Documentation of the Python code allowing the loading of the SCL.  
  
**INTRODUCTION:**

The different programs (python’s class) take care of a subset of the SCL. At this stage, the program take care of all the public parts and also the Rte specifics tag. The goal is to create python’s classes and tables for objects such as Logical Node, Data Object etc… Some parts of the SCL remains to be fully tested and fully compliant with the SCL, this concerns the “services” part and the “substation” description part (lack of SCL file to validate the code.

The SCL is loaded using the ‘minidom’ library, a very common library to load XML files.

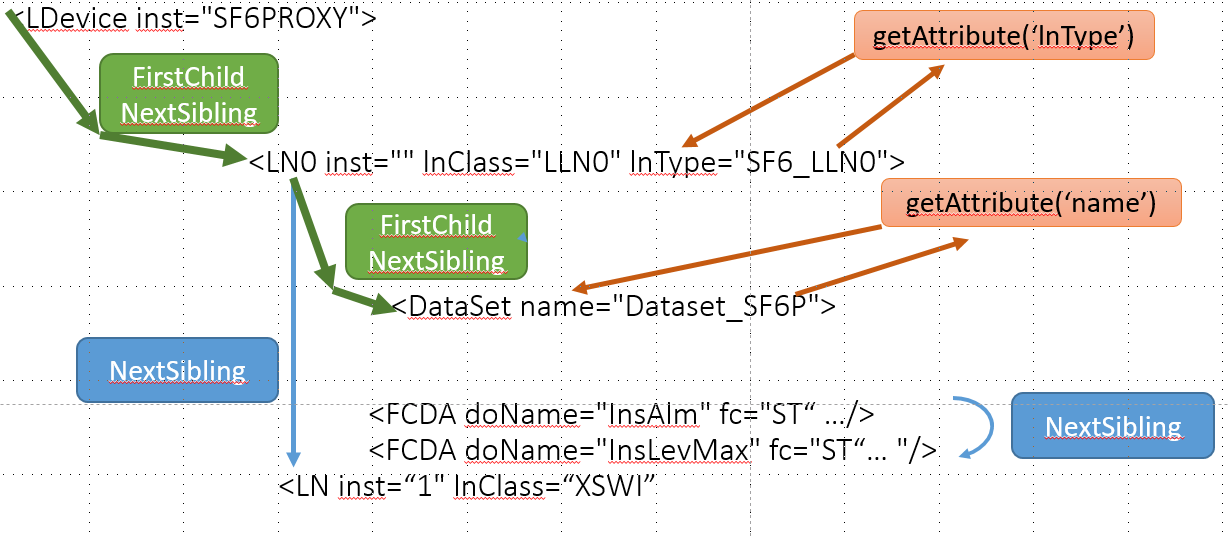
This library allows you to load an XML file, without checking the XSD of the XML in question. The library allows you to browse the tree of XML tags and retrieve attributes and their values relatively easily. The SCL management software uses two methods:

* A route of the tree for the parties:
  + Communication
  + IED / Accesspoint / Server / Data Model
  + Substaton
* A query of all instances of a specific tag, these are for:
  + LnodeType
  + DoType
  + DaType
  + EnumType
* scl = dom.parse(‘MyIED.ICD’)  
  DataType = scl.getElementsByTagName(**"DataTypeTemplates"**)

for DT in DataType # contains LNoType, DOType, DATyp and EnumType

DT = DT.firstChild.nextSibling # Acces to the branches

**if** DT.localName == **"LNodeType"**:



From the ‘minidom’ library it is require to use pointers to the SCL structure. The XML tree can be browsed by using pointers to different XML tag.

The next tag is accessed by using the nextSibling pointer, to look for a nested level, the combination of FirstChild and nextSibling will allow to get down by one level.

**CODE STRUCTURE**

The table below lists which parts of the SCL are analyzed by the different Python code. In general the name of the file contains the part of the SCL which is loaded in memory. For the vast majority of files, there is a unit test part according to the Python formalism, these tests are in the

* "if \_\_name\_\_ ==‘ \_\_main\_\_ ’:"
* …1

This is the classical way in Python to have “unit test”. When there is an issue (usually an exception…), it is convenient and faster to use the unit test code to debug the problem (instead of debugging the full application)

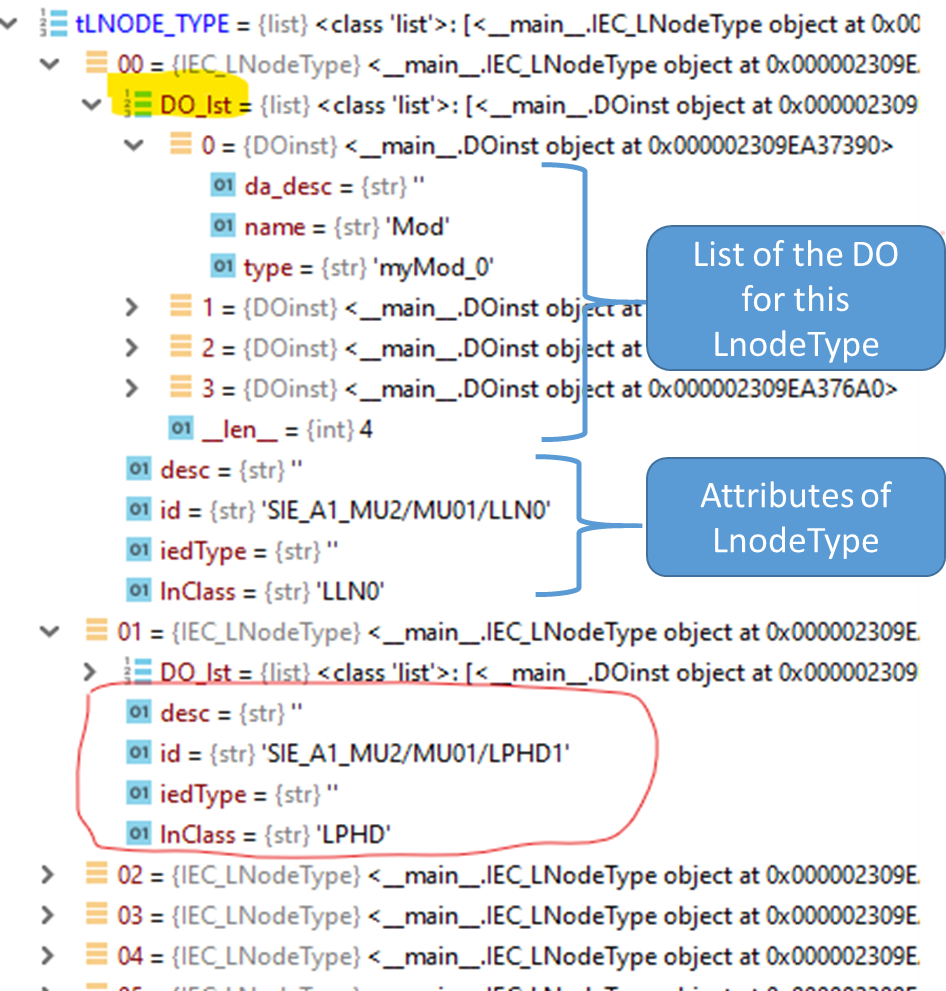
For this application, there are about 4 categories of Python classes:

* Data types for LN, DO, SDO, DA and Enum.
* The data model system parts <IED> <accessPoint><Server> … Logical Devices - LN
* The communication part
* The utility part: Trace, Sample file,

1. Data Type aspect:

|  |  |  |
| --- | --- | --- |
| **File Name** | **Purpose** | **Main data classes created** |
| IEC\_EnumType.py | Parsing the EnumTypes tags. | IEC\_EnumVal  IEC\_EnumType |
| IEC\_DAType.py | Parsing the DAType tags. | IEC\_DAType |
| IEC\_DOType.py | Parsing the DOType tags. | IEC\_DOType |
| IEC\_LNodeType.py | Parsing the LNodeType tags. | IEC\_LnodeType |

For each of these functions a table of the table of each data type is created and the list of it Data Objects (for Lnode), Data Attributes for Data Objects and so on…



1. IED and Server

|  |  |  |
| --- | --- | --- |
| **Filenames** | **Purpose** | **Main Classes** |
| IEC\_Server.py | Création de l’arborescence Server AccessPoint  DataModel/LD | Server  AccessPoint  IED  LDevice  Authentication |
| IEC\_LN.py | Créé la structure hiérarchique  IED/AP/LD/LN  Et les DOI. |  |
| IEC\_Services.py | Créé une structure (class) qui représente les Service du SCL |  |

1. Communication aspect

|  |  |  |
| --- | --- | --- |
| **Filenames** | **Purpose** | **Main Classes** |
| IEC\_Communication.py | The main purpose is to get all the IP addresses, MAC addresses and other connection related information. | SubNetWork  ConnectedAP  PType BitRate  SMV  TimeGSE  GSE  PhysConn  Communication |

1. Miscellaneous

|  |  |  |
| --- | --- | --- |
| **Filenames** | **Purpose** | **Main Classes** |
| IEC\_Trace.py | This is used to have control over software debug traces.  Various level of traces are available and be controlled by modifying the unit-test code for each class. | IEC\_Trace.py |
| IEC\_File.py | It is only a ready to use list of files for testing purpose.  There are IED levels files and system level files. | FILE.lstIed  FILE.lstSystem |
| IEC\_TypeSimpleCheck | To support the testing, this function verifies that a data of particular IEC basic types is actually in the theoretical range of its type. | Check.Type(type, value) |