FUHUA

IED Modeler/Designer

Version 2016X

User Guide
Part 3: IED Model Extension
Windows 7/8 32/64bit

August 2016

FUHUA

For more information please visit http://iedmodeler.com or contact us at info@fuhuatech.com.

FUHUA

Contents

FUHUA

1. Introduction

The IED Modeler/Designer is a comprehensive IED oriented SCL modelling tool for IEC 61850/61400 and companion standards. It has been designed to keep pace with the evolution of standards. Model Designer has been designed to address the diverse requirements of different types of users. It can be used by IEC 61850 Standard developers, IED vendors, stack vendors, researchers, engineers, utilities companies and people who are interested in applying Model Driven Architecture in system design. Feedback from this broad user base is allowing the product to evolve and we are continually adding new features to simplify the introduction of IEC 61850/61400 into production environments.

This user manual is delivered in four independent documents which need not be read in sequence.

Part One: IED Model Design
This section is fundamental and provides details on installation, GUI layout, system settings, project management, create/import/modify/export ICD/CID/IID, extract CID from SCD, and generation of TEMPLATE ICD etc., The Intelligent Creation feature greatly reduces the time and effort required to build error-free IED models. It assists in customizing data types, creation of LNs of different LNodeTypes, creation of DataSet and Control Blocks, initializing DOI values and configuring Communication parameters etc. These tasks can be conveniently completed with the help of user friendly Wizards. Batch editing of Attribute Values and Element Values is also now possible using external tools like Excel. Search Utilities allow convenient searching for project items. They use a Fuzzy search algorithm, allowing context search of documentation for each Element and Attribute.

Part Two: Domain Design
This section introduces simplified high level Domain design utilizing UML technologies. Like other UML tools in the market, Domain design is based on Diagrams. Here we introduce Data Type Diagrams and Domain Diagrams. These are much like Class Diagrams and Component Diagrams in UML but more powerful and convenient. The tool will remove the need for third party Domain support. It can simplify the creation of an Edition 2.0 package for Wind Power to a one-two hour task or even faster if worked on by an IEC member who is in charge of designing this domain.

Part Three: IED Model Extension
This section explains how to embed private model information into SCL without breaking any rules defined in IEC 61850-6. This capability is needed since SCL is often missing information e.g PLC logic equations and internal mappings etc., SCL usually does not address non-IEC 61850 and vendor-specific parameters configuration, all of which can be essential to running an application. Introducing this information requires model extension according to IEC 61850-6. Model Designer provides a cutting-edge, vendor-independent, flexible and programmable way to accomplish this task.

FUHUA

Part Four: IED Model Validation

This section introduces Schema Check, Integrity Check and Semantic Check against rules defined by IEC 61850 Standards. Schema Check is the most popular feature used in most commercially available tools to confirm that SCL is error free. However experience shows that Schema checking alone is insufficient to catch many errors in SCL dynamic structures and semantic constraints.

Consider the following example:

<FCDA lnClass="MMXU" fc="MX" daName="PhV.phsA.cVal.mag.i" lnInst="1" ldInst="LDPQ"/>
<FCDA lnClass="MMXU" fc="MX" daName="A.phsA.cVal.mag.i" lnInst="1" ldInst="LDPQ"/>

The Schema Check is OK for the two DataSet entries above. However they are actually incorrect according to IEC 61850-6. PhV.phsA is DO.SDO, not DA. They should be corrected to:

<FCDA lnClass="MMXU" fc="MX" doName=" PhV.phsA" daName="cVal.mag.i" lnInst="1" ldInst="LDPQ"/>
<FCDA lnClass="MMXU" fc="MX" doName="A.phsA" daName="cVal.mag.i" lnInst="1" ldInst="LDPQ"/>

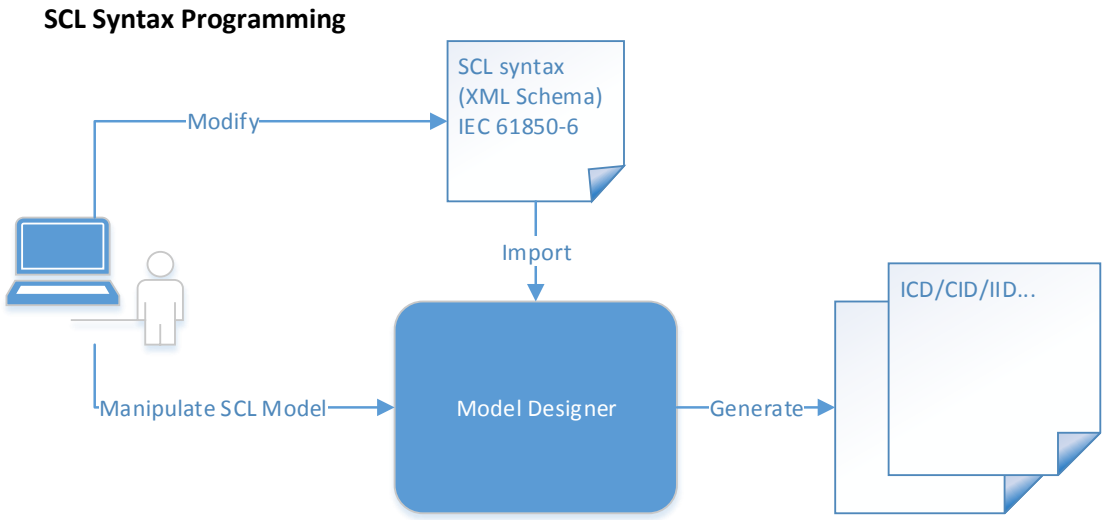
Schema Check is blind to these errors and tools from many vendors "accept" these errors. This results in IEDs arriving in production and not following rules defined in IEC 61850-6.
IED Model Validation is a critical. Model Designer implements Integrity Check which can detect errors that are blind to Schema Check.

Semantic Check is a feature under development and will be available in later releases.

FUHUA

2. IED Model Extension

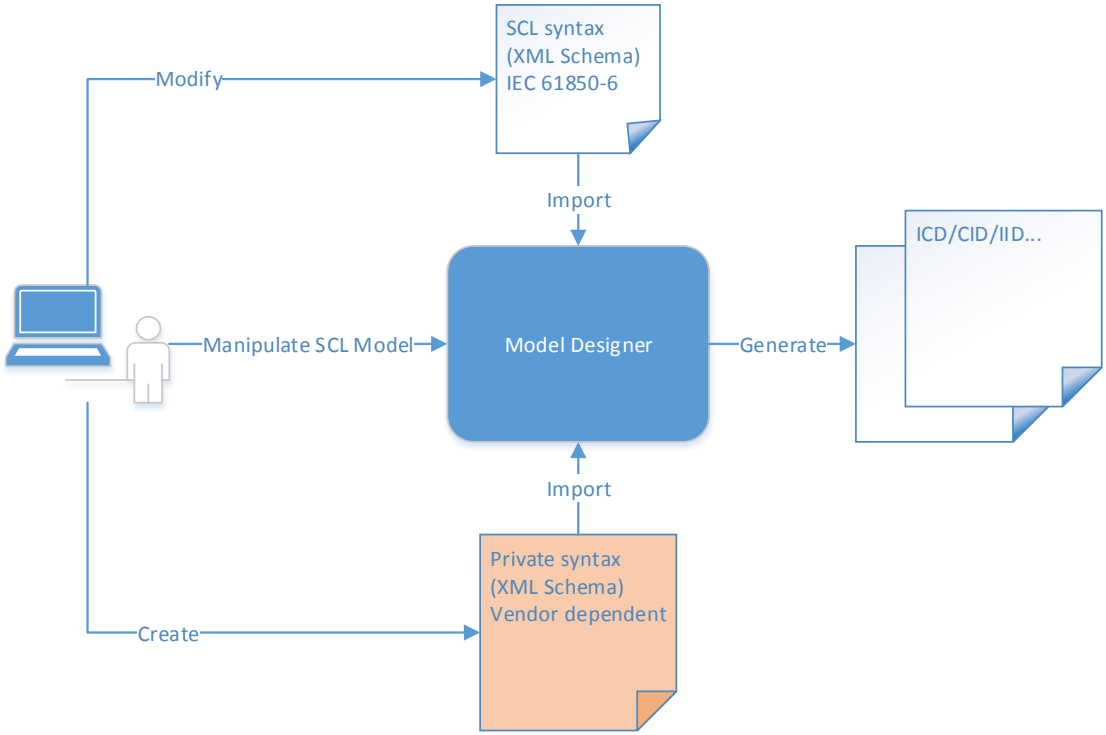
The Model Extension capability is one of the most outstanding features of the Model Designer. This Model Extension feature is vendor independent and based on XML Schema. The Model Designer has the ability to compile XML Schema data and produce Meta data which can later be used to create ICD/CID/IID. Since XML is considered a middle computer language with constructs and a comprehensive data type system, it is possible to program in XML. Model Designer is a tool which supports programs written in XML Schema:



The SCL Syntax is defined in an XML Schema specification by IEC 61850-6. However there are many variants used by different vendors throughout the world. Model Designer loads SCL Syntax dynamically and manipulates ICD/CID/IID accordingly. There is no need to update the Model Designer program assuming that the newly released version of SCL Syntax is backward compatible. Users can modify SCL syntax and then import it to the tool (picture above). The output ICD/CID/IID generated will be compliant with the SCL Syntax imported.

FUHUA

Private Namespace Programming



It is possible to develop a Private syntax to extend an SCL model according to rules defined by IEC 61850-6. Many companies making use of scl:Private elements to hold private model data for different application purposes. This is not however the only way to extend a model. Model Designer can understand a Private Syntax defined in XML Schema. Once the Private Syntax is imported the tool can assemble the extra model data (according to the Private Syntax) without breaking the SCL Syntax.

2.1 SCL Syntax Programming

This Section is for advanced users who have a complete understanding of the XML Schema defined by W3C – the SCL Schema as specified in IEC 61850-6 is defined in XML Schema.

To date there are two major editions of SCL Schema, namely edition 1.0 and edition 2.0. In practice vulnerabilities or open issues have resulted in many variants of editions derived from these two major schemas.

It is possible that some IEC 61850 users may need to modify the SCL Schema to address these types of issues. Any modification to SCL Schema affects the structure, syntax and semantics of the SCL model. Most tools in the current marketplace cannot provide a "transparent" implementation of SCL Schema Programmable.

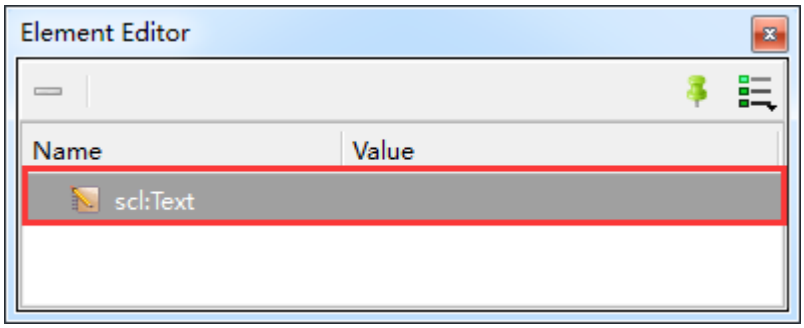
SCL Schema Programmable means that changes/modifications made to SCL Schema automatically program the behavior of the tool.

Using scl:Text as an example:

In SCL_BaseTypes.xsd it is defined as:

```
<xs:complexType name="tText" mixed="true">
   <xs:complexContent mixed="true">
         <xs:extension base="tAnyContentFromOtherNamespace">
               <xs:attribute name="source" type="xs:anyURI" use="optional"/>
         </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

From the definition we know that any Element instance of tText doesn't have any Text Node. A user cannot assign any text value to it.

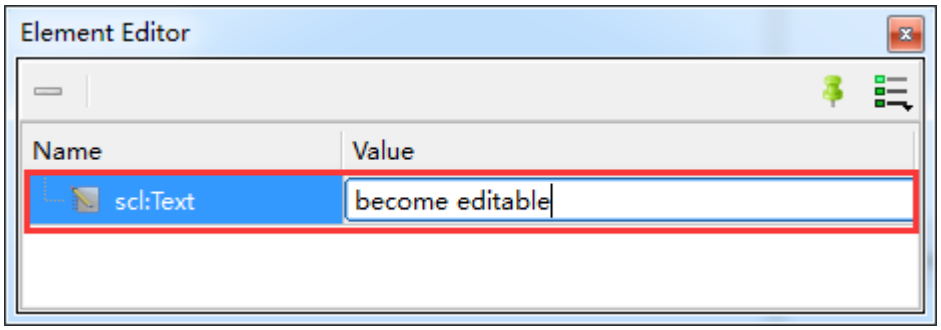


The scl:Text is grayed i.e. editing not possible.

If the tText definition is changed as follows:

```
<xs:complexType name="tText" mixed="true">
        <xs:simpleContent>
            <xs:extension base="xs:normalizedString"/>
        </xs:simpleContent>
    </xs:complexType>
```

The tool automatically reconfigures to make the field editable.

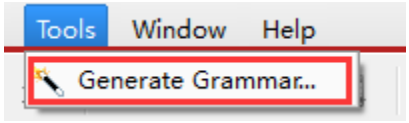


Now scl:Text is no longer grey and is editable.

The example illustrates how a user can control the tool behavior by programming the SCL Schema (without requiring modifications to the source code of the tool). In this way the tool can easily adapt and keep pace with the evolution of IEC 61850. A detailed knowledge of XML Schema is required to use the tool in this way and this is outside the scope of this documentation. This is a powerful feature that sets the tool apart from other tools in this space.

The tool loads a grammar file that is generated from the SCL Schema and other collected information. This grammar file is internal flexible metadata that instructs the tool on how to behave/respond to events issued by users/system etc.

To generate a grammar file click "**Generate Grammar…**" from **Tools** menu:



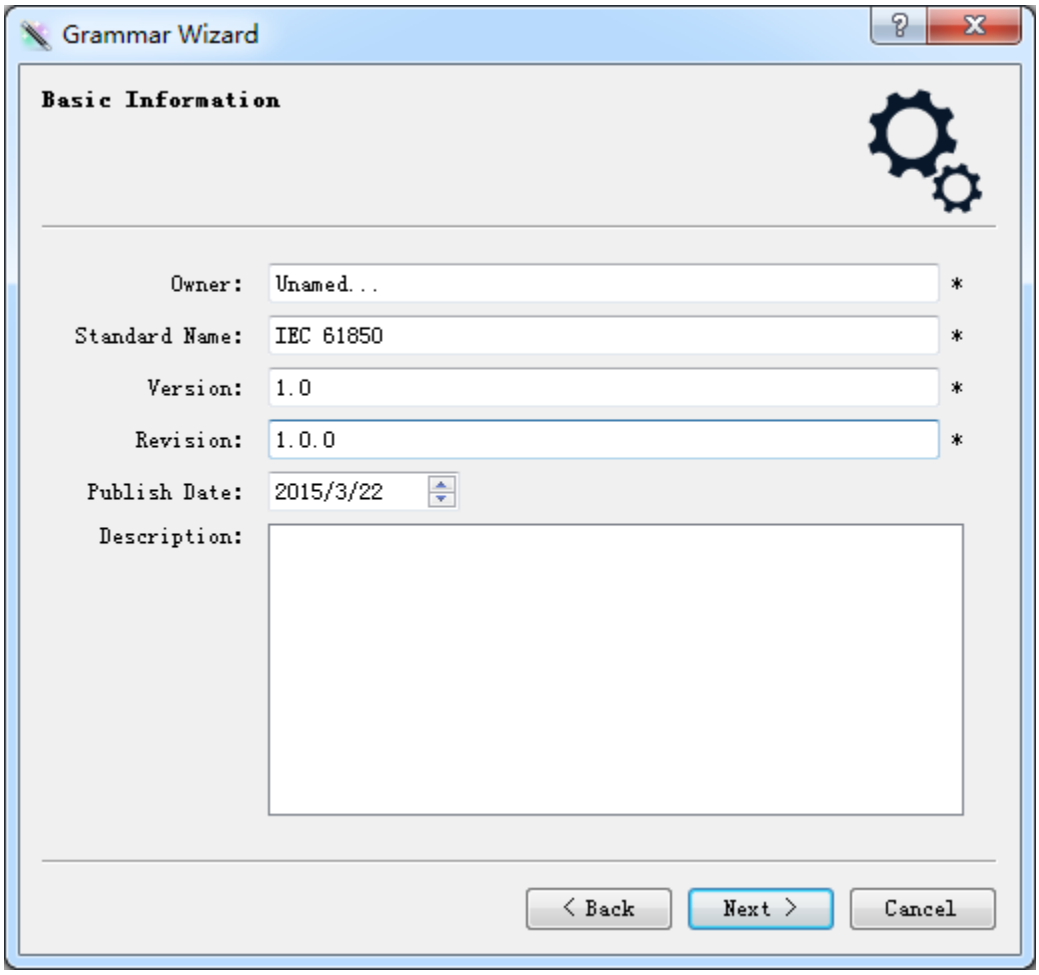
- A wizard appears

FUHUA

Note: Making a backup grammar is important for beginners

- Click Next

FUHUA



Owner: User who is generating the grammar.

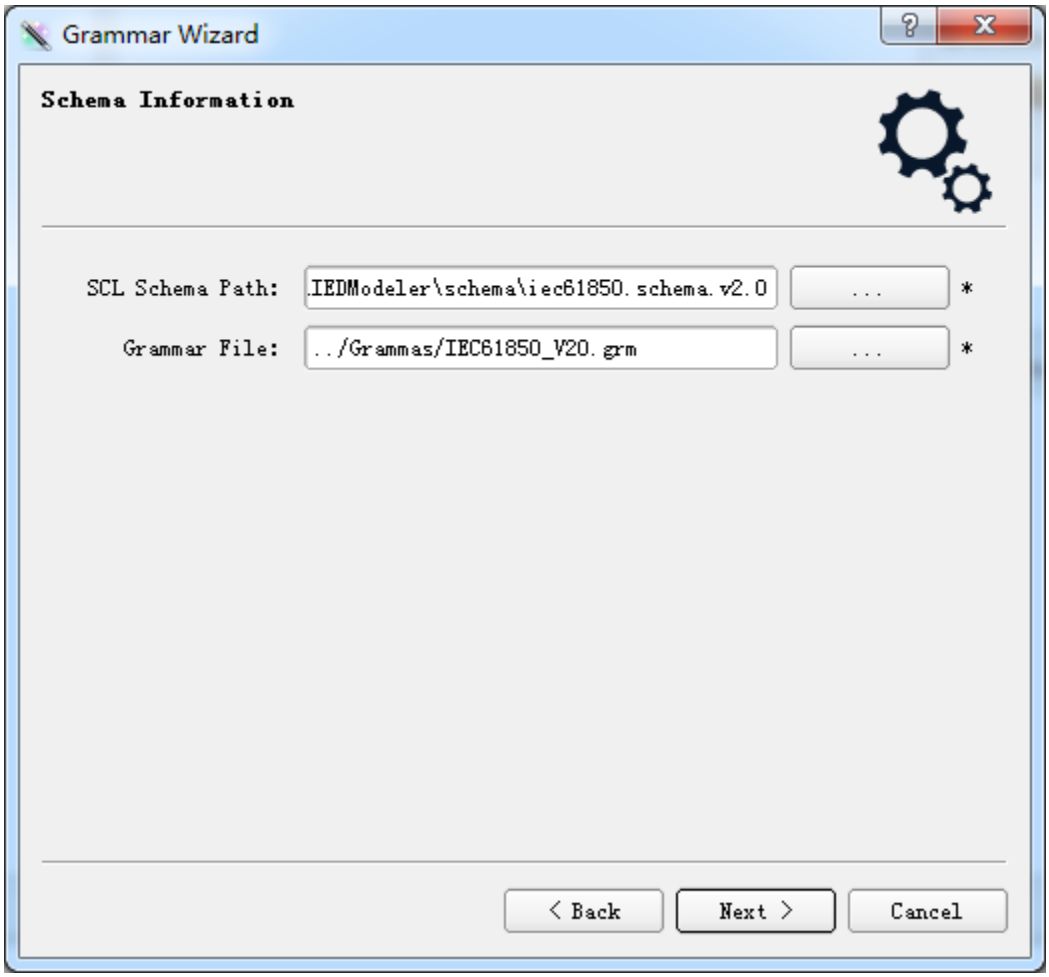
Standard Name: The standard on which the grammar is based.

Version/Revision: Version control options.

Publish Date: The date of publication of the grammar.

Description: Documentation / notes about the grammar.

- Click Next



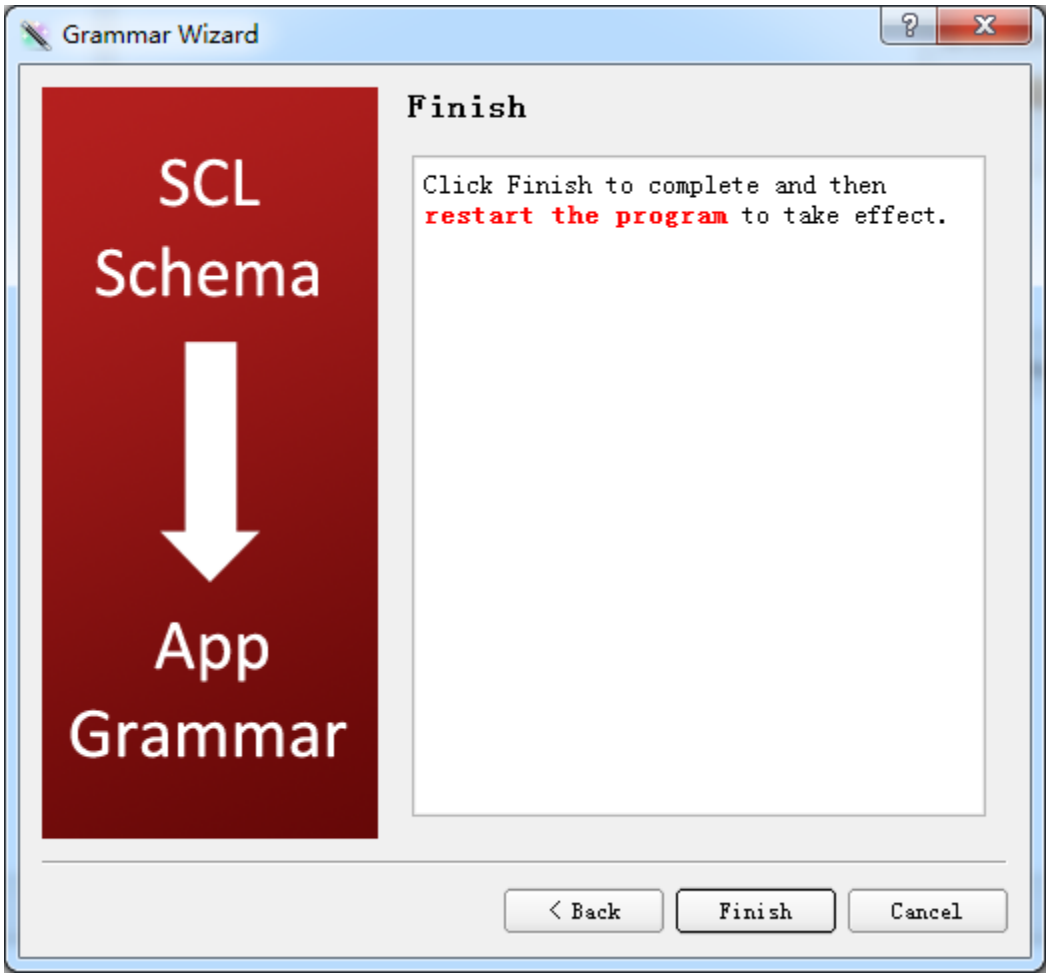
SCL Schema Path: The root directory of SCL Schema.

Grammar File: Location where the grammar file should be generated.

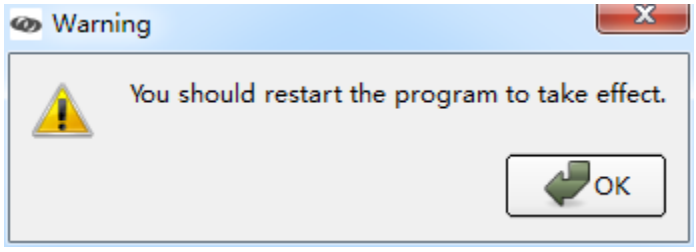
Note: When generating a grammar file in a directory other than "../Grammas", remember to place it under "../Grammas" to take effect.

- Click Next

FUHUA



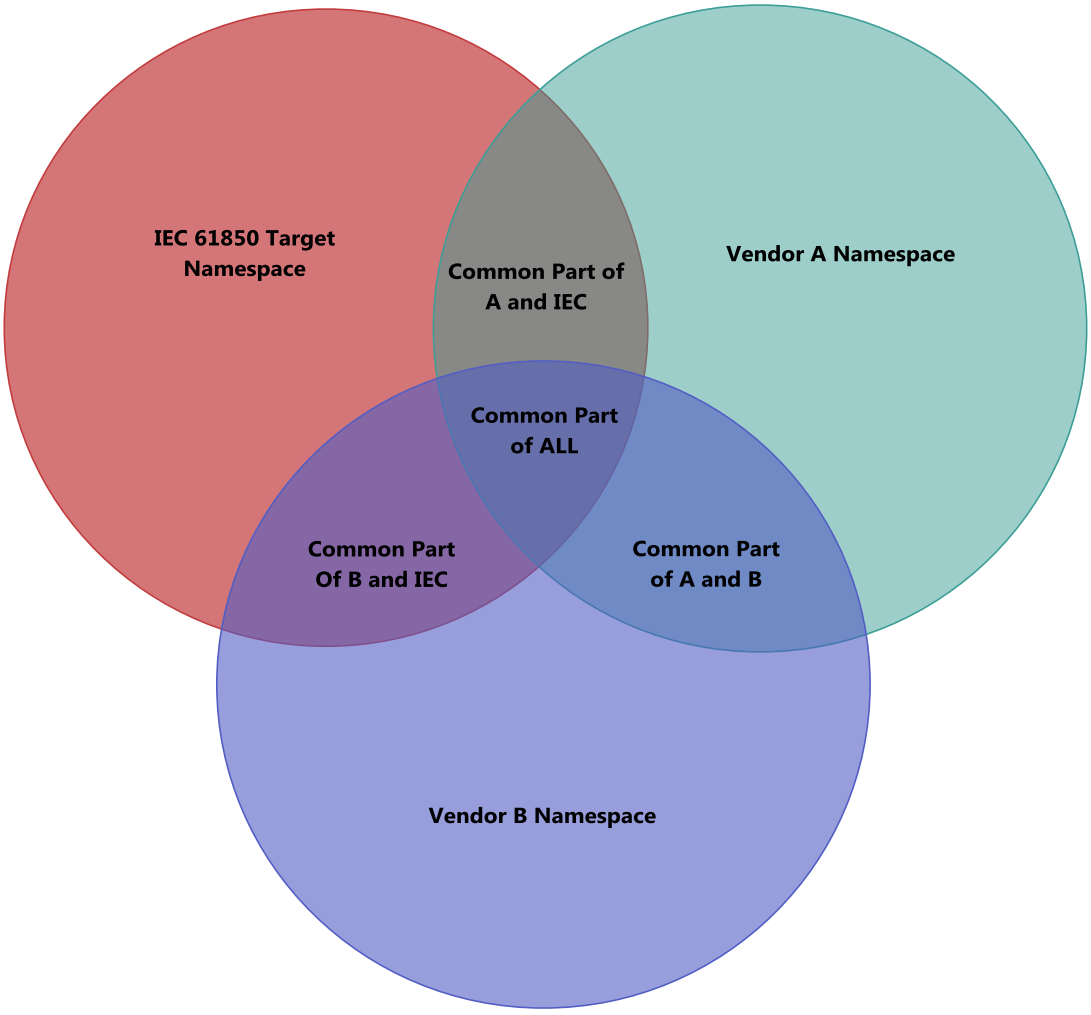
➢ Click Finish



Restart the program for the new grammar to take effect.

FUHUA

2.2 Private Namespace Programming



The diagram above is a practical illustration of the namespaces problem.

The diagram contains 3 namespaces: IEC 61850 (IEC), Vendor A and Vendor B. The Vendor A and Vendor B spaces claim IEC 61850 compatibility of a certain percentage because they have information in common with IEC 61850. However both of them keep large numbers of elements private for:

- Compatibility with other local standards (legacy issue)
- Implementation of new functions that are not yet addressed by IEC 61850
- Hiding the implementation details of their own IED(s)
- Other vendor proprietary reasons

A case where all 3 namespaces perfectly overlapped (100% overlapped) would indicate complete standardization. Real world implementations are currently far from perfect because these gaps exist between vendors and IEC 61850. Our challenge is to deliver a tool designed to accommodate current IEC 61850 and existing vendor implementations. Part of our solution to this problem is Private Namespaces Manipulation based on XML Schema.

2.2.1. How to define XML Schema for Private Namespace?

This is an advanced issue and users should have a good command of XML Schema to proceed. XML Schema knowledge is required and outside the scope of this documentation. The following is a simple example of how to map IEC 60870-5-104 to IEC 61850 as an introduction of the feature.

2.2.1.1. Overview of Example

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:IEC_60870_5_104="http://www.iec.ch/61850-80-1/2007/SCL"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.iec.ch/61850-80-1/2007/SCL"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="GlobalAddress104">
        <xs:annotation>
            <xs:documentation>Data Point address</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:attribute name="casdu" type="xs:integer" use="required"/>
            <xs:attribute name="ioa" type="xs:integer" use="required"/>
            <xs:attribute name="ti" type="xs:integer" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:attributeGroup name="agVendor">
        <xs:attribute name="attA" type="xs:boolean"/>
```

```
        <xs:attribute name="attB" type="xs:integer"/>
        <xs:attribute name="attC" type="xs:float"/>
        <xs:attribute name="attD" type="xs:normalizedString"/>
    </xs:attributeGroup>
</xs:schema>
```

Note: The example is present in the *AppRoot/examples* directory.

The XML text above defines Private Namespace (http://www.iec.ch/61850-80-1/2007/SCL) with a prefix named **IEC_60870_5_104**.

Currently the tool only recognizes global (top most) definitions of **Element** and **AttributeGroup**.

The example above has one global **Element** definition (**GlobalAddress104**) and one global **AttributeGroup** definition (**agVendor**).

2.2.1.2. AttributeGroup

```
<xs:attributeGroup name="agVendor">
        <xs:attribute name="attA" type="xs:boolean"/>
        <xs:attribute name="attB" type="xs:integer"/>
        <xs:attribute name="attC" type="xs:float"/>
        <xs:attribute name="attD" type="xs:normalizedString"/>
</xs:attributeGroup>
```

The XML text above defines four optional Attributes:

Name	Type	Condition
attA	xs:boolean	optional
attB	xs:integer	optional
attC	xs:float	optional
attD	xs:normalizedString	optional

Note: Not all the **Simple Types** defined by XML Schema are supported by the tool. The supported types are xs:boolean, xs:integer, xs:float, xs:normalizedString; any other types not listed will lead to undefined behavior.

2.2.1.3. Element

```
<xs:element name="GlobalAddress104">
        <xs:annotation>
            <xs:documentation>Data Point address</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:attribute name="casdu" type="xs:integer" use="required"/>
            <xs:attribute name="ioa" type="xs:integer" use="required"/>
            <xs:attribute name="ti" type="xs:integer" use="required"/>
        </xs:complexType>
</xs:element>
```

The XML text above defines an Element GlobalAddress104, which has an annotation described as “Data Point address”. The element has three attributes:

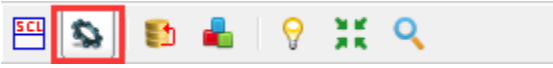
Name	Type	Condition
casdu	xs: integer	Required
ioa	xs:integer	Required
ti	xs: integer	Required

The section below demonstrates how to make use of this XML Schema as **Model Extension**.

2.2.2. Import Private Namespace

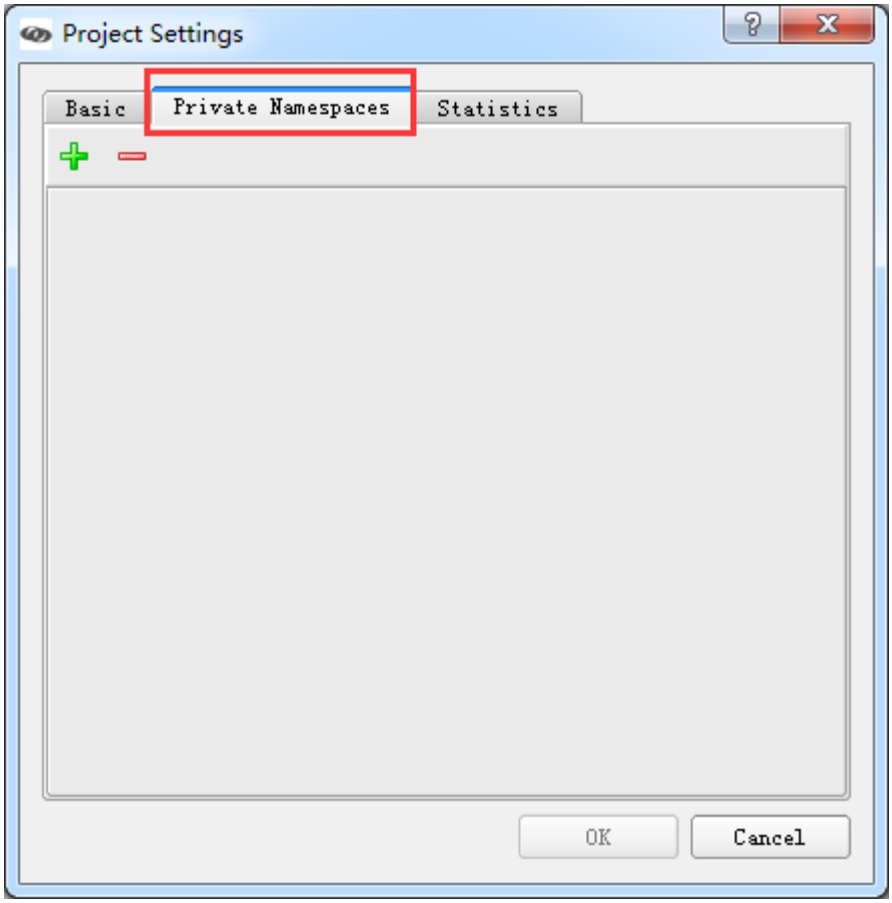
- Click “**Project Settings**” button from the toolbar

FUHUA

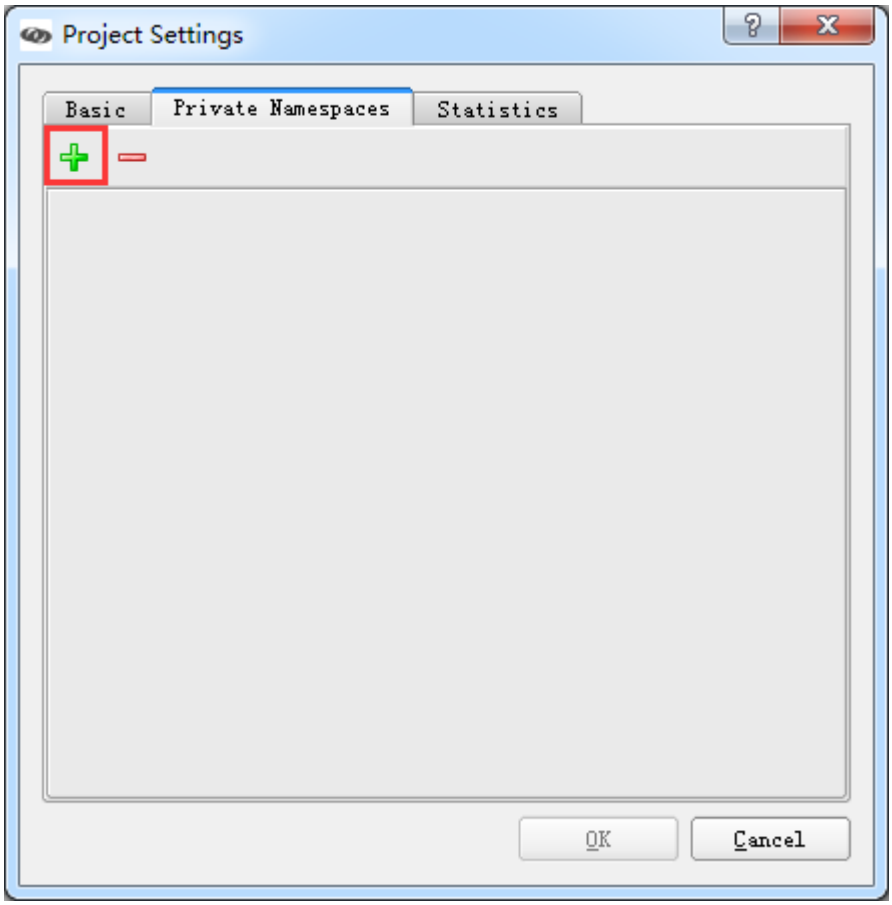


Note: If the project hasn't been saved user will be prompted to save it before continuing.

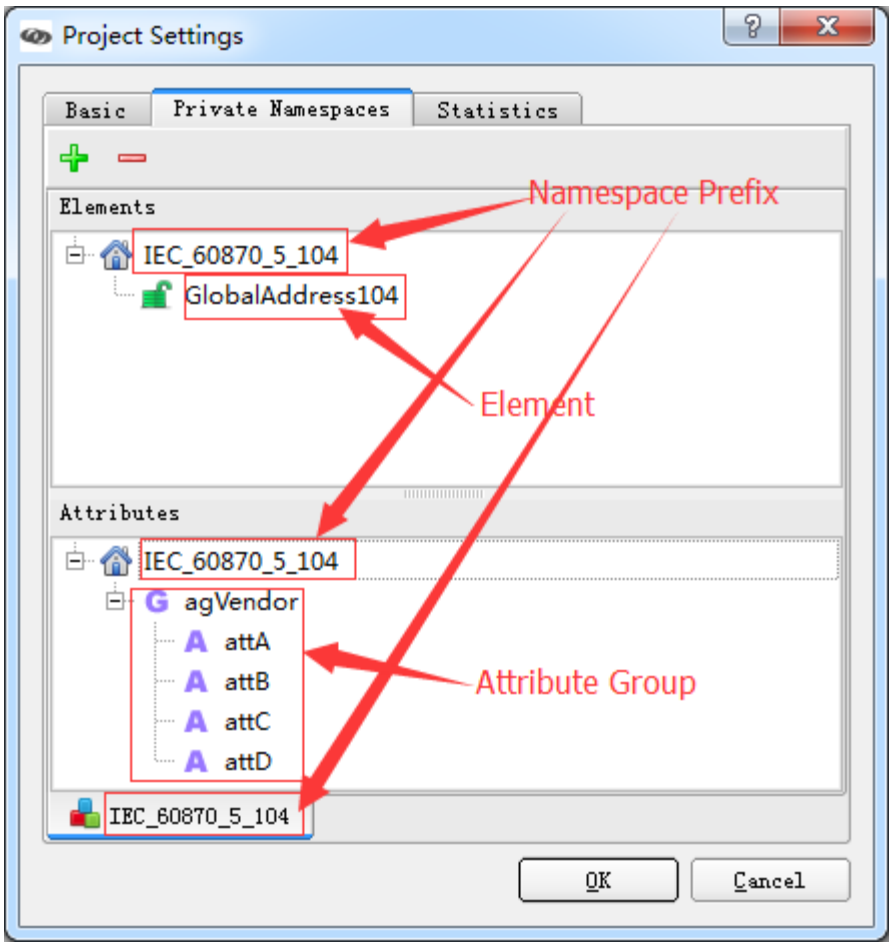
- Switch to **Private Namespaces** page



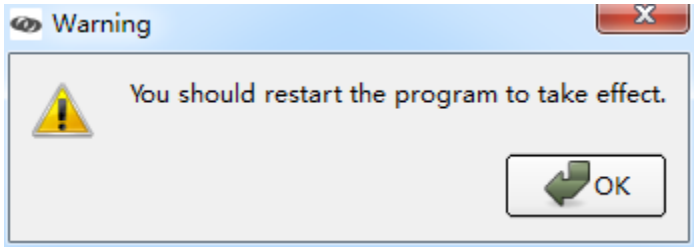
- Click Add button



- Browse and select IEC_60870_5_104.xsd file

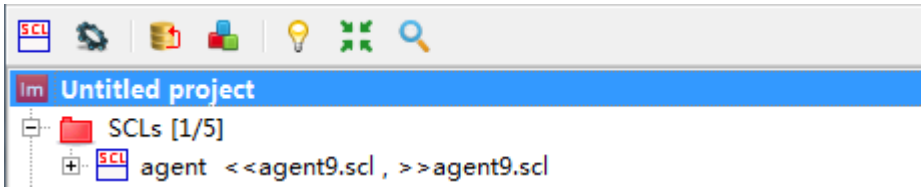


➢ Click OK and Restart the tool



Note: Don't forget to save the project before closing the tool.

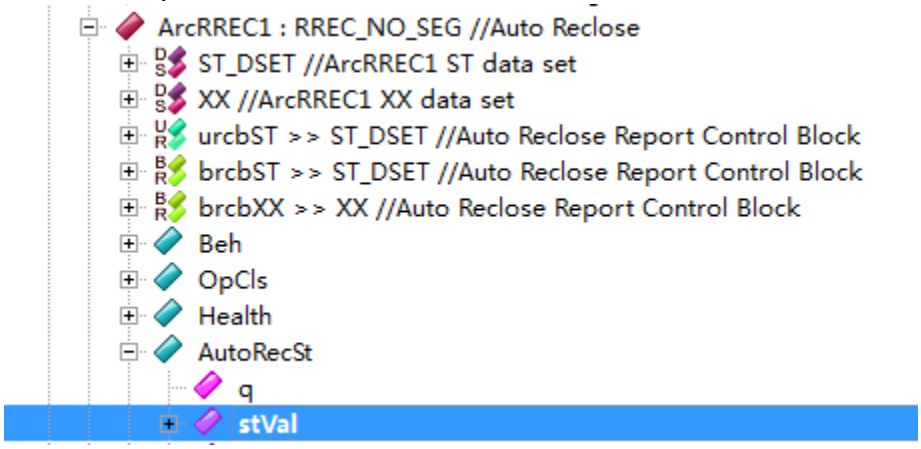
➢ Reopen the Project



FUHUA

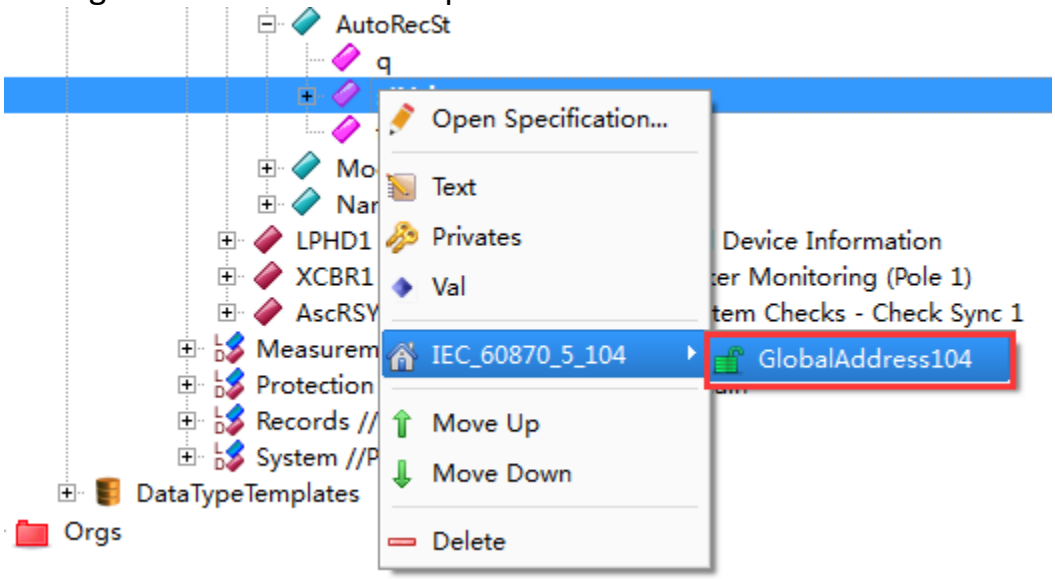
2.2.3. Create Private Element

- Select parent **Node** to add **Private Element**

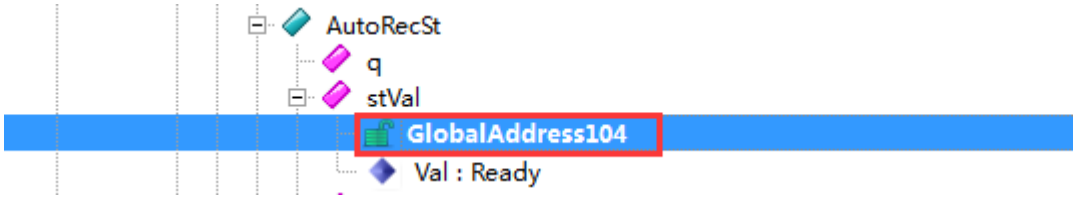


We select node **stVal** to add Private Element as model extension.

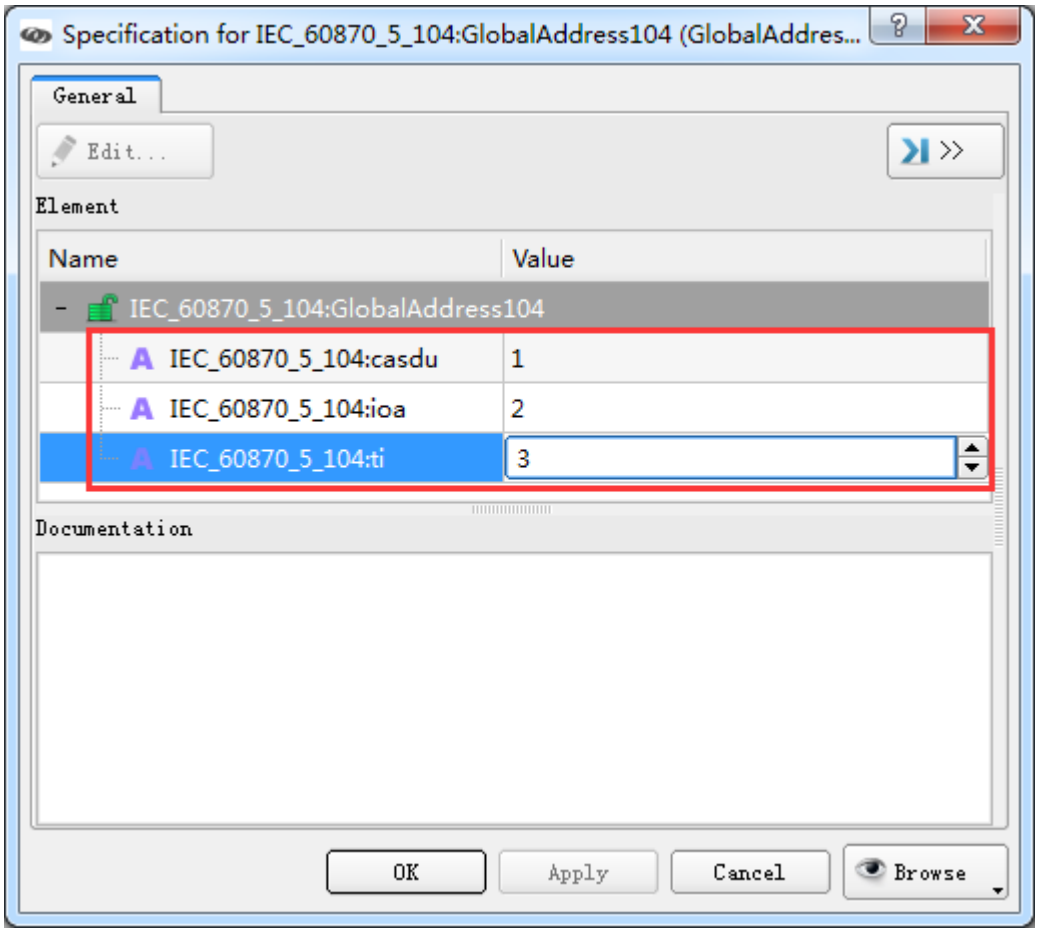
- Right click "**stVal**" to add private element



- Double click "GlobalAddress104"

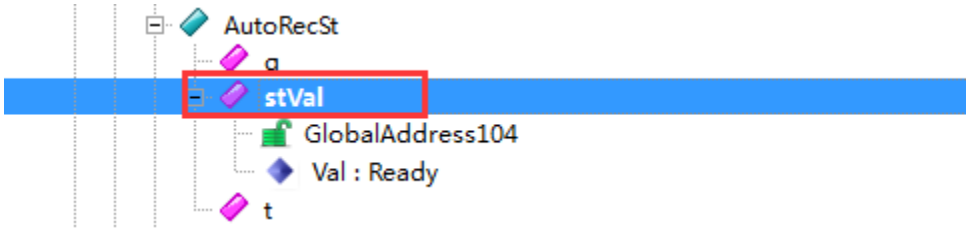


- Edit "GlobalAddress104"



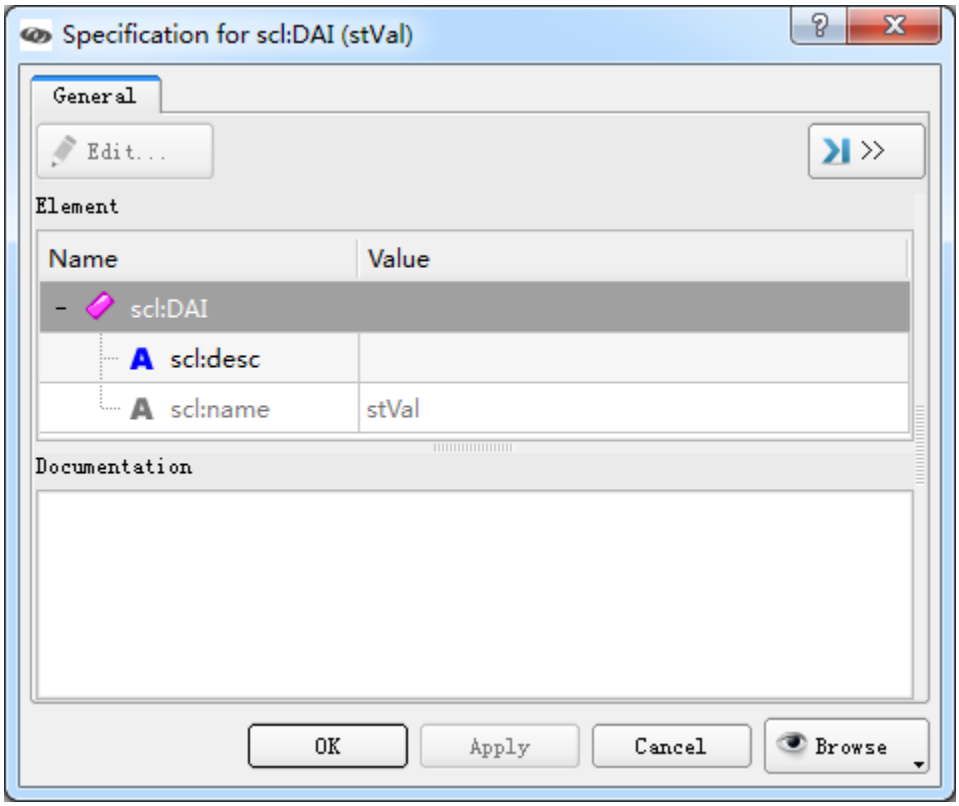
2.2.4. Create Private Attribute

- Select parent Node to add Private Attributes

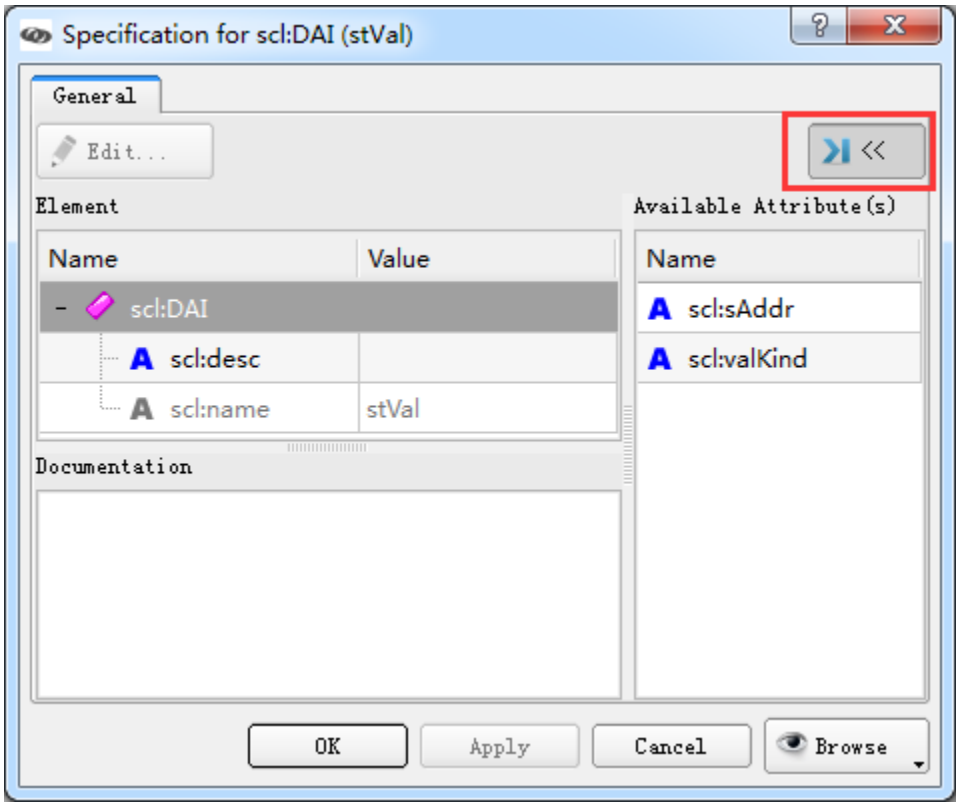


- Double click “stVal”

FUHUA

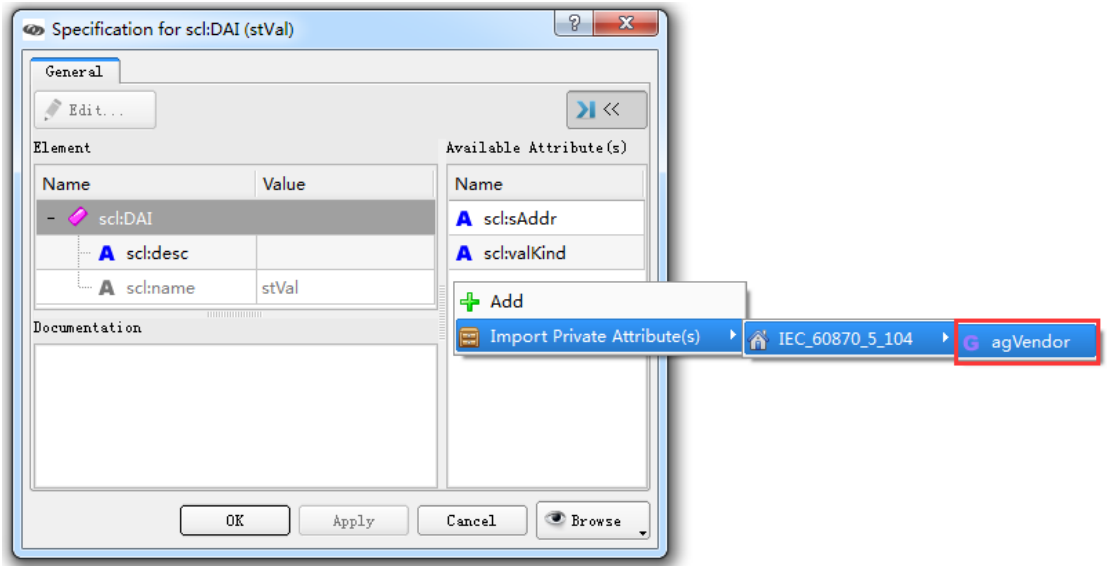


- Display “Attribute Box”

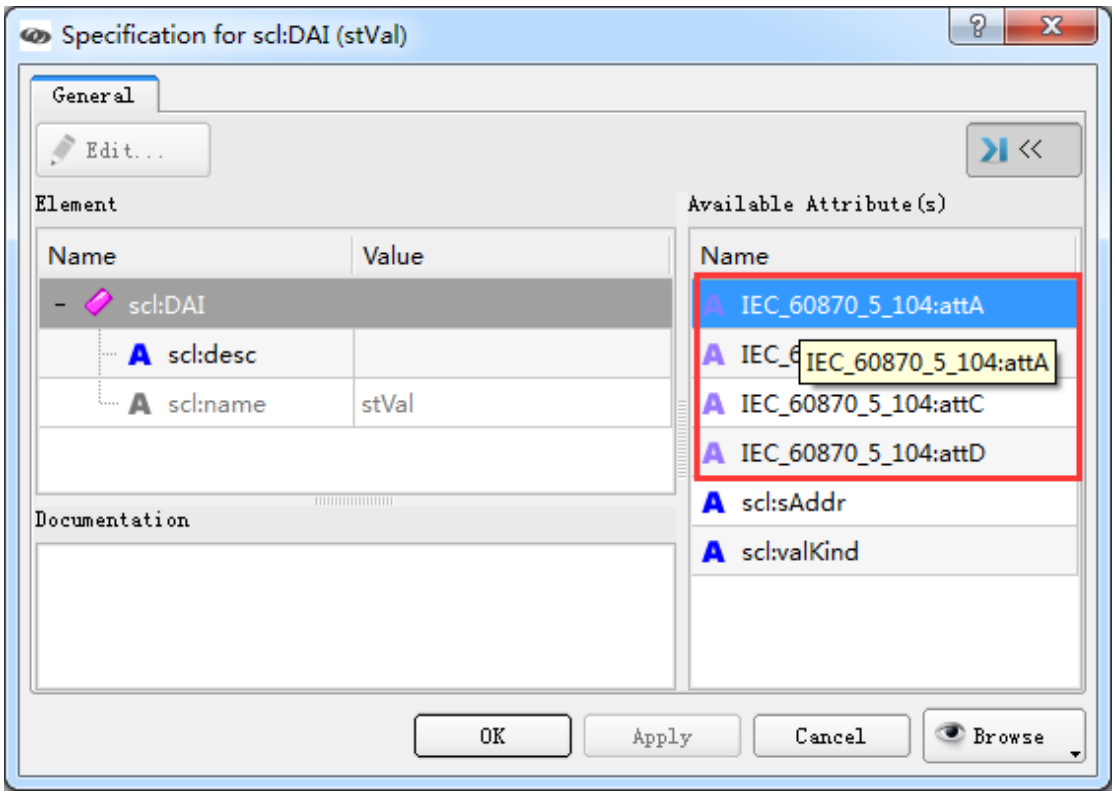


FUHUA

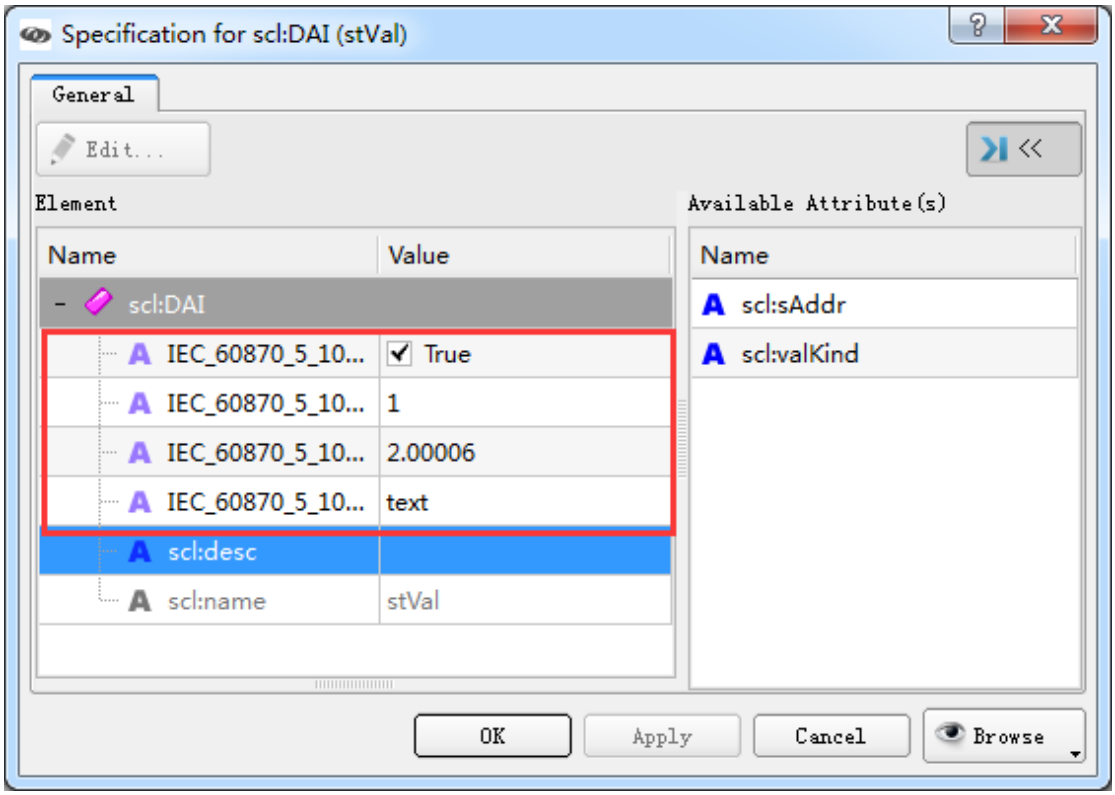
- Right click to import Private Attributes



- Add Private Attributes by Double click



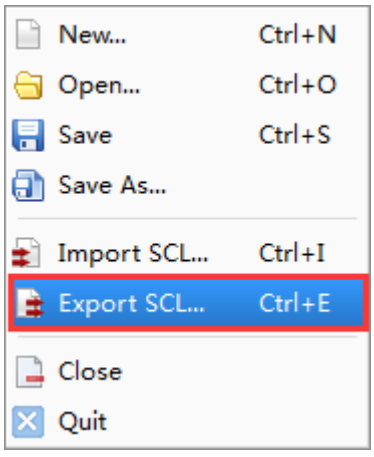
- Edit Private Attributes



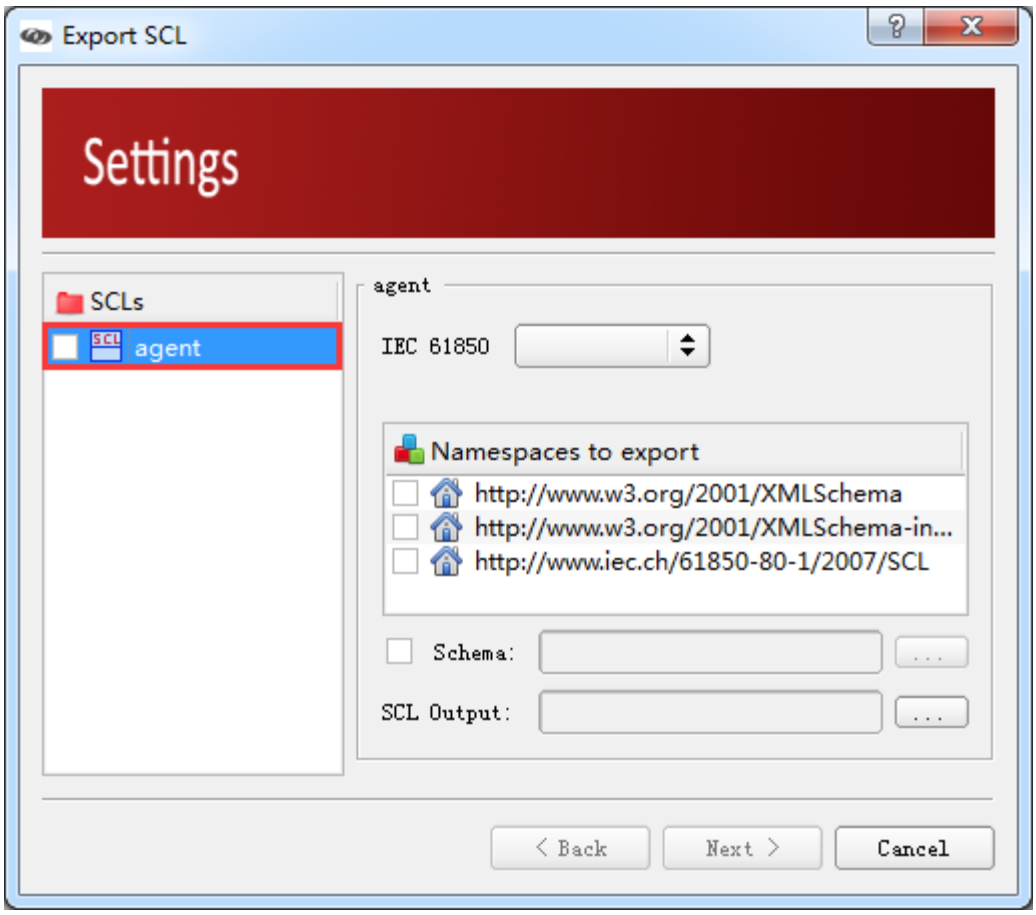
- Click OK to finish

2.2.5. Export SCL Model with Private Namespaces

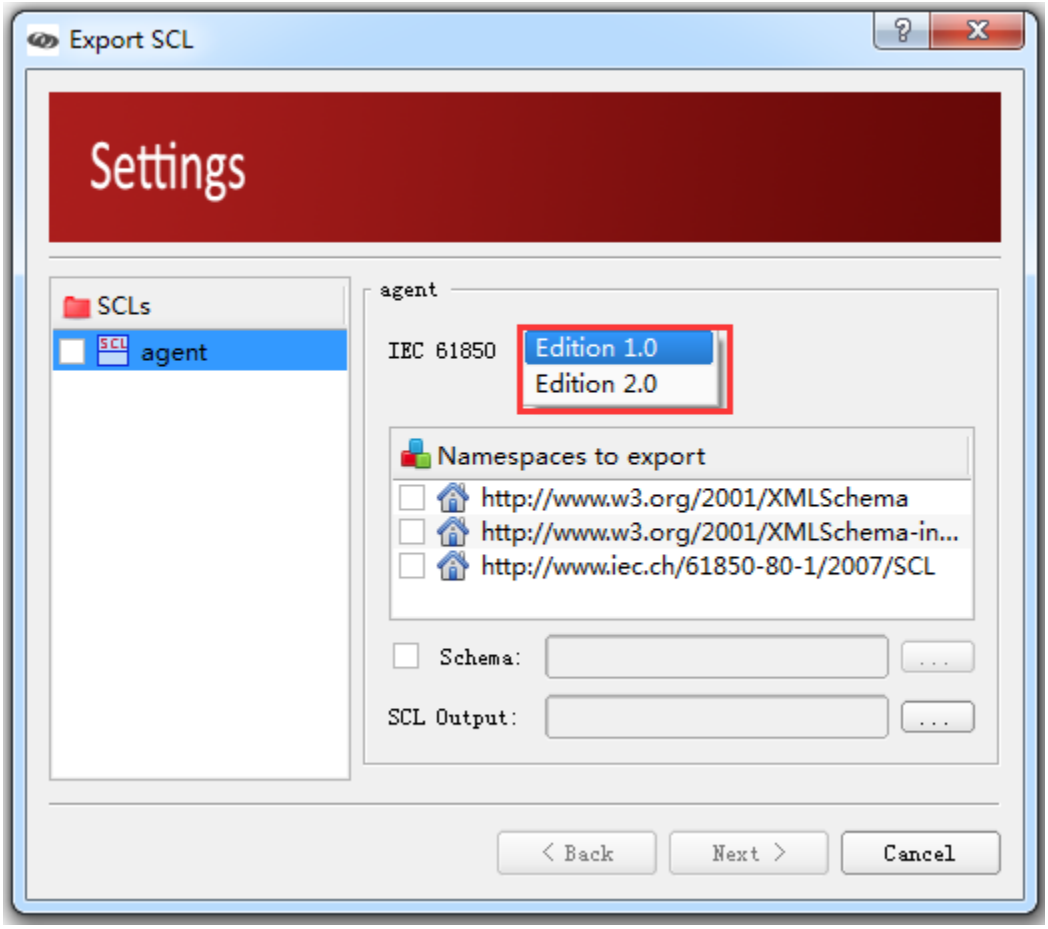
- Select **File->Export SCL…**



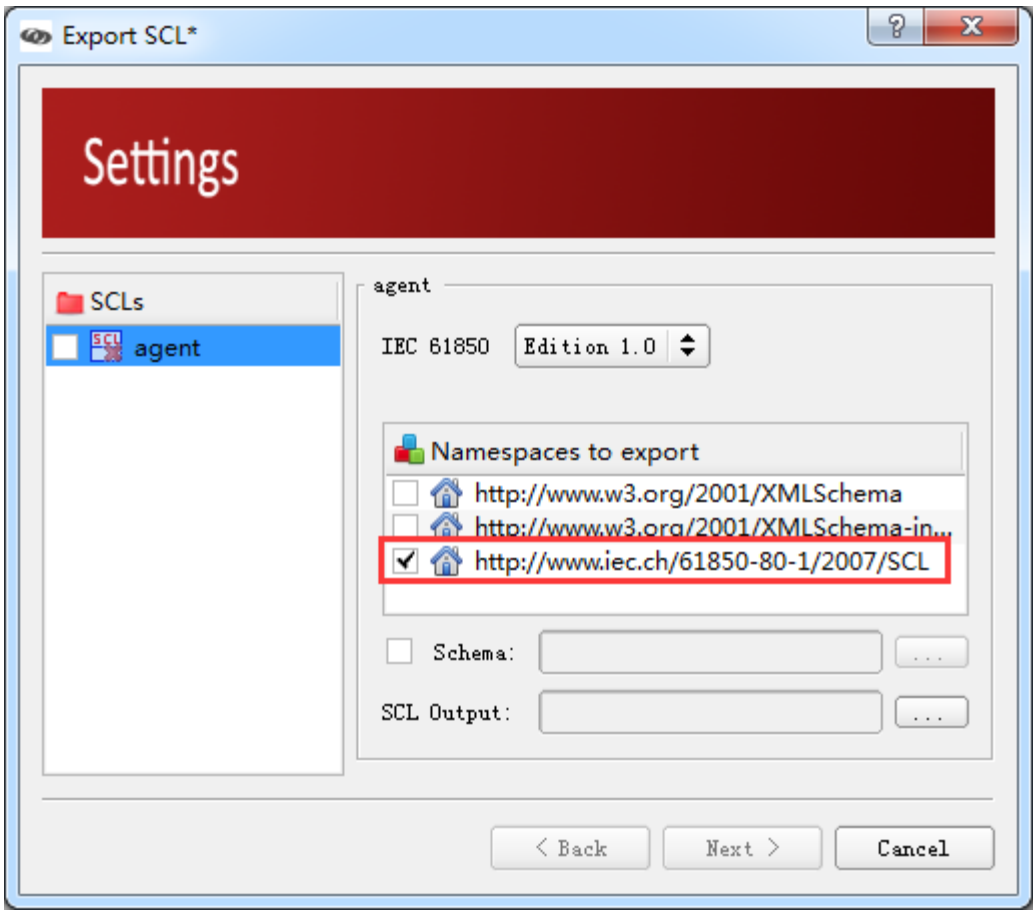
- Select the SCL to export



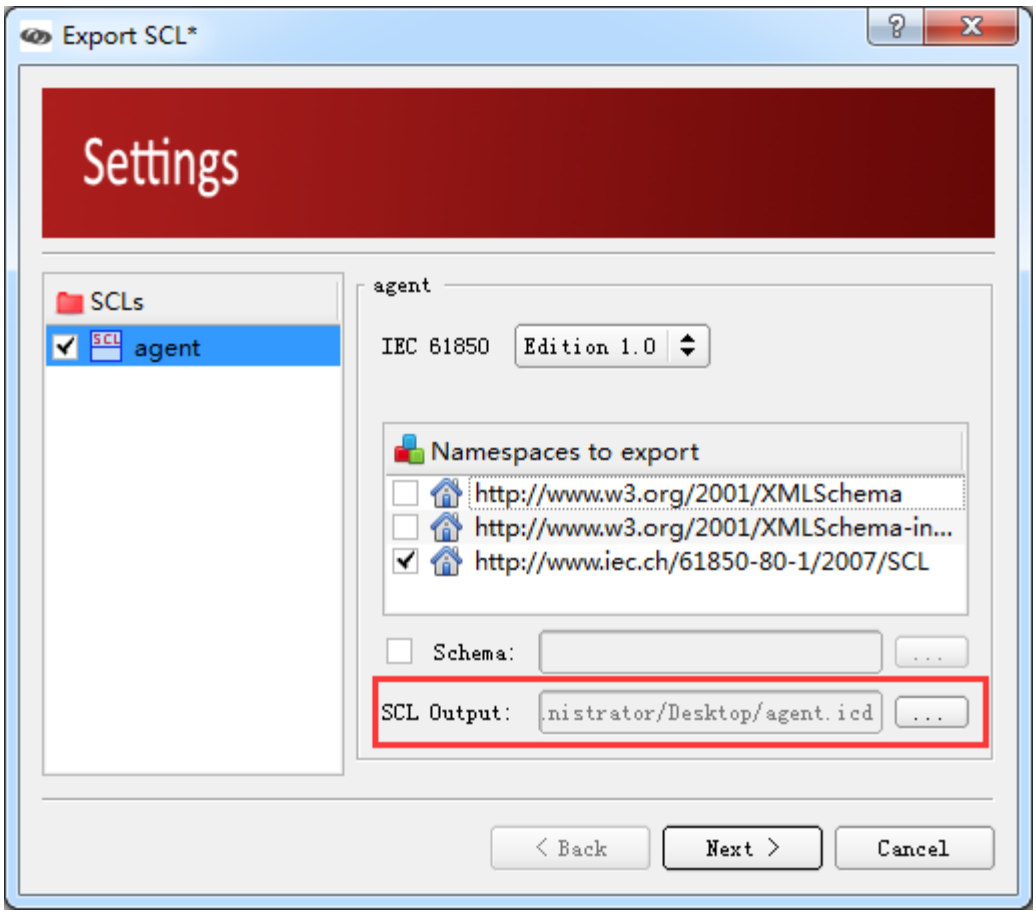
- Select the Edition option



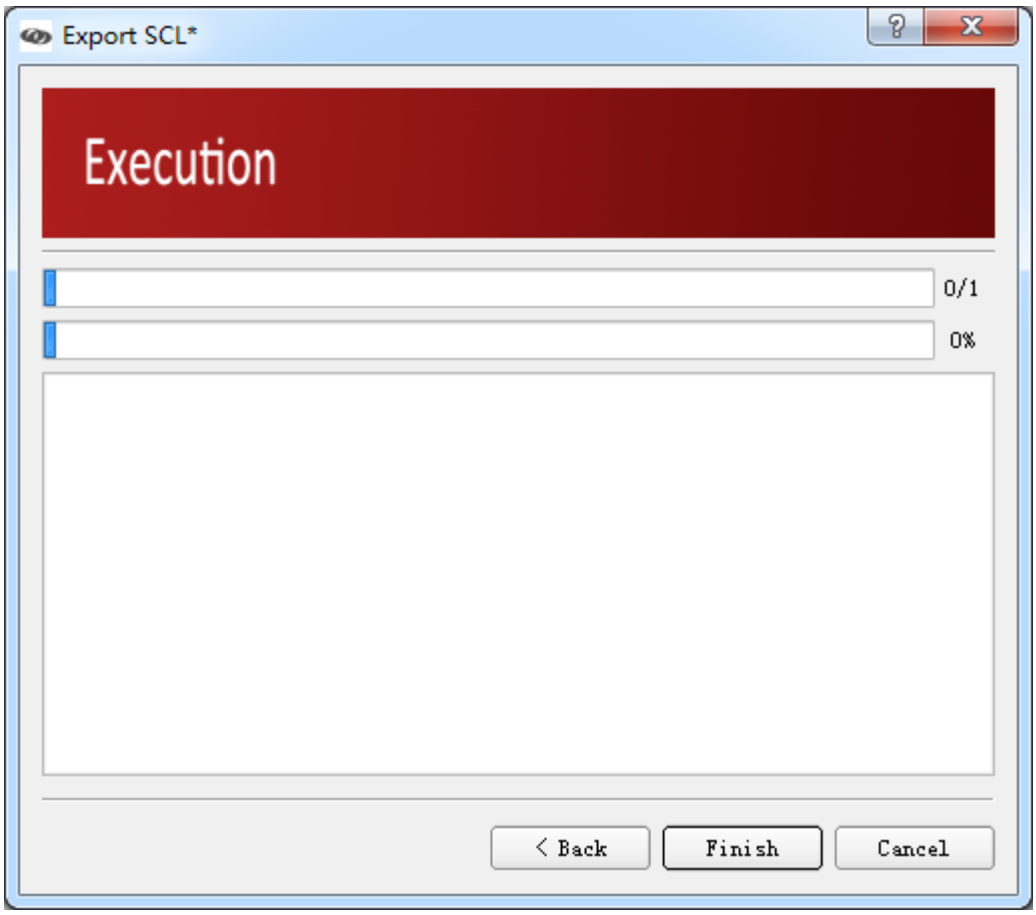
- Check the Private Namespaces to export



- Type in the output SCL file name


Export SCL*
Settings
SCLs
agent
agent
IEC 61850
Edition 1.0
Namespaces to export
http://www.w3.org/2001/XMLSchema
http://www.w3.org/2001/XMLSchema-in...
http://www.iec.ch/61850-80-1/2007/SCL
Schema:
SCL Output:
nistrator/Desktop/agent.icd
...
< Back
Next >
Cancel

- Click Next

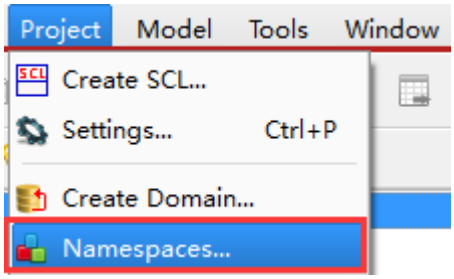


- Click Finish
- Overview of the Private Namespaces XML data

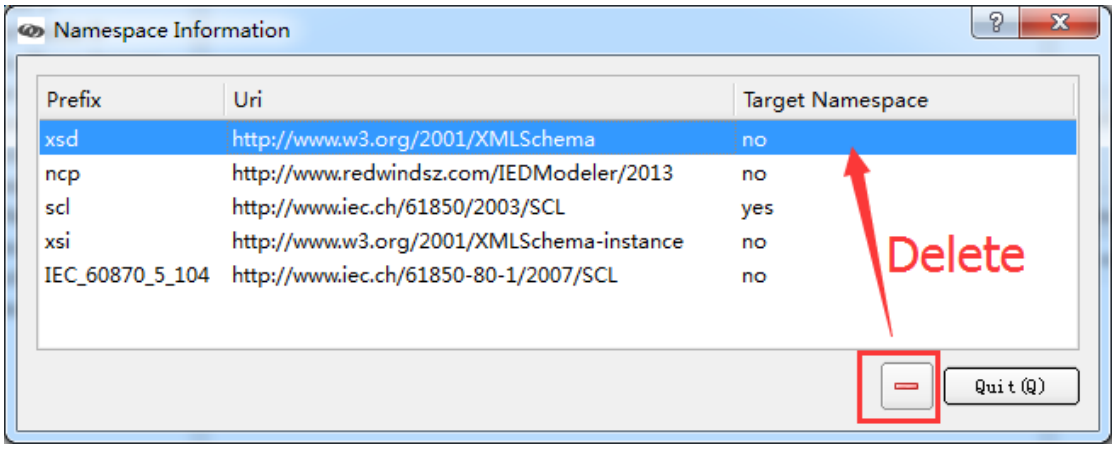
```
    </DAI>
    <DAI desc="" name="t"/>
  </DOI>
- <DOI desc="" name="AutoRecSt">
    <DAI desc="" name="q"/>
-   <DAI IEC_60870_5_104:attA="true" IEC_60870_5_104:attB="1" IEC_60870_5_104:attC="2.00006" IEC_60870_5_104:attD="text" desc="" name="stVal">
      <IEC_60870_5_104:GlobalAddress104 IEC_60870_5_104:casdu="1" IEC_60870_5_104:ioa="2" IEC_60870_5_104:ti="3"/>
      <Val>Ready</Val>
    </DAI>
    <DAI desc="" name="t"/>
```

2.3 Project Namespaces Management

Choose **Project->Namespaces…**



Namespace Information dialog:



There are three columns in **Namespace Information**.

Prefix: Unique identifier representing an Uri

Uri: Uniform resource identifier (URI) is a namespace name

Target Namespace: Only one target namespace per project is allowed

Note:

1) Users cannot import **Private Namespaces** that are already used by project
2) Users cannot delete **Target Namespace** and preserved **Namespace** (denoted as ncp)
3) Elements from a deleted **Namespace** cannot be exported