iTesla **Innovative Tools for Electrical System Security within Large Area**			
Grant agreement number	283012	**Funding scheme**	Collaborative projects
Start date	01.01.2012	**Duration**	48 months
Call identifier	FP7-ENERGY-2011-1		

Deliverable D2.3
Conversion tools from external formats to internal formats

Dissemination level		
PU	Public.	**X**
TSO	Restricted to consortium members and TSO members of ENTSO-E (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).	

Document Name: Conversion tools from external formats to internal formats
Work Package: WP2
Task: 2.3
Deliverable: D2.3
Responsible Partner: AIA

Author		Approval	
Name	Visa	Name	Date
Marc Sabaté [AIA] Geoffroy Jamgotchian [RTE] [Quinary]		Executive Board	

DIFFUSION LIST	
For action	For information
All partners	

DOCUMENT HISTORY

Index	Date	Author(s)	Main modifications
0.0	July 2013	Marc Sabaté	Document creation
0.1	July 31 2013	Geoffroy Jamgotchian	CIM import chapter added
1.0	August 1st 2013	Geoffroy Jamgotchian	Eurostag data import chapter added

Table of contents

Figures

1. INTRODUCTION

Deliverable D2.3 is of the "Prototype" type. It consists of three conversion tools from external data formats to the iTesla internal data model (IIDM):

- Import of data in the Eurostag format,
- Eurostag to Modelica data converter,
- Import of data in the CIM format.

Source codes of theses conversion tools can be found on the iTESLA source code Git repository hosted at www.bitbucket.org. The present document is an accompanying document that describes the main characteristics of these tools.

2. EUROSTAG TO MODELICA CONVERTER

2.1. General overview

Within the iTesla project, WP2 provides common services for WP3, WP4 and WP5. Among them we find the Eurostag to Modelica converter that translates Eurostag macroblocks to the Modelica language, using as input the **.frm, .fri and .par** Eurostag files. These files will be contained in the Dynamic database and by calling the Modelica converter from the user interface the corresponding Modelica file will be generated and stored inside the database.

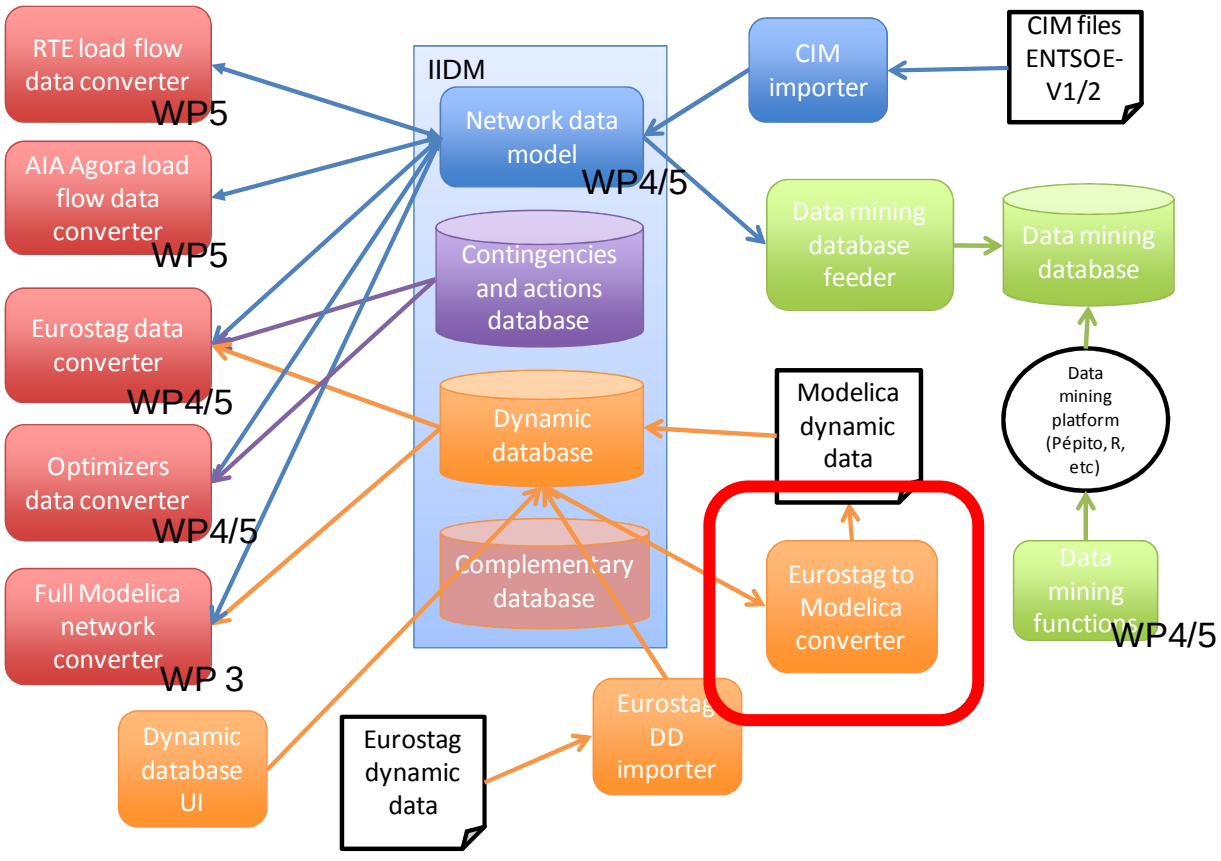


Figure 1- Eurostag to Modelica converter in the general WP2 workflow

This converter will also be a useful tool for WP3, WP4 and WP5. It will be used to populate the Dynamic database of the IIDM, which will be then used in the Eurostag data converter and in the Full Modelica network converter (see Figure 1.).

In this document, the development of the "Eurostag to Modelica converter" (highlighted block in figure 1.) is explained.

2.2. Analysis of the model file structures

2.2.1. Eurostag

A Eurostag macroblock is an assembly of elementary blocks with input-output relationship integrated into a system by defining the input variables for the existing types of machines in the file editor module. Each block of the macroblock performs a function (for further information, see [1] appendix 2) and has the following attributes:

- A block identifier.
- A graphical block identifier within the macroblock.
- From 0 to n inputs.
- A single output.
- From 0 to n parameters.
- From 0 to 1 initial value.
- From 0 to 1 offset.

Three ASCII files define a Eurostag macroblock: **.frm** file corresponds to the main scheme of the macroblock, **.fri** corresponds to the initialization scheme and **.par** corresponds to parameters.

2.2.1.1. Main Scheme

The **.frm** file defines the topology of the macroblock and has the following structure (for further details, see D2.2):

```
HEADER     03/10/08 4.4      # Header with the date and the number of version to be used with the regulation
    70    84 # Dimensions of the graphical representation (to be bypassed)
     5 # Number of blocs
 ?                    INTMAX               -KDELTAF             1.0                  ?#Blocs param
 ?                    1.0                  ?                    PROPHP*TRH           ?#Blocs param
 ?                    0.                   1.0                  TRH                  ?                    #Blocs param
 ?                    ?                    ?                    ?                    ?                    #Blocs param
 ?                    ?                    1.0                  ?                    ?                    #Blocs param
 ?                    ?                    ?                    ?                    ?                    #Blocs param
 ?                    ?                    KDELTAF              ?                    ?                        #Blocs offset
0.                   ^V1                  ?                    ^V1                  0.                   #Blocs init value
 ?        ?        OMEGA    ?        ?       #Blocs entries
 ?        ?        ?        ?        ?       #Blocs entries
 ?        ?        ?        ?        ?       #Blocs entries
?        ?        ?        ?        ?       #Blocs entries
 ?        ?        ?        ?        ?       #Blocs entries
$         /         &         $CM       $        #Blocs output
   1        2        4        5        6     # Graphical Blocks numbers
3741  3930  4153  4382  3930 # Location on the graphical representation (to be bypassed)(X axis)
3948  394839483948  4152 # Location on the graphical representation (to be bypassed) (Y axis)
    26     7     1    10    26   # Type of the bloc (need of a table of correspondence)
     0     0000# Orientation of the bloc (graphical: to be bypassed)
     1     2     4     5     6   # Graphical Blocks numbers, again???
     4 # Number of links
3822  40114011  4234 # Location on the graphical representation (to be bypassed)
...                  # Location on the graphical representation (to be bypassed)
  3990     0     00# Location on the graphical representation (to be bypassed)
     1     5     2     3   # Starting block of a link (blocs are 1,2,3,4,5, graphical numbers are: 1,2,4,5,6)
     2     3     3     4 # Ending block of a link
     1     5     3     1   # Number of entry in the extremity bloc
    10     3     1     1# (to be bypassed)
     0 # (to be bypassed)
     0 # (to be bypassed)
     0 # (to be bypassed)
     0 # (to be bypassed)
     0 # (to be bypassed)
```

This scheme will be the most important one for the conversion from Eurostag to Modelica language. It defines the topology of the macroblock: which blocks appear in the macroblock, their parameters and their connections.

2.2.1.2. Initialization Scheme

The **.fri** file indicates that the Computation Module needs to initialize a macroblock (i.e. to give initial values to all the state variables of the macroblock) within the simulation of a power system with Eurostag.

However, when a power system that contains a macroblock is simulated with Modelica, initialization calculations are previously performed with Eurostag, and the initial resulting values for the state variables of the macroblock are taken to initialize the Modelica macroblock. Hence, for the moment, the **.fri** file will not be taken into account when converting a Eurostag macroblock to Modelica code.

2.2.1.3. Parameters scheme

The **.par** file provides all the parameters of the macroblock. In case there is more than one occurrence of the macroblock in the system where it is integrated, the **.par** file also provides all the necessary sets of parameter values corresponding to each occurrence of the macroblock. The set used in the simulation will be specified into the dynamic data file.

```
HEADER     17/04/12 4.5 RTE

_end_comment_

FLR            1        2        3        4        5
ALPHATR  1.700000 2.400000 1.700000 2.400000       2.
DEUXPINU 314.1593 314.1593314.1593314.1593314.1593
K1       -5.10000 -5.10000 -3.40000 -3.40000 -2.64000
K2            18.      18.      16.      16.      22.
K3       18.70000 18.70000      12.      12.       8.
K4            13.5     13.5 11.70000 11.70000 14.40000
KE       .4900000 .4900000       1.       1. 3.300000
MESU          0.02     0.020.020.020.02
PLEXM          -4.      -4.    -3.46    -3.46    -3.46
PLEXP    4.600000 4.600000       4.       4.       4.
T              5.       5.       5.       5.       5.
TC            0.02     0.02     0.03     0.03    0.025
TE             0.2      0.2 .3300000 .3300000 .3200000
TF1      2.940000 2.940000 6.600000 6.600000       6.
TF2           0.94     0.94 4.200000 4.200000       4.
TOMD          0.02     0.020.020.020.02
TOMSL         0.04     0.040.040.040.04
TR             5.       8.       5.       8.       8.
```

2.2.2. Modelica

The Modelica code of a model that is formed by several blocks and their input-output connections has the following structure:

```
model gover1
  parameter Real init_1;
  parameter Real init_2;
  parameter Real init_5;
  parameter Real init_6;
  parameter Real INTMAX;
  parameter Real KDELTAF;
  parameter Real PROPHP;
  parameter Real TRH;
  PowerSystems.NonElectrical.Math.ImSetPoint ImSetPoint_1(V=init_1);
  PowerSystems.NonElectrical.Continuous.ImLimitedIntegrator ImLimitedIntegrator_1(
    Ymax=INTMAX,
    K=1.0,
    Ymin=0.,
    nStartValue=init_2);
  PowerSystems.NonElectrical.Math.ImSum5 ImSum5_1(
    a1=-KDELTAF,
    a2=0,
    a3=1.0,
    a4=0,
    a5=1.0,
    a0=KDELTAF,
    StartValue = false);
  PowerSystems.NonElectrical.Continuous.ImLeadLag ImLeadLag_1(
    K=1.0,
    T1=PROPHP*TRH,
    T2=TRH,
    nStartValue=init_5);
  PowerSystems.NonElectrical.Math.ImSetPoint ImSetPoint_2(V=init_6);
  PowerSystems.Connectors.ImPin p1;
  PowerSystems.Connectors.ImPin n1;
equation
  connect(ImSetPoint_1.n1, ImLimitedIntegrator_1.p1);
  connect(ImSetPoint_2.n1, ImSum5_1.p5);
  connect(ImLimitedIntegrator_1.n1, ImSum5_1.p3);
  connect(ImSum5_1.n1, ImLeadLag_1.p1);
  connect(p1, ImSum5_1.p1);
  connect(n1, ImLeadLag_1.n1);
  ImSum5_1.p2=0;
  ImSum5_1.p4=0;
end gover1;
```

Initial values declaration

Parameters declaration

Blocks declaration

Input and output connectors of the macroblock declaration

Connections between blocks

Input connectors of blocks that don't have input values

Figure 2 - Modelica structure

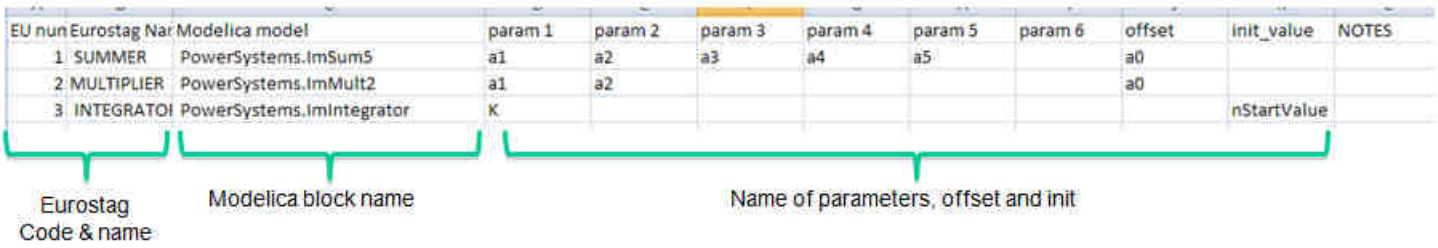
When integrating this model into a power system, initialization calculations must be performed using Eurostag in order to initialize each block in the Macroblock with the appropriate values. The reason for doing this is that the initial values of Macroblock regulators are closely related to the type of system to be regulated.

Therefore,the **.fri** file of the homologous Eurostag Macroblock cannot be taken into account until the full "system converter" from Eurostag to Modelica is implemented. At this stage, the converter is being developed with the goal of converting Macroblocks independently of the power system where they are going to be connected.

2.2.3. Tools needed for the converter

The first step in the converter is to match each block of a macroblock with the homologous block defined in the PowerSystems Modelica Library (see Deliverable D3.1 [3]), to identify the dependencies between blocks, and finally to rewrite the macroblock in Modelica language. It will be then necessary to endow each block of the Modelica library with the same attributes of the homologous Eurostag block.

On the other hand, the **.frm** file provides the Eurostag identifiers of the blocks used in the macroblock, the parameters of each block, the offset and the initial values (if they exist). Therefore, a sort of dictionary identifying each Eurostag block to the homologous Modelica model is needed; for this purpose a **correspondence table** has been created (see Annex). It identifies each block coming from Eurostag with the correspondent Modelica block, and it also provides (if they exist) the parameters names, the initial values names and the offset names of the Modelica block:



EU nun	Eurostag Nar	Modelica model	param 1	param 2	param 3	param 4	param 5	param 6	offset	init_value	NOTES
1	SUMMER	PowerSystems.ImSum5	a1	a2	a3	a4	a5		a0		
2	MULTIPLIER	PowerSystems.ImMult2	a1	a2					a0		
3	INTEGRATOI	PowerSystems.ImIntegrator	K							nStartValue	

Figure 3 - Correspondence table

Since the parameters in the **.frm** file are provided in a specific order, the parameters in the correspondence table must follow the same order.

In addition, for the correct compilation of the translated Modelica model, correct paths for the Modelica blocks have to be provided. In each new version of the library some blocks might change their location within the packages. Thereby, after each update, a text file will be automatically created with the correct paths of all the Modelica models in the PowerSystems library.

2.3. Converter

This chapter describes the Eurostag to Modelica converter. The source code is available on the iTESLA Git repository stored at www.bitbucket.com in the folder /itesla/Modelica/Eurostag to Modelica Converter/. The converter was built in Java, and it consists of a set of Java classes that implement the converter and a GUI mainly used for debugging and demonstration purposes. In the integrated platform, the output of the converter is only a .mo file stored in a preconfigured folder.

2.3.1. Packaging

The following modules are part of the converter:

- **iTESLA**
  - **iTesla.java:** it is the top level class of the converter. It is the entry point to open the GUI and to make the conversion.
- **Itesla.conversor**: this package contains the following classes:
  - **Block.java**: each object of this class is a Eurostag block with the information contained in the .frm file.

Block	
Param	**String[8].** It stores the parameters, the offset and the init value of each Eurostag block.
Entries	**String[5].** It stores the entries of each block.
Output	**String.** It stores the output of each block.
GraphicalNumber	**Integer.** It stores the Eurostag ID of each Eurostag block.
idEu	**Integer.** It stores the Eurostag ID of each Eurostag block.
Count	**Integer.** It stores how many times this type of block has appeared in the macroblock before.

  - **Elements.java**: each object of this class is a correspondence between a Eurostag block (identified by its Eurostag id) and its homologous Modelica block.

Element	
idEu	**Integer.** It stores the Eurostag ID of each Eurostag block.
nameEu	**String**. It stores the name of the Eurostag block.
pathModelica	**String.** It stores the path of the homologous Modelica block in the PowerSystems Library.
nameModelica	**String**. It stores the name of the homologous Modelica block.
Param	**List<String>.** It stores the name of the parameters of the Modelica block.

  - **ModelicaModel.java**: this class uses the previous classes to implement the conversion. It calculates and returns a list of lines for each section of a Modelica block code (see Figure2).

ModelicaModel	
outputHeading	**String.** It returns the Heading sentence of the Modelica model.
outputParamDeclaration	**List<String>.** It returns the parameters declaration of the Modelica model.
outputParamInit	**List<String>.** It returns the initial values declaration for the state variables.
outputPositivePin	**List<String>.** It returns the input pins declaration of the Modelica model.
outputNegativeImPIn	**List<String>.** It returns the output pins declaration of the Modelica model.

outputBlocksDeclaration	**List<String>.** It returns the blocks declaration of the Modelica model.
outputConnection	**List<String>.** It returns the declaration of the connection between blocks.
outputInputConnection	**List<String>.** It returns the existing connections between block and the input pin of the Macroblock.
outputOutputConnection	**List<String>.** It returns the existing connections between block and the output pin of the Macroblock.

- **Itesla.ui.main**&**iTesla.ui.util**: these packages contain a set of classes that define the GUI of the converter.

2.3.2. GUI

The converter's GUI has two panels and a browser. After selecting the Eurostag macroblock which is meant to be converted to Modelica code, the **.frm**, **.fri** and **.par** files are printed on each tab of the left panel, and after converting it to Modelica code the result is printed on the right panel, and it is saved as a **.mo** file.

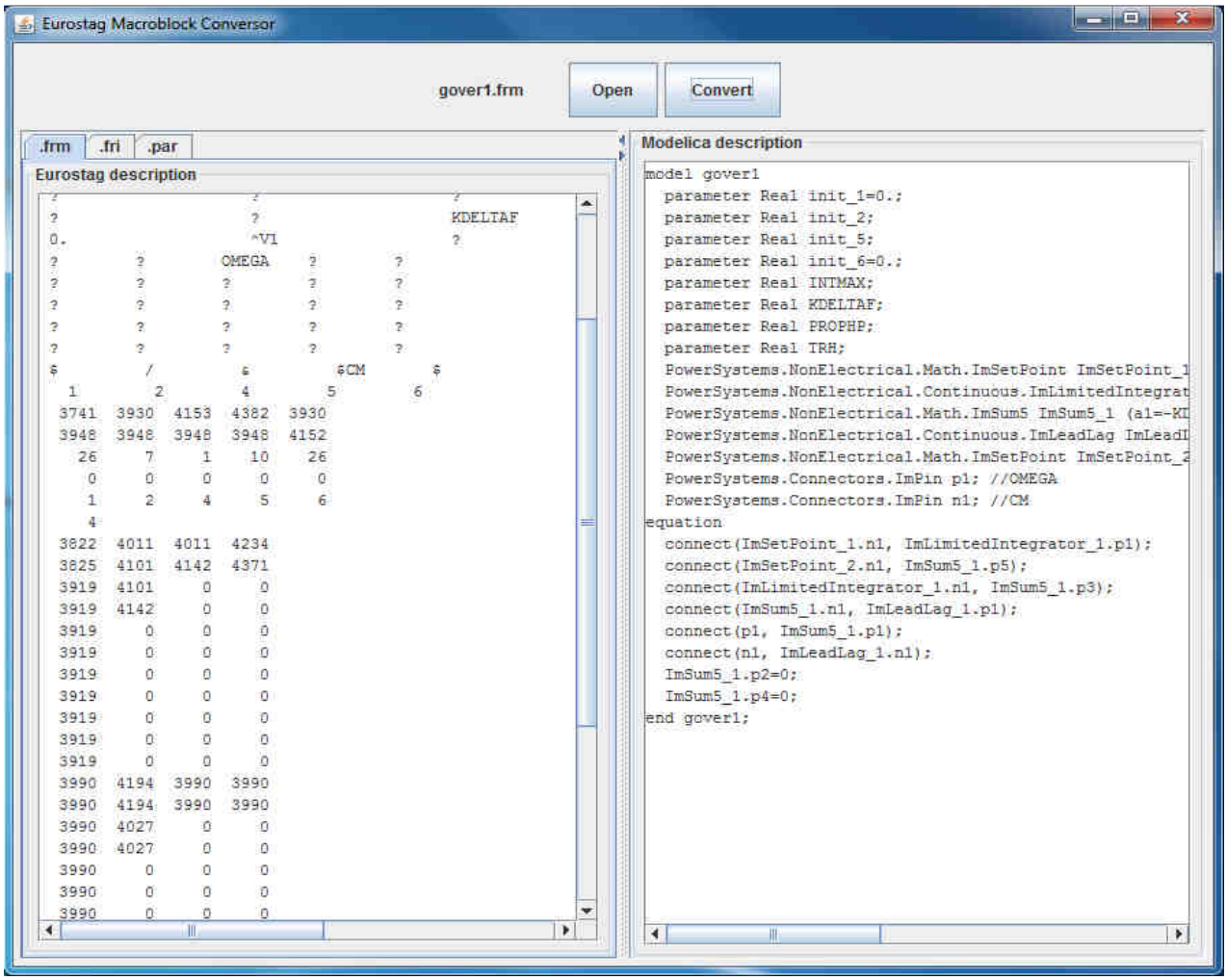


Figure 4 - Converter GUI

2.3.3. Conversion issues

At this point of the project, we have found one important issue to directly translate Eurostag macroblocks to Modelica macroblocks.

When a Eurostag macroblock contains one of the following blocks:

- Terminal Voltage.
- Current field.
- Active Power.
- Reactive Power.

In these blocks, the current measurement is needed to perform all the necessary calculations. Therefore, these particular blocks should be connected **in series** in order to have the current as an input value. This would imply inserting blocks in the power system (outside the macroblock) but as explained before, at this stage this converter is designed for converting Macroblocks independently of the power system where they are going to be connected.

Our solution for this issue is replacing these particular blocks by an input pin identified with the name of the corresponding block. Then, this block must be connected in series in the system (outside the macroblock) and connected to the Macroblock.

Another solution would be performing all the calculations of this block inside the machine, and inserting an additional output pin on the machine that connects directly to the macroblock. We will not use this option here for the moment.

2.3.4. Results

So far, we have tested the converter on the voltage regulators and voltage governors supplied in the Eurostag tutorial, and we have validated them in some simple test systems:

- GOVER1
- AVR2
- GOVER2
- AVR3
- GOVER3

The converter has also been tested on the excitation system of 64 blocks built in Modelica during the Pegase project (see [4]) in order to make the necessary improvements for the conversion of this type of complex macroblocks.

2.3.5. Next steps

Among the different steps that will be taken to enhance the converter, an important one involves the graphical part of the translated Macroblock. So far, when the converter translates a Macroblock to the Modelica code, it only generates the code, without generating the part concerning the graphical information of the model. If this translated model has to be eventually used by a user, then it would be appropriate to have this information in order to make the model more user-friendly. As it can be seen in the structure of a .frm file, there exists graphical information about the layout of the blocks in the **Eurostag model editor**. Therefore, it would be necessary to determine the necessary transformations to map this Eurostag graphical information to Modelica graphical information.

On the other hand, this translator will be integrated in a larger converter that will transform the Eurostag dynamic data of a system into a Modelica system. It will be adapted to be executed from an automatic call generated in the iTESLA platform in order to insert the generated file in the Dynamic database.

Finally, error codes will be defined with the aim of having warning messages when trying to convert blocks or components that do not exist in the PowerSystems library.

3. CIM TO IIDM CONVERTER

CIM files will be used in the iTESLA platform to provide network (static) data. As shown in the following diagram a converter has been developped to convert a CIM file to the iTESLA internal network data model.

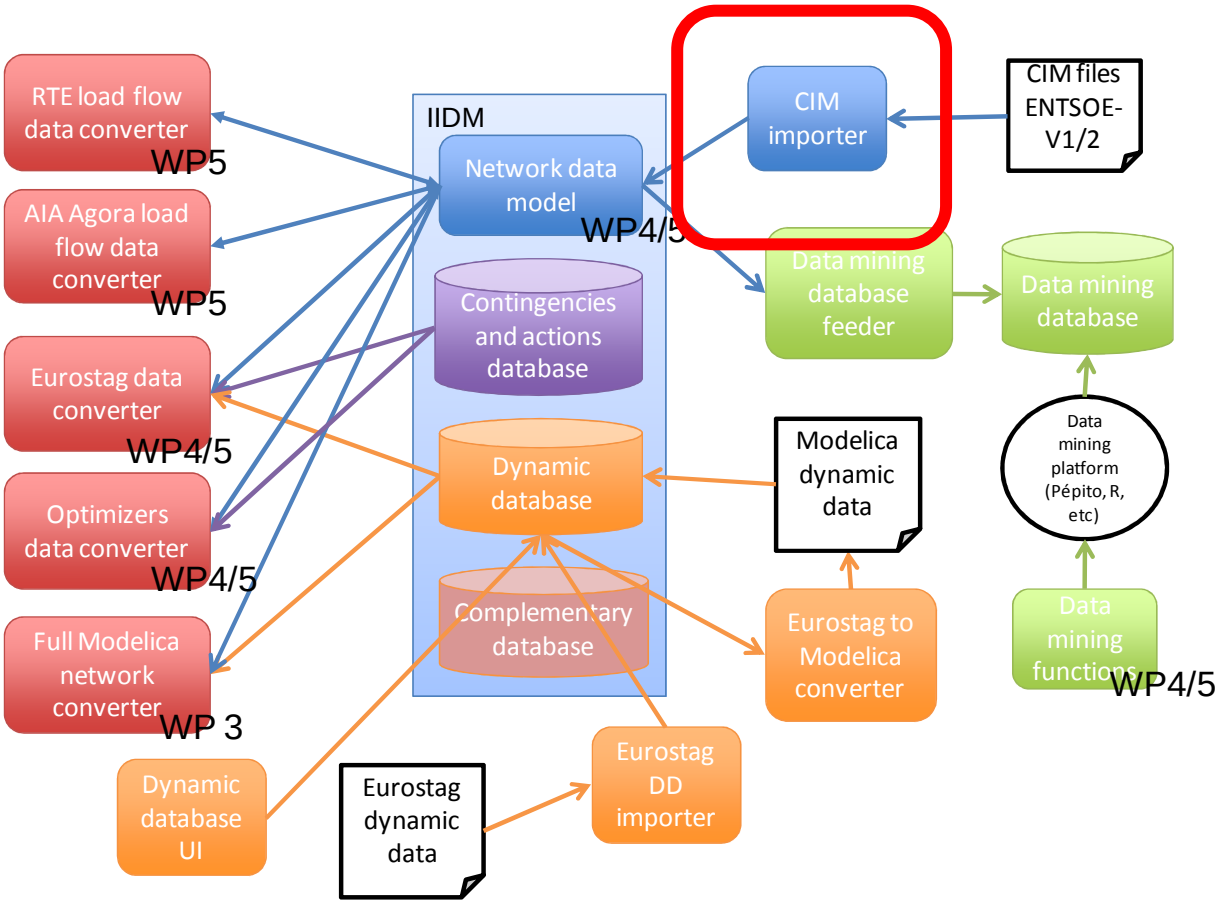


Figure 5 - CIM importer in the general WP2 workflow

3.1. CIM version supported

Only CIM ENTSOE profile is supported. The converter is actually based on the first version of the ENTSOE profile which is a restriction on the v14 general profile. The second version of the ENTSOE profile which is a restriction on the v16 general profile is not yet supported but will probably be before the end of the project.

3.2. Converter software architecture

CIM converter uses the RTE CIM gateway library to easily manage CIM RDF XML data. This library allows us to generate Java classes of the data model and the parser from CIM UML specification. Java class generation is integrated into the Maven build so that generated classes can be used and packaged with the converter. The next diagram describes the build workflow of the converter.

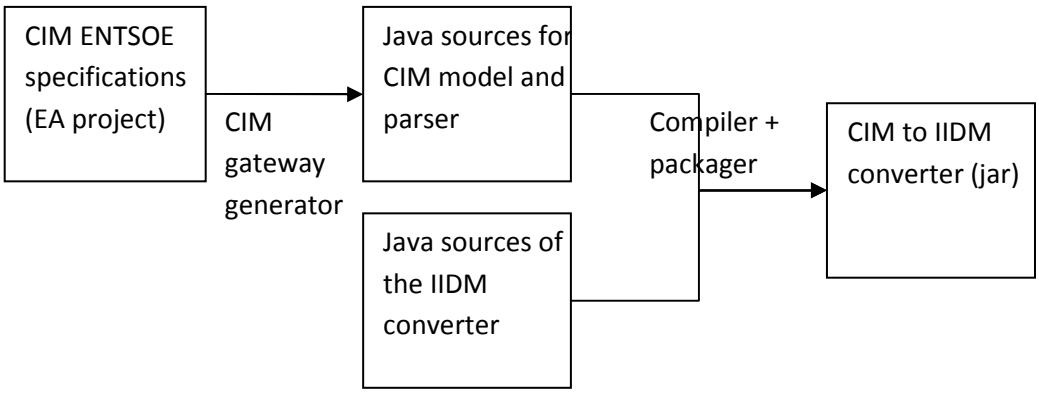


Figure 6 - Converter build workflow

Source code of the converter can be found on the iTESLA source code repository hosted on www.bitbucket.org. Here is the source code layout:

```
platform/cim1-import/
├── pom.xml
└── src
    └── main
        ├── java
        │   └── eu
        │       └── itesla_project
        │           └── iidm
        │               └── import_
        │                   └── cim1
        │                       ├── CIM1Exception.java
        │                       ├── CIM1Importer.java
        │                       ├── ModelConverter.java
        │                       └── Strings2.java
        └── resources
            ├── cim1
            │   ├── cim_IEC61970CIM14v02_ucte_extended_20090204.xml
            │   ├── ENTSO-E_Boundary_Set_EU_EQ.xml
            │   ├── ENTSO-E_Boundary_Set_EU_TP.xml
            │   ├── include_Edition1.xml
            │   ├── profile_1stEdition.simple-owl-augmented.xml
            │   └── subsets_IEC61970CIM14v02_EQ.TP.SV.xml
            └── META-INF
                └── services
                    └── eu.itesla_project.iidm.import_.Importer
```

`platform/cim1-import/src/main/java/eu/itesla_project/iidm/import/cim1` directory contains the Java sources of the converter. The most important file is ModelConverter.java which contains the CIM model (generated by CIM gateway) to IIDM model converter.

`platform/cim1-import/src/main/resources` directory contains the CIM ENTSOE V1 specification files which are used by the CIM gateway to generated the CIM in-memory model.

3.3. Using the converter

The next diagram shows how to use the CIM to IIDM converter to create an IIDM network model from a CIM file. The workflow is very simple, starting from a CIM file (or 3 CIM files, one for each data set), the converter allows us to create an in memory model of IIDM network.

Figure 7 - Converter run workflow

Here is a Java code snippet that shows the converter Java API. Using the converter is really straightforward as it only requires one line of code to create the IIDM network model (variable n).

```
import eu.itesla_project.iidm.import_.Importers;
import eu.itesla_project.iidm.network.Network;

public class Demo {

    public static void main(String[] args) {
        Network n = Importers.import_("CIM1", "/tmp", "mycimfile", null);
    }
}
```

4. EUROSTAG DYNAMIC DATA IMPORTER

A Eurostag dynamic data importer has been developed so that a TSO which has its dynamic data in the Eurostag format can easily feed the iTESLA platform with dynamic data and then run dynamic simulations on its perimeter.

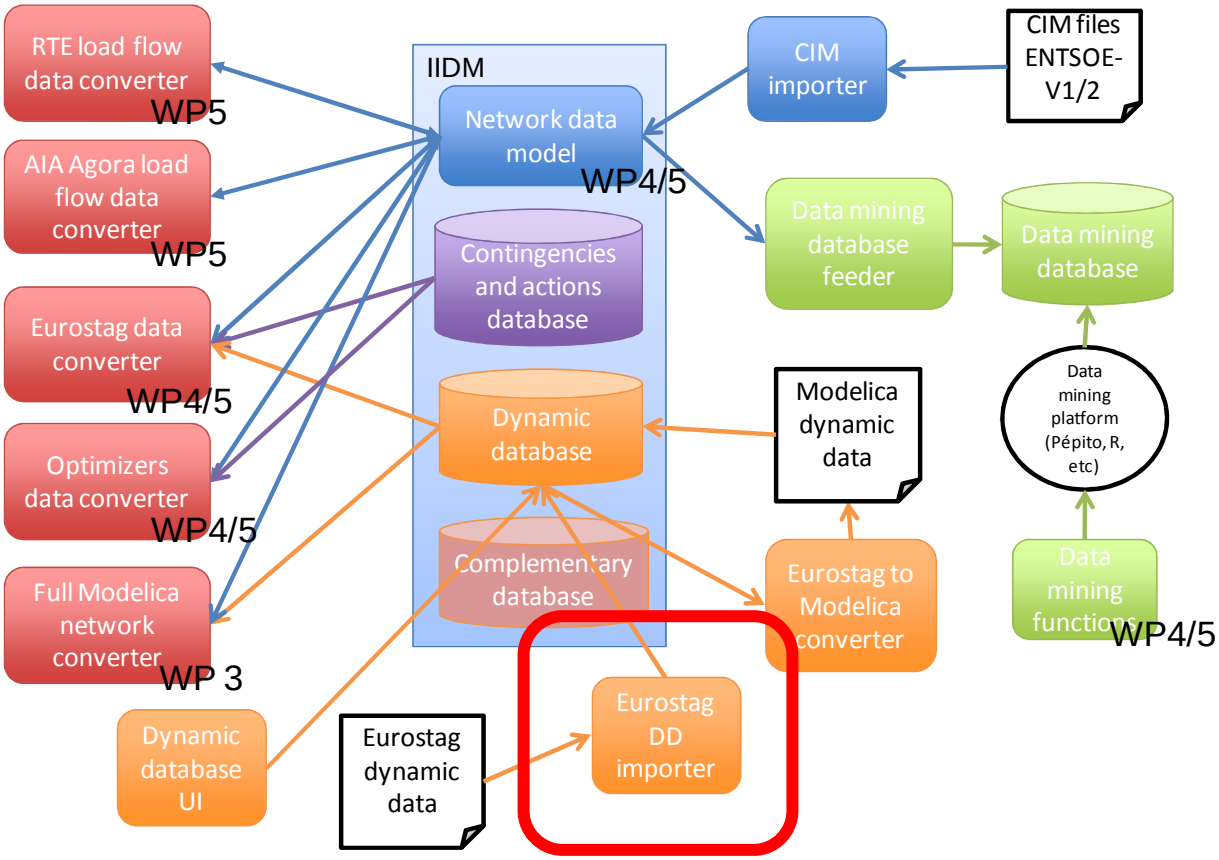


Figure 8 - Eurostag dynamic data importer in the WP2 general workflow

The previous diagram explains how the converter works. Starting from a .dd Eurostag file (.dta chunks) and a dictionary (CSV file) containing the mapping between Eurostag equipment identifiers and IIDM equipment identifiers (based on CIM ID), it is possible to automatically fill the dynamic database (see deliverable 2.2 for more detail on the dynamic database). This tool is callable from command line and only requires providing the directory where .dd files are stored, the identifiers dictionary and connection information to the JBoss application server which is responsible to manage the access to the dynamic database.

The source code is available on the iTESLA repository at www.bitbucket.com. It is made of Java classes grouped within *iidm-ddb-eurostag-import-export* Maven module. You will find next a dump of the source code tree. The main class is *eu.itesla_project.iidm.ddb.eurostag_imp_exp.DdbDtaImpExp.java* and contains most of the code that makes the conversion.

```
platform/iidm-ddb/iidm-ddb-eurostag-import-export
├── pom.xml
└── src
    └── main
        ├── java
        │   └── eu
        │       └── itesla_project
        │           └── iidm
        │               └── ddb
        │                   └── eurostag_imp_exp
        │                       ├── DdbDtaImpExp.java
        │                       ├── DtaParser.java
        │                       ├── EurostagRecord.java
        │                       ├── FortranFormat.java
        │                       ├── MemoryBufferedReader.java
        │                       └── utils
        │                           └── Utils.java
        └── resources
            └── log4j.xml
```

5. REFERENCES

[1] Tractebel – RTE. MODEL EDITOR USER'S MANUAL
[2] iTESLA Deliverable D2.2, WP2
[3] iTESLA Deliverable D 3.1 annex, WP3.2
[4] Pegase Project Deliverable D5.2.

-

6. ANNEX

The Modelica PowerSystem Library is being enlarged as new models are needed for the translation of new Eurostag macroblocks.

We have worked on a 22 buses system that should cover the 80% of the needed devices for a large system. This is the correspondence table with the dictionary between the Eurostag blocks and the existent homologous Modelica blocks in the PowerSystems library. This table is a .csv file read by the main module.

EU num	Eurostag Name	Modelica model	param 1	param 2	param 3	param 4	param 5	param 6	offset	init_value
1	SUMMER	PowerSystems.NonElectrical.Math.ImSum5	a1	a2	a3	a4	a5		a0	nStartValue
2	MULTIPLIER	PowerSystems.NonElectrical.Math.ImMult2	a1	a2					a0	
3	INTEGRATOR	PowerSystems.NonElectrical.Continuous.ImIntegrator	K							nStartValue
4	RELAY	PowerSystems.NonElectrical.Nonlinear.ImRelay								
5	SQUARE-ROOT	PowerSystems.NonElectrical.Math.ImSqrt								
6	SIMPLE-LAG	PowerSystems.NonElectrical.Continuous.ImSimpleLag	K	T						nStartValue
7	LIMITED-INTEGRATOR	PowerSystems.NonElectrical.Continuous.ImLimitedIntegra	Ymax	K	Ymin					nStartValue
8	LIMITED-SIMPLE-LAG	PowerSystems.NonElectrical.Continuous.ImLimitedSimple	Ymax	K	T	Ymin				nStartValue
9	DERIVATIVE-LAG	PowerSystems.NonElectrical.Continuous.ImDerivativeLag	K	T						pStartValue
10	LEAD-LAG	PowerSystems.NonElectrical.Continuous.ImLeadLag	K	T1	T2					nStartValue
11	CONSTANT	PowerSystems.NonElectrical.Math.ImConstant	K							
12	EXPONENTIAL	PowerSystems.NonElectrical.Math.ImExponential	A	B						
13	AND	PowerSystems.NonElectrical.Logical.ImAnd								
14	OR	PowerSystems.NonElectrical.Logical.ImOr								
15	FUNCTION	PowerSystems.NonElectrical.ImFunction	a	b						
16	GAIN	PowerSystems.NonElectrical.Math.ImGain	K							
17	ABS	PowerSystems.NonElectrical.Math.ImAbs								
18	POWER	PowerSystems.NonElectrical.Math.ImRampUpToAPower	A							
19	LIMITER	PowerSystems.NonElectrical.Nonlinear.ImLimiter	Ymax	Ymin						
20	DEAD BAND	PowerSystems.NonElectrical.Nonlinear.ImDeadBand	Xmax	Xmin						
21	HYSTERSIS									
22	MINIMUM_SELECTOR	PowerSystems.NonElectrical.Math.ImMin2								
23	MAXIMUN_SELECTOR									
24	VARIABLE-LIMITER	PowerSystems.NonElectrical.Nonlinear.ImVariableLimiter								
25	SCHMIDT-TRIGGER	PowerSystems.NonElectrical.Nonlinear.ImSchmidtTrigger	XMAX	XMIN	initValue					
26	SET POINT	PowerSystems.NonElectrical.Math.ImSetPoint								V
27	TERMINAL-VOLTAGE	PowerSystems.TERMINAL-VOLTAGE								
28	ACTIVE-POWER	PowerSystems.Electrical.Sensors.Eurostag.PwActivePower	UNIT	SNREF	SN	PN	PNALT	QNALT		
29	OUMAINT	PowerSystems.NonElectrical.Logical.ImOumaint								nStartValue
30	DELAY_2	PowerSystems.NonElectrical.Continuous.ImDelay_2	T	SimpleLagStartValue						
31	REACTIVE-POWER	PowerSystems.Electrical.Sensors.Eurostag.PwReactivePow	UNIT	SNREF	SN	PN	PNALT	QNALT		
32	MONOSTABLE	PowerSystems.NonElectrical.Nonlinear.ImMonostable	S	T						
33	DELAY_1	PowerSystems.NonElectrical.Continuous.ImDelay_1	T							
34	LN	PowerSystems.NonElectrical.Math.ImLN	a1						a0	
35	DIVIDER	PowerSystems.NonElectrical.Math.ImDiv2	a1	a2					a0	nStartValue
36	INVERSE_FUNCTION	PowerSystems.NonElectrical.ImInverseFunction	a	b						
37	SET-RESET	PowerSystems.NonElectrical.Math.ImSetReset								nStartValue
38	RESET-SET	PowerSystems.NonElectrical.Math.ImResetSet								nStartValue
39	SECOND_ORDER	PowerSystems.NonElectrical.Continuous.ImSecondOrder	A0	A1	A2	B1	B2			nStartValue
40	INTEGRATOR-FOLLOWER	PowerSystems.NonElectrical.Continuous.ImIntegratorFollo	K	T						
41	PULSE	PowerSystems.NonElectrical.Nonlinear.ImPulse	tau	S	T					
42	COSINE	PowerSystems.NonElectrical.Math.ImCosine								
43	SINE	PowerSystems.NonElectrical.Math.ImSine								
44	TANGENT									
45	ARC –TANGENT	PowerSystems.NonElectrical.Math.ImArcTangent	a1	a2	x0	x			offset	
46	ARC-COSINE									
47	ARC-SINE									
48	COUNTER									
49	REFER_FREQ									
50	FIELD_CURRENT									
51	LIMITED-LEAD-LAG									
52	RELAY-DELAY									
54	I2R									
55	I2I									
56	P2									
57	STATOR-STATUS									
58	FUNCTION_XY									
59	VOLTAGE-ANGLE									
60	CURRENT									
61	CURRENT ANGLE									

END