connections_for_Import.xls** file.



A	B	C	D	E	F	G	H	I	J	K
Project namespace	Object owner	Object 1 URI	Name	Label	Port 1 URI	Port 2 URI	Name2	Label2	Object 2 URI	Object owner
http://data.example.org/project/	=PR001-PU023.PFB.41	http://data.example.org/project/id=PR001-PU023.PFB.41	I1	I1	http://data.example.org/project/id=PR001-PU023.PFB.41~I1	http://data.example.org/project/id=PR001-PU023-PS137~O1	O1	O1	http://data.example.org/project/id=PR001-PU023-PS137	=PR001-PU023-PS137
	=PR001-PU023.PFB.411	http://data.example.org/project/id=PR001-PU023.PFB.41	O1	O1	http://data.example.org/project/id=PR001-PU023.PFB.411~	http://data.example.org/project/id=PR001-PU023-PS142~I1	I1	I1	http://data.example.org/project/id=PR001-PU023-PS142	=PR001-PU023-PS142
	=PR001-PU023.PFB.412	http://data.example.org/project/id=PR001-PU023.PFB.41	O1	O1	http://data.example.org/project/id=PR001-PU023.PFB.412~	http://data.example.org/project/id=PR001-PU023-PS143SEG1	I1	I1	http://data.example.org/project/id=PR001-PU023-PS143	=PR001-PU023-PS143SEG1
	=PR001-PU023.PFB.413	http://data.example.org/project/id=PR001-PU023.PFB.41	I1	I1	http://data.example.org/project/id=PR001-PU023.PFB.413~	http://data.example.org/project/id=PR001-PU023-PS147~O1	O1	O1	http://data.example.org/project/id=PR001-PU023-PS147	=PR001-PU023-PS147
	=PR001-PU023.PFB.414	http://data.example.org/project/id=PR001-PU023.PFB.41	O1	O1	http://data.example.org/project/id=PR001-PU023.PFB.414~	http://data.example.org/project/id=PR001-PU023-PS149~I1	I1	I1	http://data.example.org/project/id=PR001-PU023-PS149	=PR001-PU023-PS149
	=PR001-PU023.PFB.415	http://data.example.org/project/id=PR001-PU023.PFB.41	O1	O1	http://data.example.org/project/id=PR001-PU023.PFB.415~	http://data.example.org/project/id=PR001-PU023-PS179SEG1	I1	I1	http://data.example.org/project/id=PR001-PU023-PS179	=PR001-PU023-PS179SEG1
	=PR001-PU023.PFB.416	http://data.example.org/project/id=PR001-PU023.PFB.41	O1	O1	http://data.example.org/project/id=PR001-PU023.PFB.416~	http://data.example.org/project/id=PR001-PU023-PS181SEG1	I1	I1	http://data.example.org/project/id=PR001-PU023-PS181	=PR001-PU023-PS181SEG1
	=PR001-PU023.PFB.417	http://data.example.org/project/id=PR001-PU023.PFB.41	O1	O1	http://data.example.org/project/id=PR001-PU023.PFB.417~	http://data.example.org/project/id=PR001-PU023-PS192SEG1	I1	I1	http://data.example.org/project/id=PR001-PU023-PS192	=PR001-PU023-PS192SEG1
	=PR001-PU023.PFB.418	http://data.example.org/project/id=PR001-PU023.PFB.41	O1	O1	http://data.example.org/project/id=PR001-PU023.PFB.418~	http://data.example.org/project/id=PR001-PU023-PS191SEG1	I1	I1	http://data.example.org/project/id=PR001-PU023-PS191	=PR001-PU023-PS191SEG1
	=PR001-PU023.PFB.419	http://data.example.org/project/id=PR001-PU023.PFB.41	O1	O1	http://data.example.org/project/id=PR001-PU023.PFB.419~	http://data.example.org/project/id=PR001-PU023-PS183SEG1	I1	I1	http://data.example.org/project/id=PR001-PU023-PS183	=PR001-PU023-PS183SEG1
	=PR001-PU023.PFB.420	http://data.example.org/project/id=PR001-PU023.PFB.42	O1	O1	http://data.example.org/project/id=PR001-PU023.PFB.420~	http://data.example.org/project/id=PR001-PU023-PS157SEG1	I1	I1	http://data.example.org/project/id=PR001-PU023-PS157	=PR001-PU023-PS157SEG1
	=PR001-PU023.PFB.424	http://data.example.org/project/id=PR001-PU023.PFB.42	I1	I1	http://data.example.org/project/id=PR001-PU023.PFB.424~	http://data.example.org/project/id=PR001-PU023-PS179SEG2	O1	O1	http://data.example.org/project/id=PR001-PU023-PS179	=PR001-PU023-PS179SEG2
	=PR001-PU023.PFB.425	http://data.example.org/project/id=PR001-PU023.PFB.42	I1	I1	http://data.example.org/project/id=PR001-PU023.PFB.425~	http://data.example.org/project/id=PR001-PU023-PS181SEG2	O1	O1	http://data.example.org/project/id=PR001-PU023-PS181	=PR001-PU023-PS181SEG2
	=PR001-PU023.PFB.426	http://data.example.org/project/id=PR001-PU023.PFB.42	I1	I1	http://data.example.org/project/id=PR001-PU023.PFB.426~	http://data.example.org/project/id=PR001-PU023-PS192SEG2	O1	O1	http://data.example.org/project/id=PR001-PU023-PS192	=PR001-PU023-PS192SEG2
	=PR001-PU023.PFB.427	http://data.example.org/project/id=PR001-PU023.PFB.42	I1	I1	http://data.example.org/project/id=PR001-PU023.PFB.427~	http://data.example.org/project/id=PR001-PU023-PS191SEG2	O1	O1	http://data.example.org/project/id=PR001-PU023-PS191	=PR001-PU023-PS191SEG2
	=PR001-PU023.PFB.428	http://data.example.org/project/id=PR001-PU023.PFB.42	I1	I1	http://data.example.org/project/id=PR001-PU023.PFB.428~	http://data.example.org/project/id=PR001-PU023-PS183SEG2	O1	O1	http://data.example.org/project/id=PR001-PU023-PS183	=PR001-PU023-PS183SEG2
	=PR001-PU023.PFB.430	http://data.example.org/project/id=PR001-PU023.PFB.43	I1	I1	http://data.example.org/project/id=PR001-PU023.PFB.430~	http://data.example.org/project/id=PR001-PU023-PS141SEG1	O1	O1	http://data.example.org/project/id=PR001-PU023-PS141	=PR001-PU023-PS141SEG1
	=PR001-PU023.PFB.432	http://data.example.org/project/id=PR001-PU023.PFB.43	I1	I1	http://data.example.org/project/id=PR001-PU023.PFB.432~	http://data.example.org/project/id=PR001-PU023-PS157SEG2	O1	O1	http://data.example.org/project/id=PR001-PU023-PS157	=PR001-PU023-PS157SEG2
	=PR001-PU023.PFB.44	http://data.example.org/project/id=PR001-PU023.PFB.44	I1	I1	http://data.example.org/project/id=PR001-PU023.PFB.44~I1	http://data.example.org/project/id=PR001-PU023-PS076~O1	O1	O1	http://data.example.org/project/id=PR001-PU023-PS076	=PR001-PU023-PS076
	=PR001-PU023.PFB.45	http://data.example.org/project/id=PR001-PU023.PFB.45	I1	I1	http://data.example.org/project/id=PR001-PU023.PFB.45~I1	http://data.example.org/project/id=PR001-PU023-PS066~O1	O1	O1	http://data.example.org/project/id=PR001-PU023-PS066	=PR001-PU023-PS066
	=PR001-PU023.PFB.48	http://data.example.org/project/id=PR001-PU023.PFB.48	I1	I1	http://data.example.org/project/id=PR001-PU023.PFB.48~I1	http://data.example.org/project/id=PR001-PU023-PS100~O1	O1	O1	http://data.example.org/project/id=PR001-PU023-PS100	=PR001-PU023-PS100
	=PR001-PU023.PFB.49	http://data.example.org/project/id=PR001-PU023.PFB.49	I1	I1	http://data.example.org/project/id=PR001-PU023.PFB.49~I1	http://data.example.org/project/id=PR001-PU023-PS139~O1	O1	O1	http://data.example.org/project/id=PR001-PU023-PS139	=PR001-PU023-PS139
	=PR001-PU023-FIL001	http://data.example.org/project/id=PR001-PU023-FIL001	O1	O1	http://data.example.org/project/id=PR001-PU023-FIL001~O	http://data.example.org/project/id=PR001-PU023-PS097~I1	I1	I1	http://data.example.org/project/id=PR001-PU023-PS097	=PR001-PU023-PS097
	=PR001-PU023-FIL001	http://data.example.org/project/id=PR001-PU023-FIL001	002	002	http://data.example.org/project/id=PR001-PU023-FIL001~00	http://data.example.org/project/id=PR001-PU023-PS098~I1	I1	I1	http://data.example.org/project/id=PR001-PU023-PS098	=PR001-PU023-PS098
	=PR001-PU023-FIL001	http://data.example.org/project/id=PR001-PU023-FIL001	003	003	http://data.example.org/project/id=PR001-PU023-FIL001~00	http://data.example.org/project/id=PR001-PU023-PS101~I1	I1	I1	http://data.example.org/project/id=PR001-PU023-PS101	=PR001-PU023-PS101
	=PR001-PU023-FIL001	http://data.example.org/project/id=PR001-PU023-FIL001	001	001	http://data.example.org/project/id=PR001-PU023-FIL001~00	http://data.example.org/project/id=PR001-PU023-PS207~O1	O1	O1	http://data.example.org/project/id=PR001-PU023-PS207	=PR001-PU023-PS207
	=PR001-PU023-FIL002	http://data.example.org/project/id=PR001-PU023-FIL002	O1	O1	http://data.example.org/project/id=PR001-PU023-FIL002~O	http://data.example.org/project/id=PR001-PU023-PS117~I1	I1	I1	http://data.example.org/project/id=PR001-PU023-PS117	=PR001-PU023-PS117
	=PR001-PU023-FIL002	http://data.example.org/project/id=PR001-PU023-FIL002	I1	I1	http://data.example.org/project/id=PR001-PU023-FIL002~I1	http://data.example.org/project/id=PR001-PU023-PS125~O1	O1	O1	http://data.example.org/project/id=PR001-PU023-PS125	=PR001-PU023-PS125
	=PR001-PU023-MIX001	http://data.example.org/project/id=PR001-PU023-MIX001	I1	I1	http://data.example.org/project/id=PR001-PU023-MIX001~I1	http://data.example.org/project/id=PR001-PU023-PS150~O1	O1	O1	http://data.example.org/project/id=PR001-PU023-PS150	=PR001-PU023-PS150
	=PR001-PU023-MIX001	http://data.example.org/project/id=PR001-PU023-MIX001	O1	O1	http://data.example.org/project/id=PR001-PU023-MIX001~O	http://data.example.org/project/id=PR001-PU023-PS151~I1	I1	I1	http://data.example.org/project/id=PR001-PU023-PS151	=PR001-PU023-PS151

Column A contains project data namespace to form URIs for entities

We"ll use IDs for the connected objects (columns B and K) to form object URIs in columns C and J, using the same concatenation Excel formula we've used before.

We also need some schema to construct URIs for new port objects. To do it we'll concatenate object (port owner) URIs with "~" symbol and with port Label from column E or I and write port URIs to columns F and G:

=CONCATENATE(C2;"~";E2)

=CONCATENATE(J2;"~";I2)

UUID generation can be used at a later stage when the adapter is tested and implemented as a separate code.

Columns L and M contain the same metadata as before.

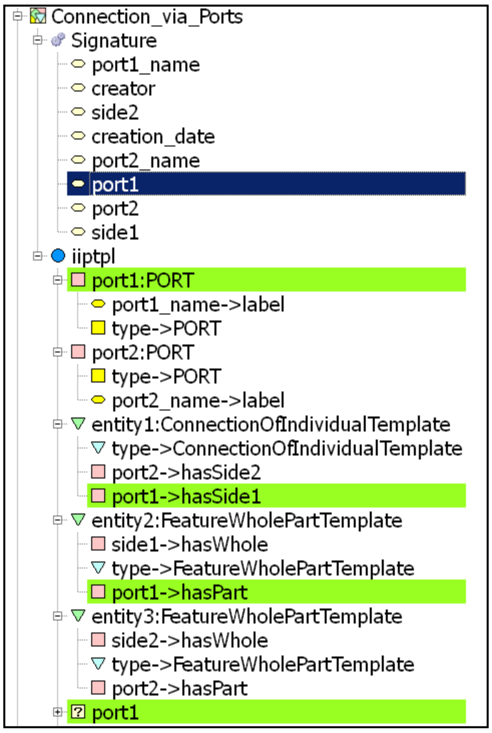
10. Defining pattern for Connectivity

To import connectivity spreadsheet content into the project data source we need a pattern which will describe the structure of the spreadsheet. Open the panel with the **diag_example_patterns.patt** again (project-specific pattern data source), find *Connection_via_Ports* pattern and fully unfold all its nodes.

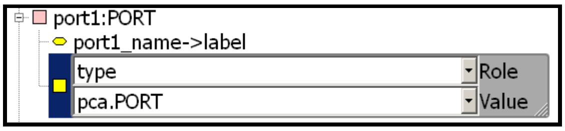
Click on pattern nodes to see the way template parts are connected together.

The pattern has a signature that corresponds to the columns of the imported spreadsheet. One single mapping to templates and properties is defined for this pattern. It is named *iiptpl* to reflect the fact that it contains mapping to the templates from IIP template set.

Two objects of the PORT type are described in the pattern; they'll be created in the data source for each recorded connection. PCA ClassOfFunctionalObjectis PORT is used as type of port objects.



```
Connection_via_Ports
  Signature
    port1_name
    creator
    side2
    creation_date
    port2_name
    port1
    port2
    side1
  iiptpl
    port1:PORT
      port1_name->label
      type->PORT
    port2:PORT
      type->PORT
      port2_name->label
    entity1:ConnectionOfIndividualTemplate
      type->ConnectionOfIndividualTemplate
      port2->hasSide2
      port1->hasSide1
    entity2:FeatureWholePartTemplate
      side1->hasWhole
      type->FeatureWholePartTemplate
      port1->hasPart
    entity3:FeatureWholePartTemplate
      side2->hasWhole
      type->FeatureWholePartTemplate
      port2->hasPart
    port1
```



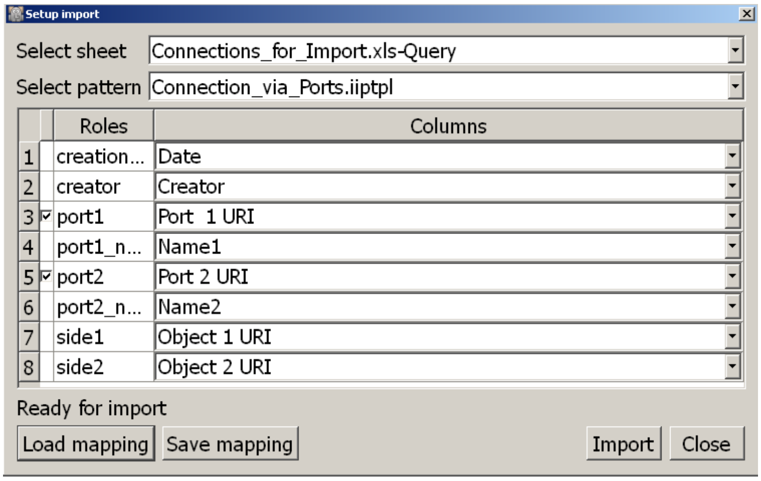
```
port1:PORT
  port1_name->label
  type       Role
  pca.PORT   Value
```

Three templates are included in the pattern, two are describing ports as features of the corresponding project objects, and one is describing connection between ports.

The pattern again contains the separate part for each previously described element with metadata properties assigned to facilitate a repeated import.

11. Importing Connectivity

Check that **Connections_for_Import.xls** file is open on your computer and **ExampleData** panel is an active panel in the Editor. Call pattern import extension (*Build patterns from MS Excel* in *Extensions* menu).



Setup import

Select sheet: Connections_for_Import.xls-Query

Select pattern: Connection_via_Ports.iiptpl

		Roles	Columns
1		creation...	Date
2		creator	Creator
3	☑	port1	Port 1 URI
4		port1_n...	Name1
5	☑	port2	Port 2 URI
6		port2_n...	Name2
7		side1	Object 1 URI
8		side2	Object 2 URI

Ready for import

Select sheet *Query* and load mapping **connections.json** from **\Scripts** folder. Correspondence between pattern roles described above and spreadsheet columns is established. Two check marks at the *port* roles indicate that entities in these role should be created with URIs recorded in the corresponding columns (all other URIs for new entities will be generated by the Editor).

12. Removing duplicates and typing missing objects

Now we will solve in the Editor some problems too difficult to solve during data preprocessing in Excel.

a. Port type assignment

Objects with the type PORT should receive classifications with PCA RDL classes INPUT and OUTPUT dependant on the letter used in their names.

b. Stream import completion

Stream and connector objects (which are now occupying roles in the connectivity templates) should be properly declared with label and local ID properties, and classified with project RDL entities based on their IDs (Process Stream or PFB connectors).

c. Duplication removal

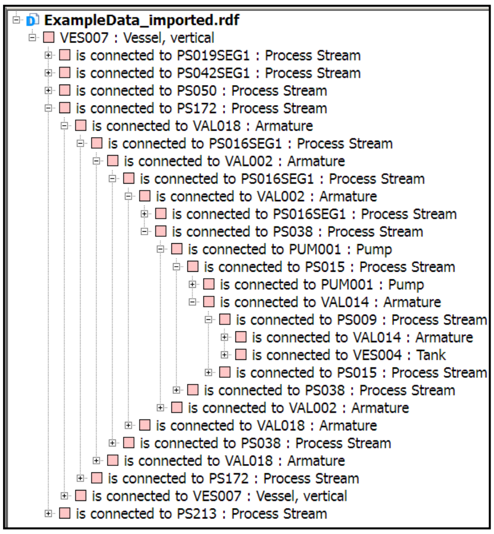
We have to remove duplicate instances of the ConnectionOfIndividualTemplate, where ports A and B occupy hasSide1 and hasSide2 roles once in the direct order and once in the opposite order. Notice that no duplicates were created for ports themselves or for FeatureWholePartTemplate, although each was also processed twice during the import – adapter will never duplicate fully identical objects.

To solve these problems find **project_scripts.py** file in the **\Scripts** folder and run its content in the Editor's console. The execution of the script can take some time, the message *Done* will be printed in the console window upon completion.

An import is finished. Now we have representation of the exported diagram data in the ISO 15926 RDF file.

13. Exporting and comparing diagram

It is not an easy task to check import correctness by navigation through an RDF file, although pattern view (simplified entity view) in the Editor to some degree allows verification of data.



```
ExampleData_imported.rdf
  VES007 : Vessel, vertical
    is connected to PS019SEG1 : Process Stream
    is connected to PS042SEG1 : Process Stream
    is connected to PS050 : Process Stream
    is connected to PS172 : Process Stream
      is connected to VAL018 : Armature
        is connected to PS016SEG1 : Process Stream
          is connected to VAL002 : Armature
            is connected to PS016SEG1 : Process Stream
              is connected to VAL002 : Armature
                is connected to PS016SEG1 : Process Stream
                is connected to PS038 : Process Stream
                  is connected to PUM001 : Pump
                    is connected to PS015 : Process Stream
                      is connected to PUM001 : Pump
                      is connected to VAL014 : Armature
                        is connected to PS009 : Process Stream
                          is connected to VAL014 : Armature
                          is connected to VES004 : Tank
                        is connected to PS015 : Process Stream
                    is connected to PS038 : Process Stream
                  is connected to VAL002 : Armature
              is connected to VAL018 : Armature
            is connected to PS038 : Process Stream
          is connected to VAL018 : Armature
        is connected to PS172 : Process Stream
      is connected to VES007 : Vessel, vertical
    is connected to PS213 : Process Stream
```

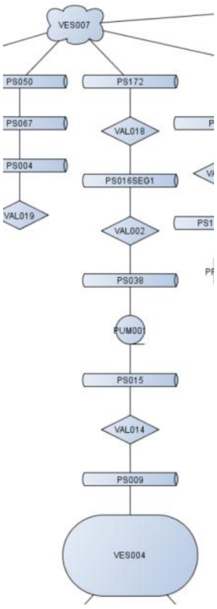
We have implemented a basic graphical viewer for ISO 15926 data and will use it to compare our results with the source diagram.

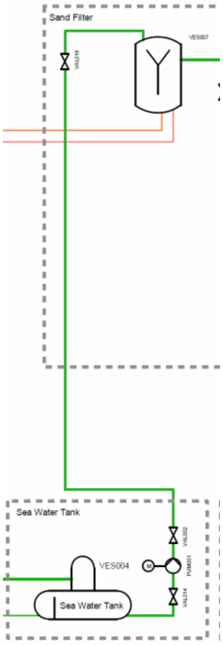
The viewer is just a Python script which generates an **.xgml** graph file. This file can be opened, automatically arranged and explored in the free **yEd** graph editor (downloadable from http://www.yworks.com/en/products_yed_about.html).

Install **yEd** on your computer, then find **viewer.py** file in the **\Scripts** folder and run its content in the Editor's console. The execution will take some time, the message *Done* will be printed in the console window upon completion.

Locate **pid_view.xgml** file in the folder with Editor executable and open it in **yEd**. Use automatic layout via *Layout* menu (good results are obtained with *Tree* layout, just check the box *Allow General Graphs* in layout options). You can also find arranged **pid_view.xgml** and exported **pid_view.png** files in the **\Imported** folder.

The viewer uses shapes predefined in the standard yEd libraries to render equipment of different types, and resulting picture is not very similar to the standard PFD or P&ID.





Nevertheless visual comparison is possible and its results are quite satisfactory.

It appears that the diagram falls into several disconnected components because exported connectivity data aren’t complete and doesn’t contain connections for some items around the Cartridge Filter module.

14. Viewing Linked Data pages for the project

Let’s explore another way to look at ISO 15926 RDF data sets. There is a growing interest in the engineering community in the Semantic Web approaches to data representation, publishing and management. Linked Data is one such approach.

Open-source Linked Data extension is developed for the .15926 Editor using open source Python web toollkits - Flask (http://flask.pocoo.org/) and Tornado (http://www.tornadoweb.org/).

Linked Data extension turns your Editor into a web server capable to deliver HTML pages based on the RDF data sets. In the basic configuration the server works locally on your computer. It can service interconnected human-readable pages for represented concepts processing diverse URIs and turning them into local page URLs. Advanced configuration possibilities allow use of the extension on the Internet with differentiated processing for server owner’s own URIs and URIs of external data.

Unlike the most other Linked Data server applications, in our Linked Data extension content of the pages is defined by patterns and is open for customisation. It is possible to model some relationship or concept as a complex RDF graph (pattern), and describe its preferred human-readable appearance on a web page by HTML template. The Editor will search for the pattern in RDF data and put its information on the page in a comprehensible form.

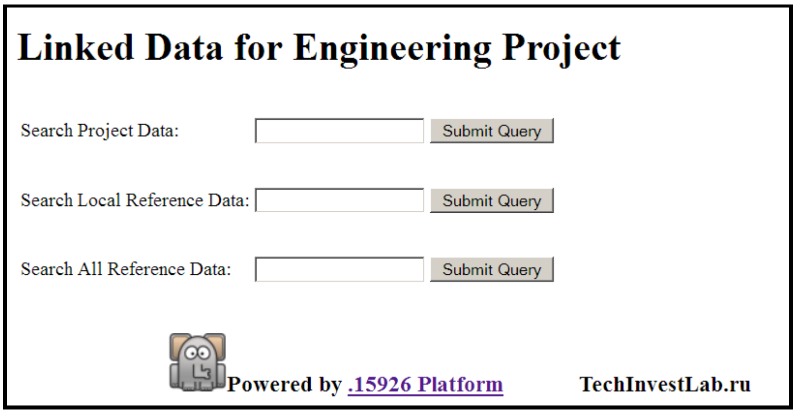
The search in the extension is also pattern-oriented and has semantic capabilities.

Searching for the string it locates this string in all identifiers and in all classifiers of data entities. For example, searching for “pump” will return all entities with “pump” in identifier and all entities classified with entities with “pump” in identifier.

Identifiers and classifiers are in turn defined by corresponding patterns. For example, objects of both *rdfs:label* and *http://data.example.org/properties/hasLocalId* can be defined as identifiers.

Classifiers are searched recursively across the data sources. An entity with *rdf:type* X will get X as direct classifier and all superclasses of X as inferred classifiers.

To see Linked Data extension working you can open the project **ProcessLinkedData.15926** from **/Imported** folder. This project contains the same data we’ve just imported from spreadsheets. Go to *Extensions* menu and select *Start/stop linked data demo* command. Point your browser to ***http://localhost:5000/***



Linked Data for Engineering Project

Search Project Data:
Submit Query
Search Local Reference Data:
Submit Query
Search All Reference Data:
Submit Query
Powered by .15926 Platform
TechInvestLab.ru

Three search fields on the screen allow you to search in the project data, in the local reference data or in all reference data (local and PCA RDL).

Search for “valve” in *Project Data* field. Although there are no entities with “valve” substring in the label, there are many entities classified with *Armature* class, which is in turn subclass of the *VALVE* class in PCA RDL. Semantic search of the extension will return all project entities for which *VALVE* is a direct or inferred classifier.

If you search for “artefact” in *Project Data* field – you will get all equipment items (valves, pumps, vessels, etc.). PCA *ARTEFACT* is an inferred superclass for all equipment types.

And if you search for “thing” in *Project Data* – all entities in the project will be returned, as PCA *ISO 15926-4 THING* is inferred classifier for all project entities.

Navigate to a particular equipment item page, for example to http://localhost:5000/entity?uri=http://data.example.org/project/id=PR001-PU023-VAL071. You can see information on entity Identifiers, Definitions and Descriptions, Direct classifiers, Inferred classifiers. IIP template pattern allows identifying parts of an entity (ports of a valve in this case). And connection via ports described in the patterns allows us to see connected process streams.

From this page you can navigate the project by links to connected entities, or look at various classifiers to get more understanding of their nature.

Using *Search All Reference Data* field you can search both local reference data library and all PCA RDL. Or you can go directly to the http://localhost:5000/entity?uri=http://posccaesar.org/rdl/RDS327239 and compare its rich information content with the PCA LD page for the PUMP - http://posccaesar.org/rdl/RDS327239 .

Linked Data extension allows you to explore any project in which there are data sources with module names **pca**, **projrdl** and **projdata**. You can have only one or two of these data sources in your project. For example, open PCA RDL in a new project, assign it **pca** module name in project properties and start Linked Data extension. Only one search field will be present on ***http://localhost:5000/.***