**Enterprise Architect to Protocol Buffer Exporter User’s Guide**

Copyright 2017 - 2018 Duke Energy Corporation and Open Energy Solutions, Inc.

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License.

Change Log:

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Who** | **Comment** |
| 01/12/2018 | 0.1 | LSC | Initial Draft |
| 08/08/2018 | 1.0 | LSC | Initial Release |
| 12/12/2018 |  |  |  |

[1 Overview 4](#_Toc532374590)

[2 Installation 5](#_Toc532374591)

[2.1 Windows Installer 5](#_Toc532374592)

[2.2 Manual 8](#_Toc532374593)

[3 Running the Enterprise Architect to Protobuf Exporter Add-In 10](#_Toc532374594)

[3.1 Start Enterprise Architect 10](#_Toc532374595)

[3.2 Run the Enterprise Architect to Protobuf Exporter 12](#_Toc532374596)

[3.3 Generate Protobuf 16](#_Toc532374597)

[3.4 Errors During Generate Protobuf 17](#_Toc532374598)

[3.5 Save Protobuf 18](#_Toc532374599)

[3.6 Enterprise Architect to Protobuf Exporter Add-In About Menu 19](#_Toc532374600)

[4 Writing OpenFMB Profiles 20](#_Toc532374601)

[4.1 What is UML tagged value 20](#_Toc532374602)

[4.2 Pre-defined tags 20](#_Toc532374603)

[4.3 How to add UML tags: 21](#_Toc532374604)

[4.3.1 Package level 21](#_Toc532374605)

[4.3.2 Class & association level 22](#_Toc532374606)

[4.3.3 Steps to add a tagged value for Generalization: 23](#_Toc532374607)

[4.3.4 Steps to add a tagged value for attribute: 23](#_Toc532374608)

[4.3.5 Steps to add a tagged value for association: 24](#_Toc532374609)

[4.3.6 Top Package Level 24](#_Toc532374610)

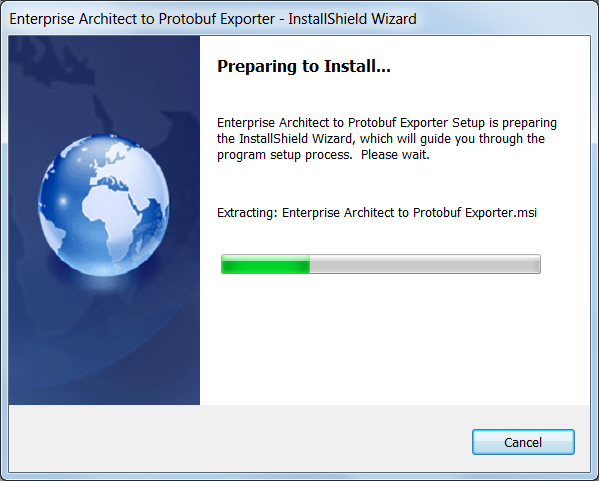
# Overview

The Enterprise Architect to Protocol Buffer (Protobuf) Exporter utilizes the Enterprise Architect Add-In facility to provide a new menu that allows users to convert an OpenFMB UML model into Protocol Buffer definition files.

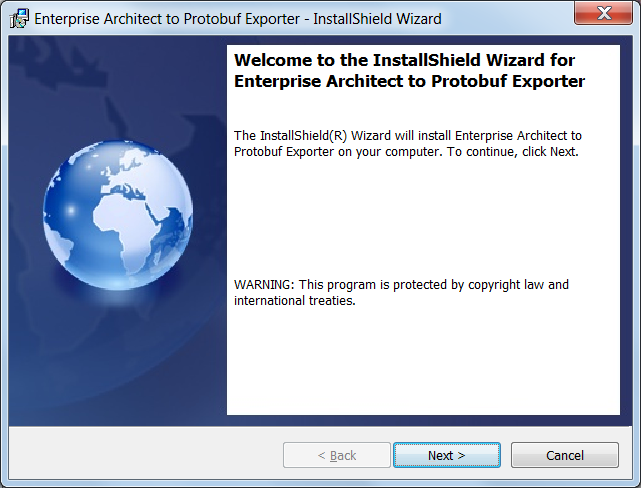
# Installation

## Windows Installer

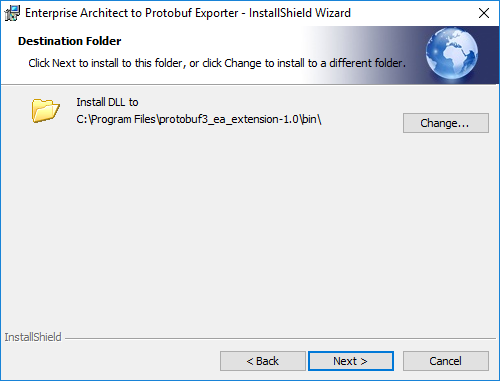
1. Run “protobuf3\_ea\_extension-installer-1.1.exe” as Administrator to bring up “InstallShield Pro 2018”



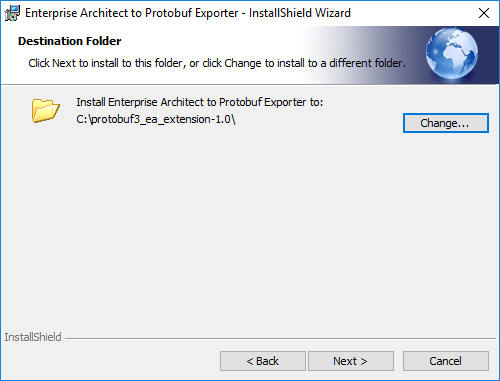
2) Press Next to continue



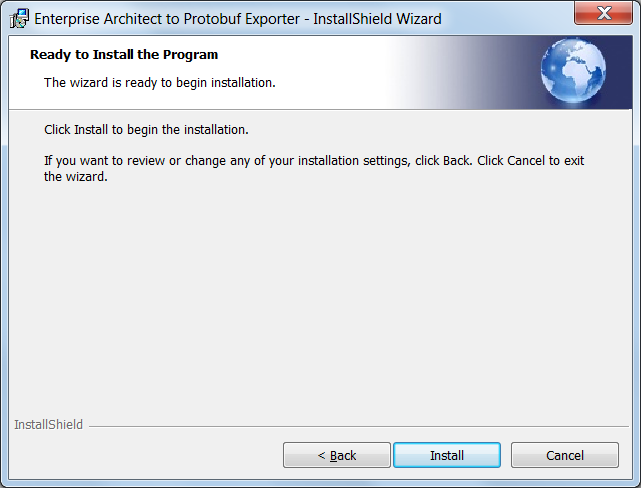
1. Accept the default installation directory or select another directory in which to install the DLL



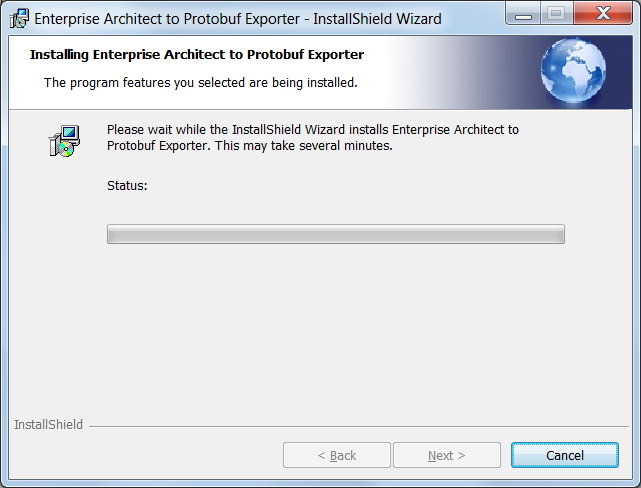
1. Accept the default installation directory or select another directory in which to install the source code



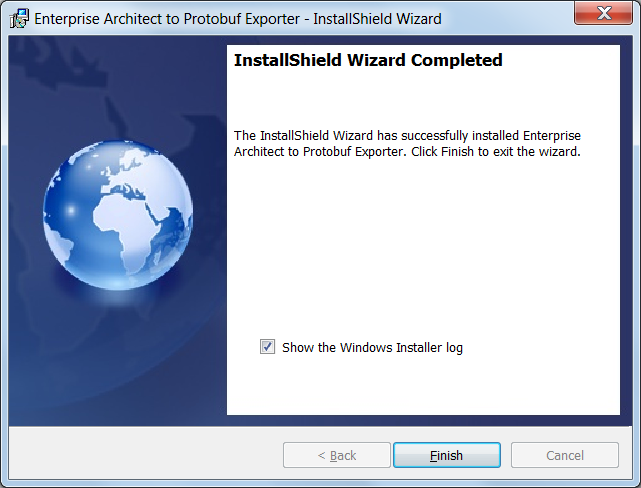
1. Press Install to begin installation



5) Wait for the installation to complete



6) Press Finish

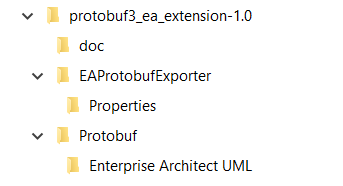


If you accept the default directories during installation you will see the following directory structures

* The DLL will be installed under C:\Program Files



* The source code and documentation will be installed under C:\



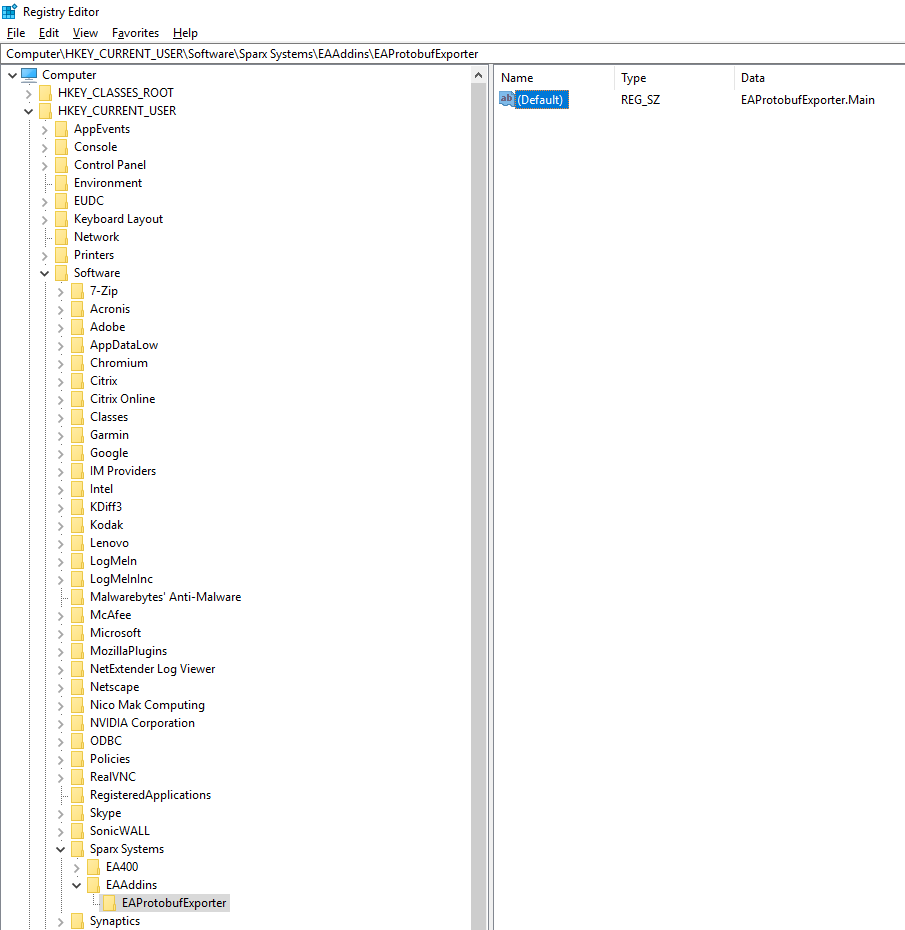
## Manual

To manually install the Enterprise Architect to Protobuf Exporter follow these steps:

1. Copy the EAProtobufExporter.dll to the desired directory (i.e. “*C:\Program Files\OpenFMB\bin*”)
2. Register the DLL by running the Assembly Registration Tool

C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe "C:\Program Files\OpenFMB\bin\EAProtobufExporter.dll" /codebase

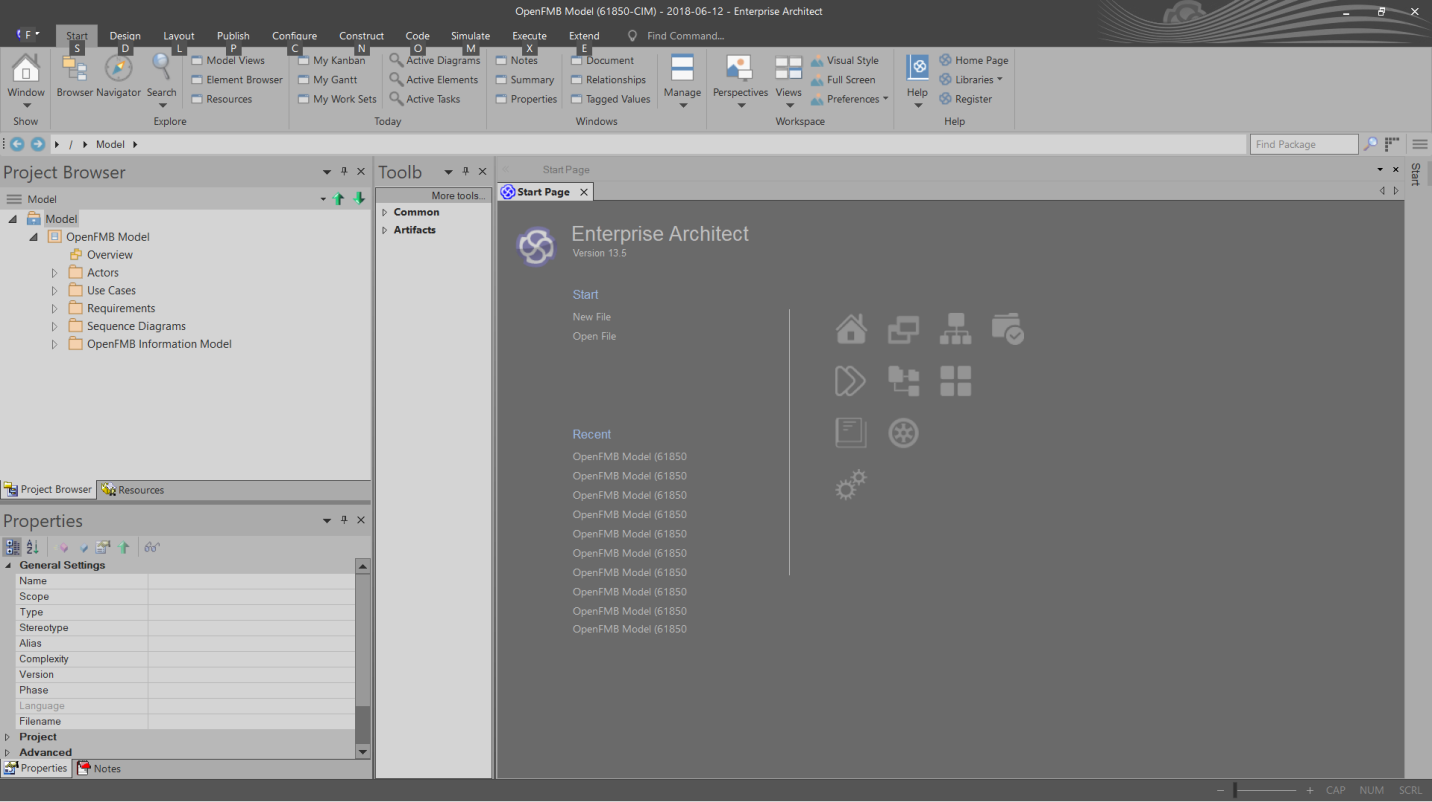
1. Create a new entry in the registry by running regedit. This will allow Enterprise Architect to recognize the presence of the Enterprise Architect to Protobuf Exporter Add-In. Add the new key value “*EAAddIns*” under the appropriate location:
   1. For single users: “*HKEY\_CURRENT\_USER\Software\Sparx Systems*”
   2. For multiple users: “*HKEY\_LOCAL\_MACHINE\Software\Sparx Systems*”
2. Under the “*EAAddIns*” key, add a new key value using the project name “*EAProtobufExporter*” of the Enterprise Architect to Protobuf Exporter Add-In
3. Under the “*EAProtobufExporter*” key, modify the default value by entering the <project name>.<class name> of the Enterprise Architect to Protobuf Exporter Add-In “*EAProtobufExporter.Main*”

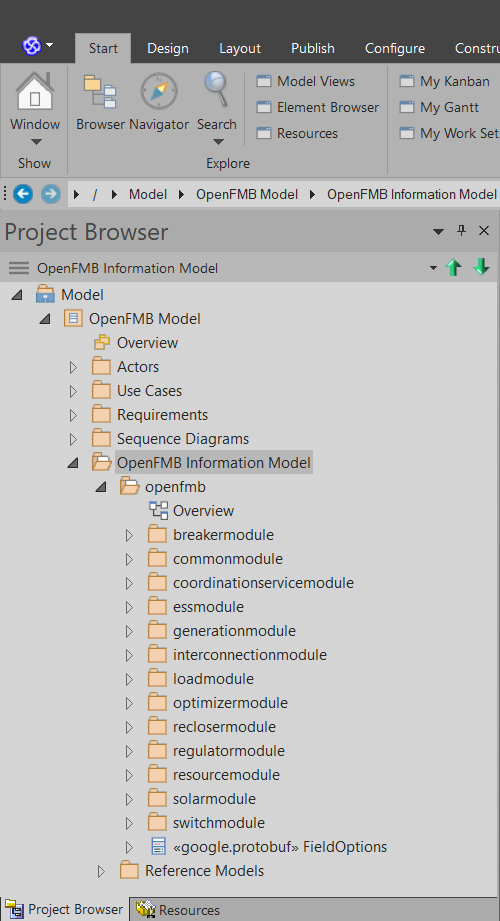


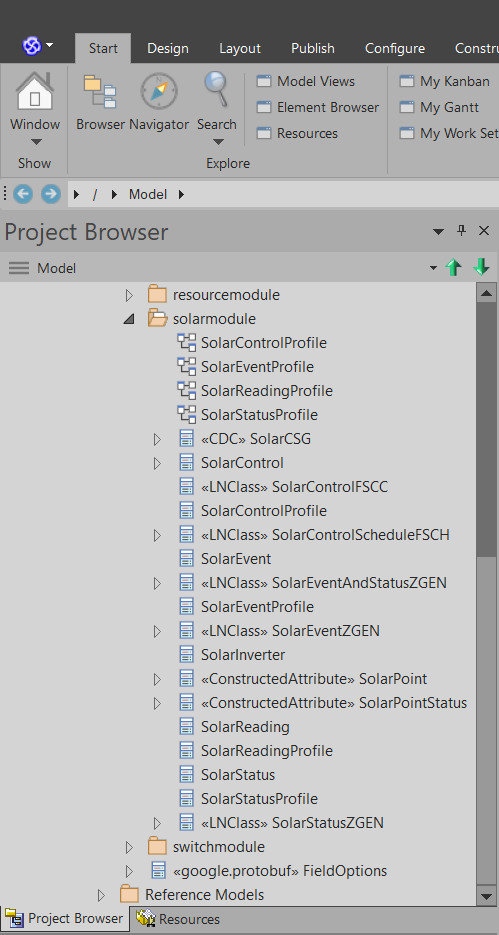
# Running the Enterprise Architect to Protobuf Exporter Add-In

## Start Enterprise Architect

Start Enterprise Architect and load the appropriate OpenFMB model.

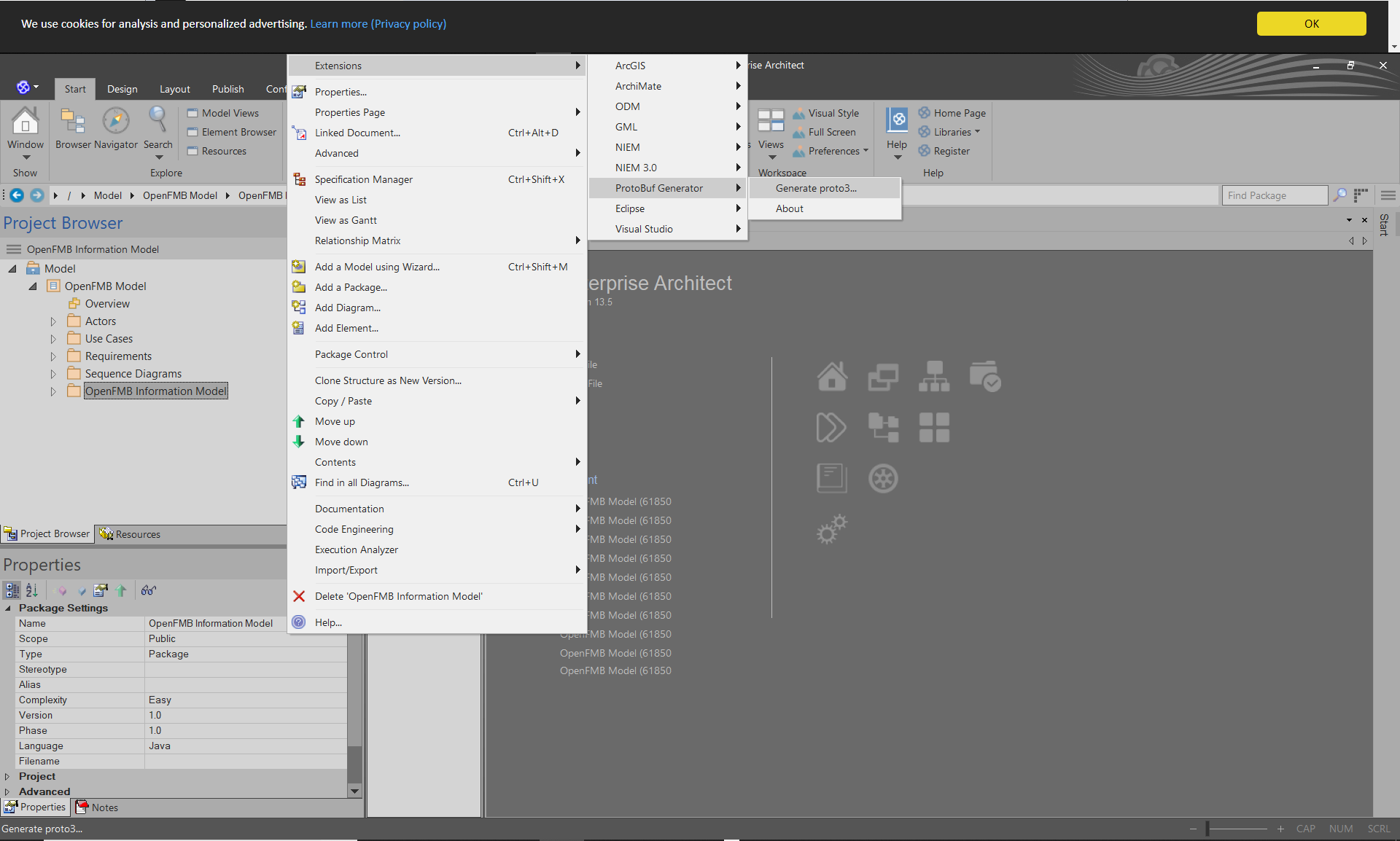




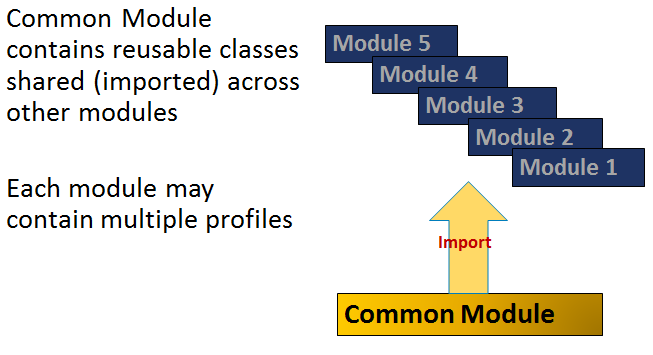


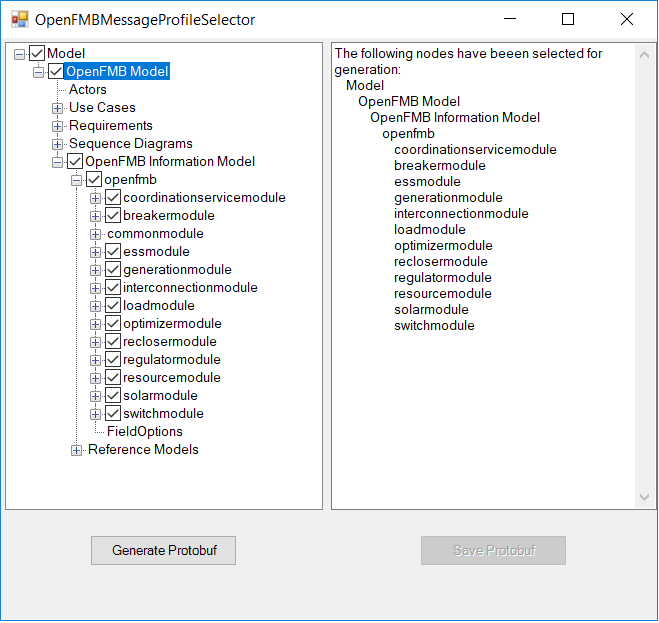
## Run the Enterprise Architect to Protobuf Exporter

Right click in the “*Project Browser*” window and highlight the “*Extensions*” menu to show a list of the available extensions or add-ins. Highlight the “*Protobuf Generator*” and select “*Generator proto3…*”.

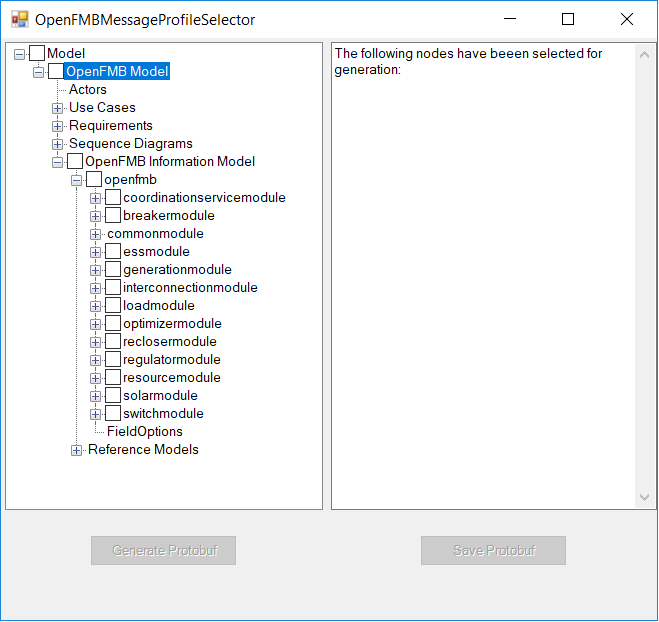


The Enterprise Architect to Protocol Buffer (Protobuf) Exporter will build a Tree View containing the model information with check boxes next to the Tree Nodes that are specific to the generation of the Protobuf files in the left pane and processing information in the right pane. Since it is used by the other modules, the Common is always checked.

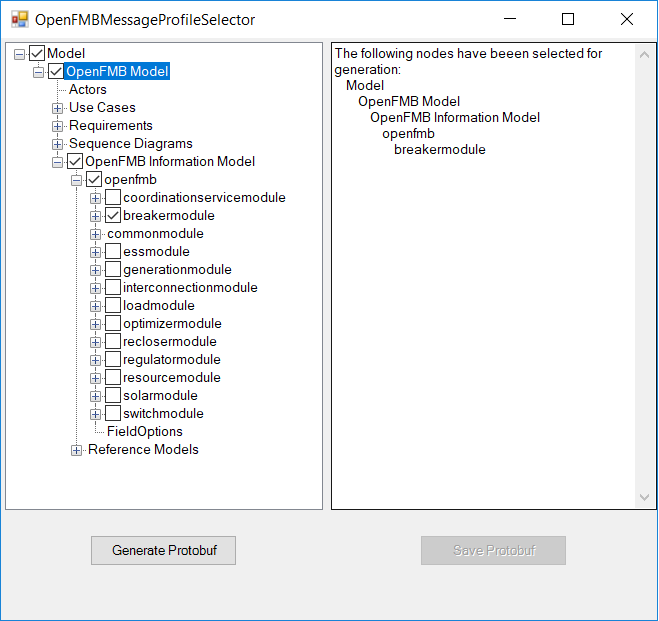




If all the check boxes are unchecked the right pane will update to indicate that no nodes are selected.



Reselecting nodes will result in the right pane updating to reflect the nodes that are selected.



## Generate Protobuf

Once the nodes have been selected for generation, click the “*Generate Protobuf*” button to initiate the generation of the Protobuf information.

Note: At this point the “*Generate Protobuf*” button will be grayed out until a node is reselected in the Tree View (left pane).



During the generation of the Protobuf information, any processing *Information* messages and *Error* messages will be displayed in the right pane.

## Errors During Generate Protobuf

If an error is encountered during processing, the following will be displayed after all error messages:

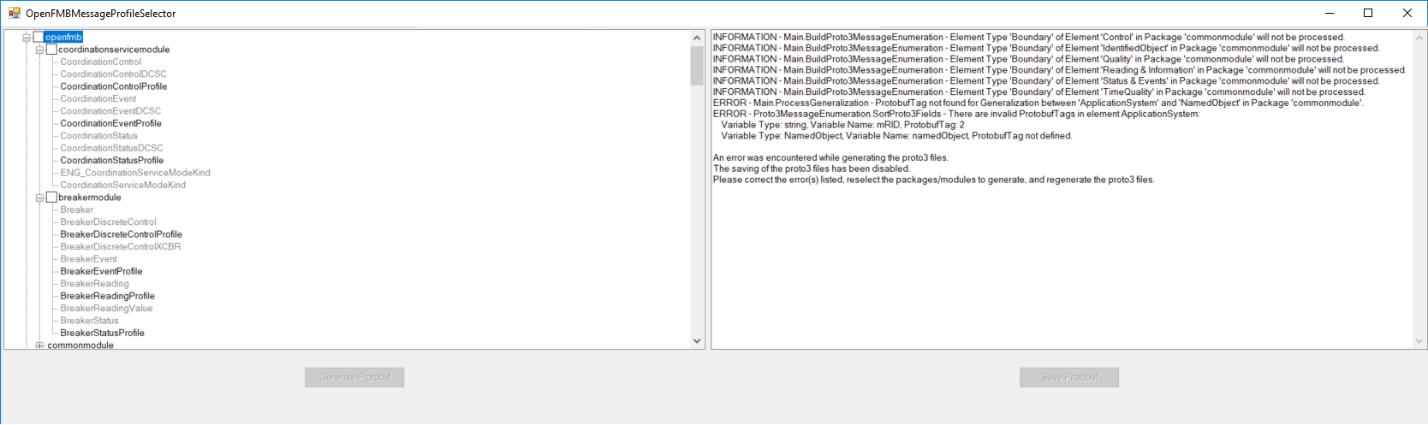
*An error was encountered while generating the proto3 files.*

*The saving of the proto3 files has not been disabled.*

*Please correct the error(s) listed and regenerate the proto3 files.*

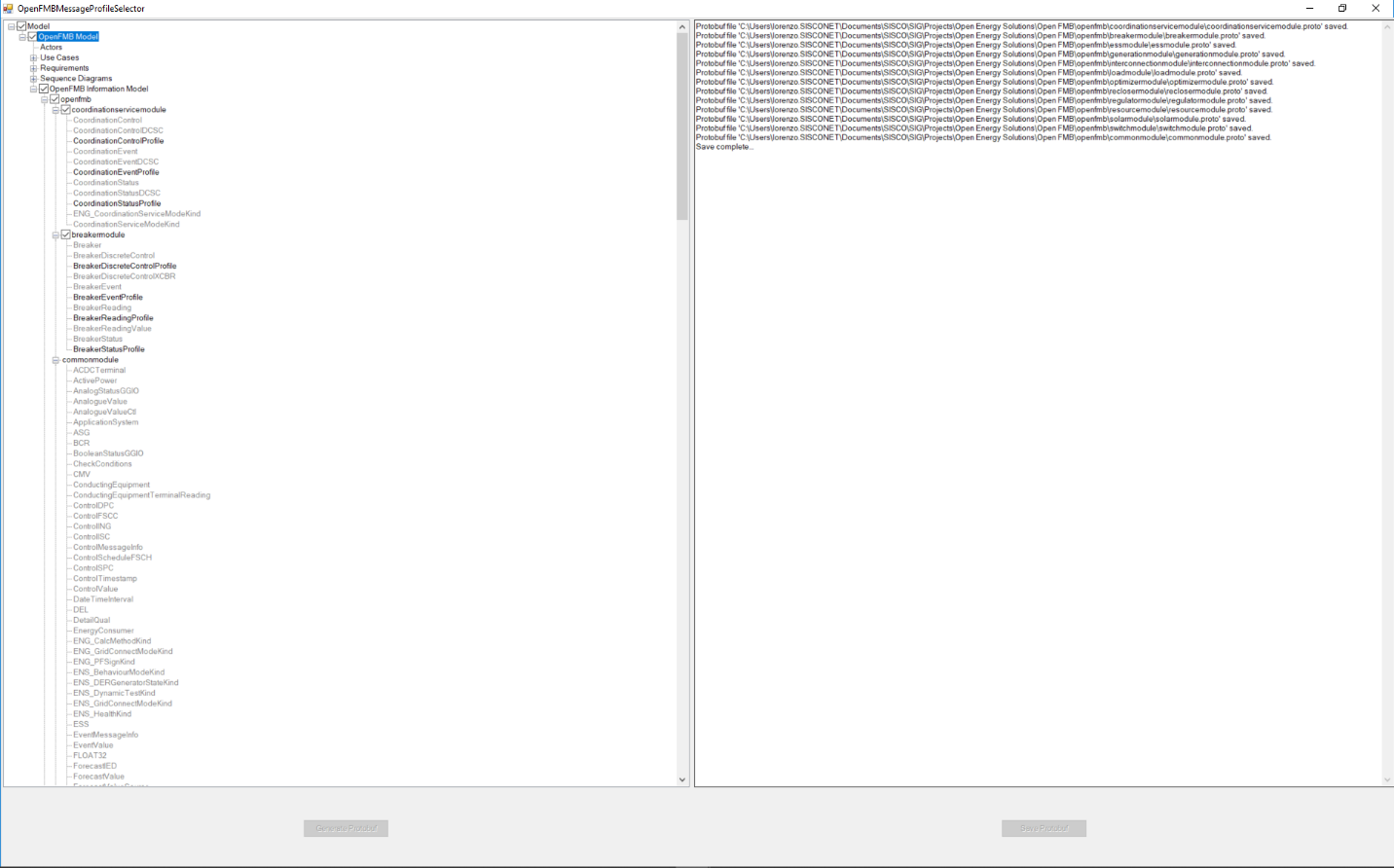
At this point the user has the following options:

* Reselect the nodes that do not have errors and generate the Protobuf information. This option will not generate a full set of Protobuf files.
* Close the Tree View window, fix the errors, and regenerate the Protobuf information

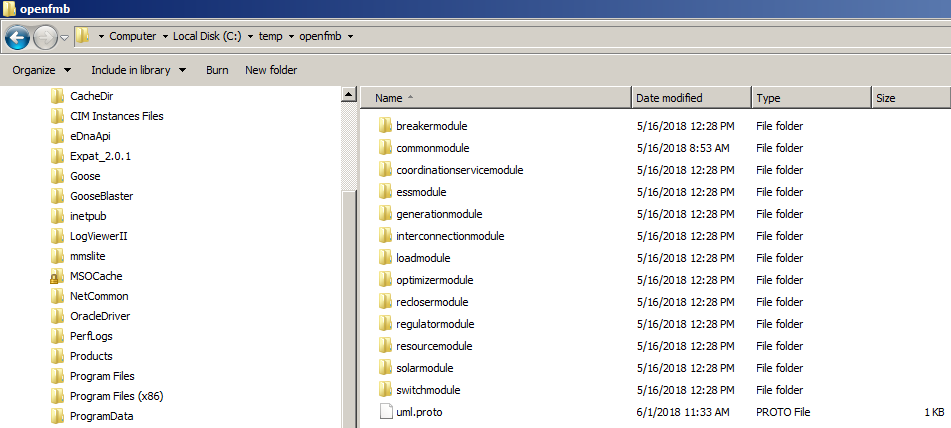


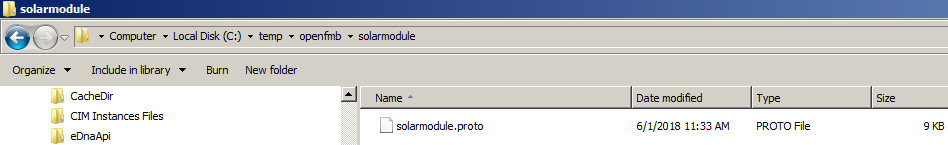
## Save Protobuf

When the generation process completes with no errors, the “*Save Protobuf*” button will be enabled. Clicking the “*Save Protobuf*” button will bring up a “*Browse For Folder*” window to allow the user to browse to the directory where the Protobuf files will be saved. When the save process completes, the right pane will display the Protobuf files saved with their absolute paths and the message “*Save complete…”*



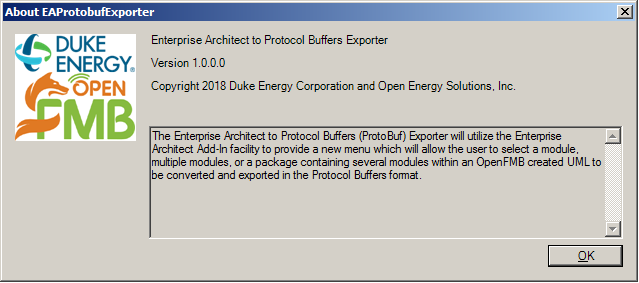
The save process will create the Protobuf directory structure if it does not exist. If the directory structure does exist, then the save process will overwrite the existing files. The Protobuf directory structure under the user selected directory is openfmb/<*Protobuf Module Name*>.





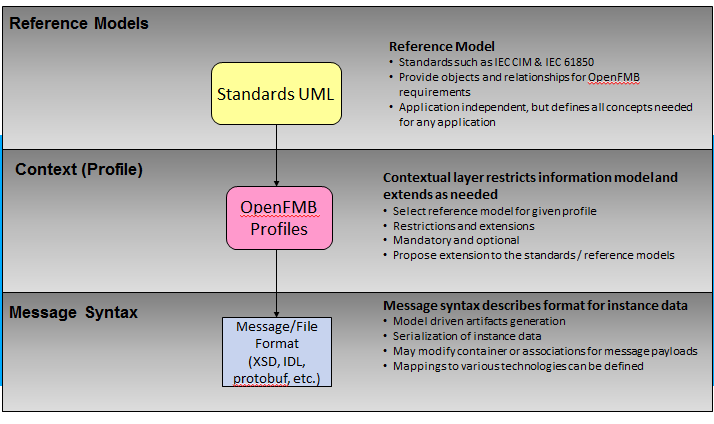
## Enterprise Architect to Protobuf Exporter Add-In About Menu

Right click in the “*Project Browser*” window and highlight the “*Extensions*” menu to show a list of the available extensions or add-ins. Highlight the “*Protobuf Generator*” and select “*About*”.



# Writing OpenFMB Profiles

OpenFMB UML is a Platform Independent Model information model, which is converted to a Platform Specific Model message syntax for implementation, in this case Protocol Buffers.



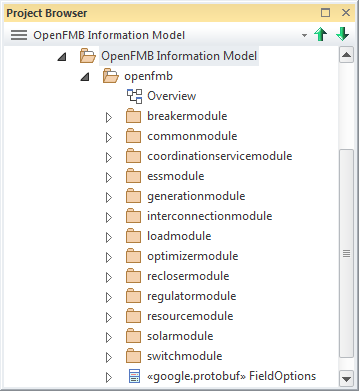
Protocol Buffers require additional metadata beyond that available in standard UML models. To express this information OpenFMB has defined specific UML Tagged Values. In general, the UML tagged values ([Sparx Enterprise Architect users guide on tagged values](http://www.sparxsystems.com/enterprise_architect_user_guide/12/modeling_basics/thetaggedvaluestab.html)) are used to extend an existing UML model for a specific purpose. In this case, the UML tagged values are designed for the Protobuf file generation.

## What is UML tagged value

An UML tagged value is basically a property/value pair. For instance, a pre-defined “ProtobufTag” as a property can have a sequential number as its value such as 1, 2, and 3. [Sparx Enterprise Architect users guide on tagged values](http://www.sparxsystems.com/enterprise_architect_user_guide/12/modeling_basics/thetaggedvaluestab.html) has more information.

## Pre-defined tags

There are two groups of tags defined, one for UML package and one for individual UML class & association. Here is a high-level package structure in the OpenFMB model. Each sub package under the top “openfmb” package contains protocol buffer related tags.



## How to add UML tags:

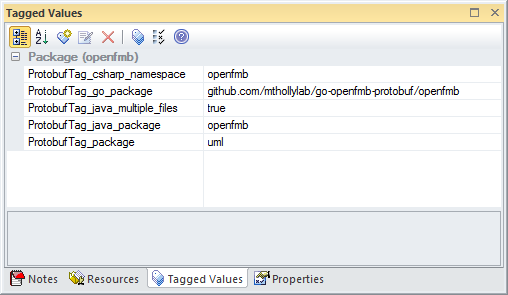
As mentioned, the Protobuf related UML tagged values are used for UML package and individual UML model element such as class and association.

### Package level

The five tags listed in the image below are used for each package listed under the top “openfmb” package. Note the top package is also included. These package level tags provide guidance for where build tools place the different language files generated for that package by the Protobuf compiler (protoc).

Here are the steps to add these tagged values to a UML package:

1. Bring up a package’s “Tagged Values” window (if not listed as default in Project Browser, double click a package and then click on the “Tags” tab at the lower right corner).
2. Select the Package (…) row in the “Tagged Values” window and click the “add tagged value” symbol (circled in red). Naming convention are also applied such as the “ProtobufTag\_java\_package=openfmb.<subpackage\_name>”. The ones listed below is for the top package, hence no sub package name as part of the tagged value.



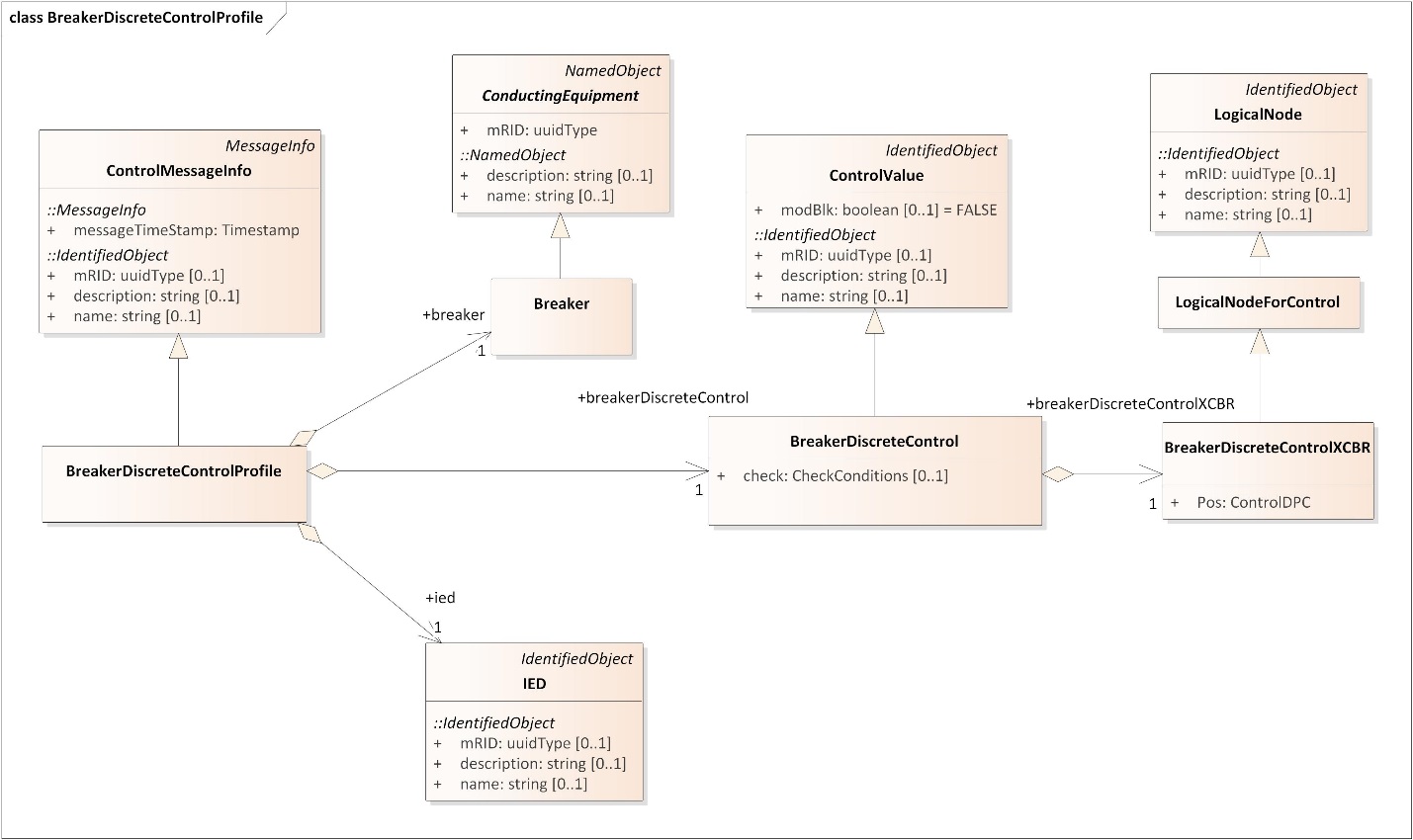
* ProtobufTag\_csharp\_namespace – Specifies the namespace during Java code generation
* ProtobufTag\_go\_package – Specifies the package name during Go code generation
* ProtobufTag\_java\_multiple\_files – Causes top-level messages, enums, and services to be defined at the package level, rather than inside an outer class named after the .proto file
* ProtobufTag\_java\_package – Specifies the package name during Java code generation
* ProtobufTag\_package – Specifies the package name for the protobuf message types

### Class & association level

Four Protocol tagged values can be potentially applied to a class but in most cases, the “ProtobufTag” is the one in use. This tag specifies the sequence number of associations and attributes with respect to a class. The order to assign sequence number follows the list below:

1. Generalization (if exist)
2. Attribute (if exist)
3. Association (if exist)

Here is one example illustrated in the image below. The four ProtobufTag in red are the tagged values applied with respect to the BreakerDiscreteControlProfile class. Note all four tags are applied to the associations that BreakerDiscreteControlProfile is the source class. Now pick another class, BreakedDiscreteControl, note that this class does not only contain associations but also an attribute “check”. As a result, the ProtobufTag=1 (colored in blue) is assigned to its generalization association, the ProtobufTag=2 is assigned to its attribute, “check”, and the ProtobufTag=3 is assigned to its association with BreakerDiscreteControlXCBR.



ProtobufTag=3

ProtobufTag=2

ProtobufTag=1

ProtobufTag=4

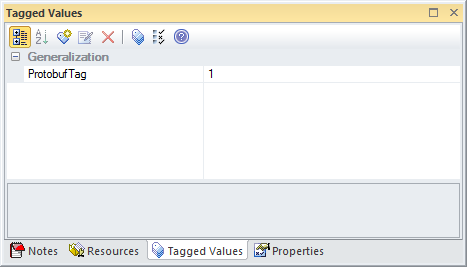
ProtobufTag=3

ProtobufTag=2

ProtobufTag=1

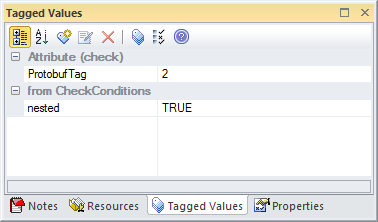
### Steps to add a tagged value for Generalization:

1. Select a generalization and bring up its “Tagged Values” window
2. Select Generalization row and add ProtobufTag with corresponding value



### Steps to add a tagged value for attribute:

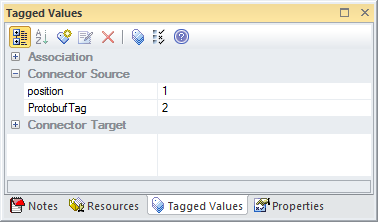
1. Select an attribute and bring up its “Tagged Values” window
2. Select Attribute row and add ProtobufTag with corresponding value



*Note the “nested” tagged value can be ignored in this context. It is used for IDL generation.*

### Steps to add a tagged value for association:

1. Select an association and bring up its “Tagged Values” window
2. Select Connector Source row and add ProtobufTag with corresponding value



*Note the “position” tagged value can be ignored in this context. It is used to label an element sequence position for XSD generation.*

The ProtobufTag is mostly in use. Other specialized tags are:

* ProtobufTag\_extend – Specifies the class as containing Protobuf extensions (on FieldOptions only)
* ProtobufTag\_UUID – Specifies the attribute of the class as type UUID
* ProtobufTag\_Key – Specifies the attribute of the class as key

### Top Package Level

Also there is a top package level class for field level options for Protobuf file generation. This metadata is propagated to the Protobuf using applications in uml.proto

