OpenFMB Protobuf Generator User's Guide

Change Log:

Date	Version	Who	Comment
01/12/2018	0.1	LSC	Initial Draft
08/08/2018	1.0	LSC	Initial Release
04/18/2019	1.0		Installer screen images and directory locations
08/26/2021	2.0	CN	Update app name and about screenshot

1 Overview

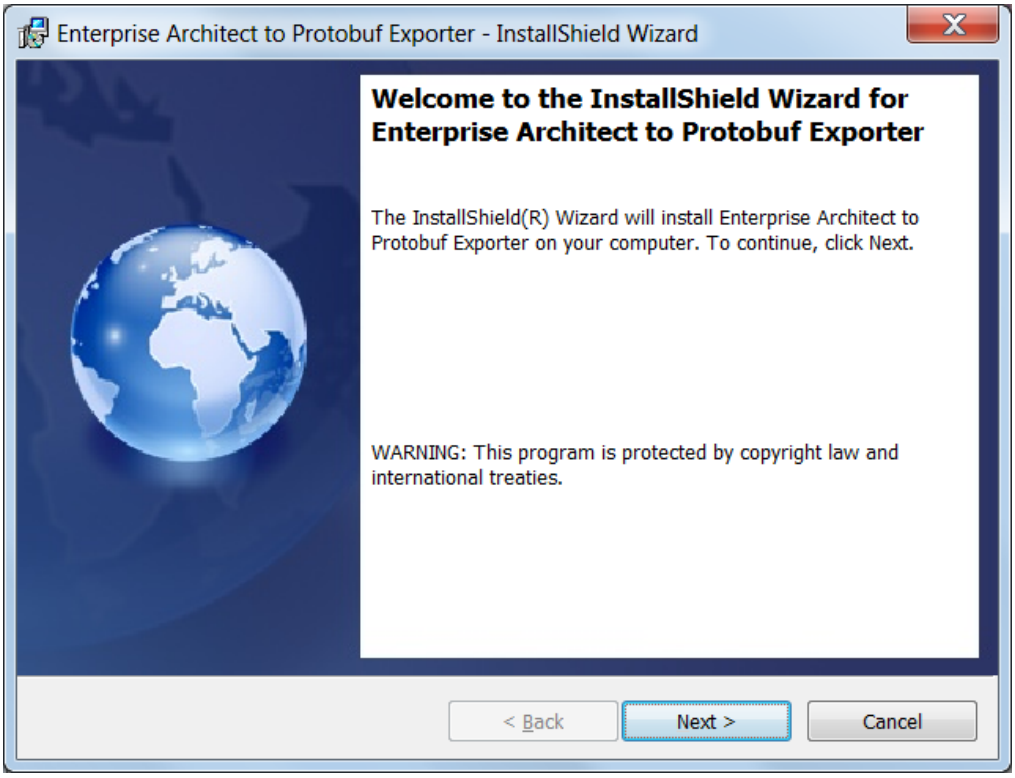
The OpenFMB Protobuf Generator utilizes the Enterprise Architect Add-In facility to provide a menu that allows users to convert an OpenFMB UML model into Protocol Buffer definition files.

2 Installation

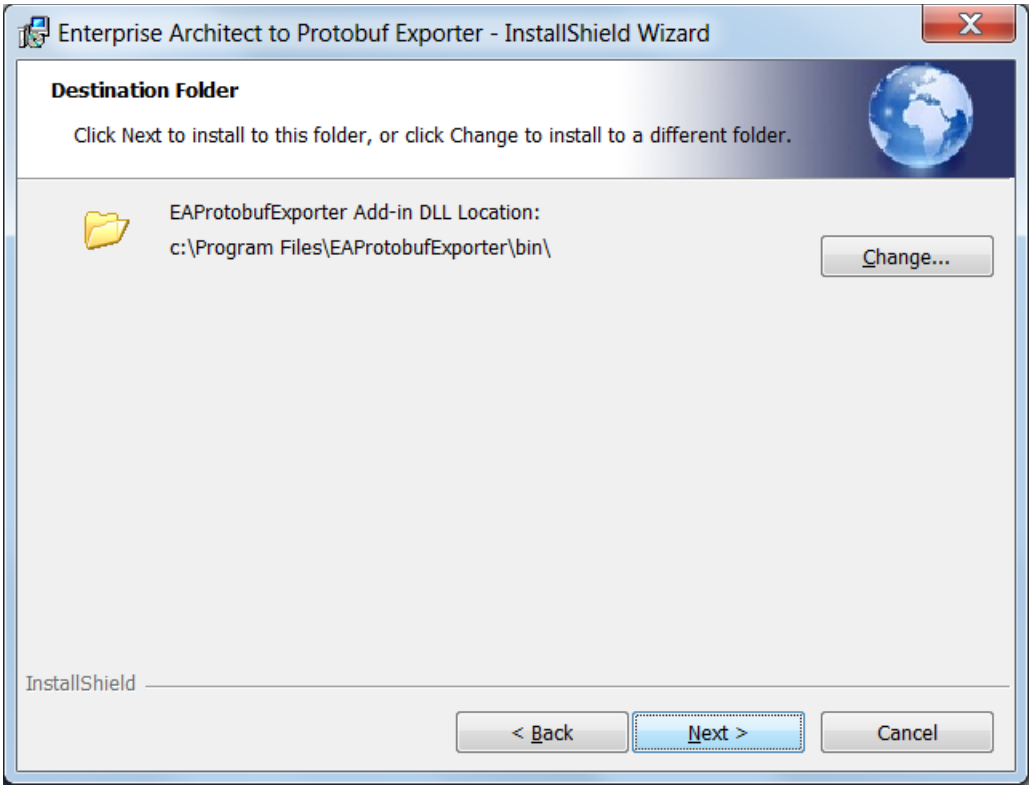
2.1 Windows Installer

1) Run "EAProtobufExporter-installer-1.0.exe" with Administrator permissions to start the installer.

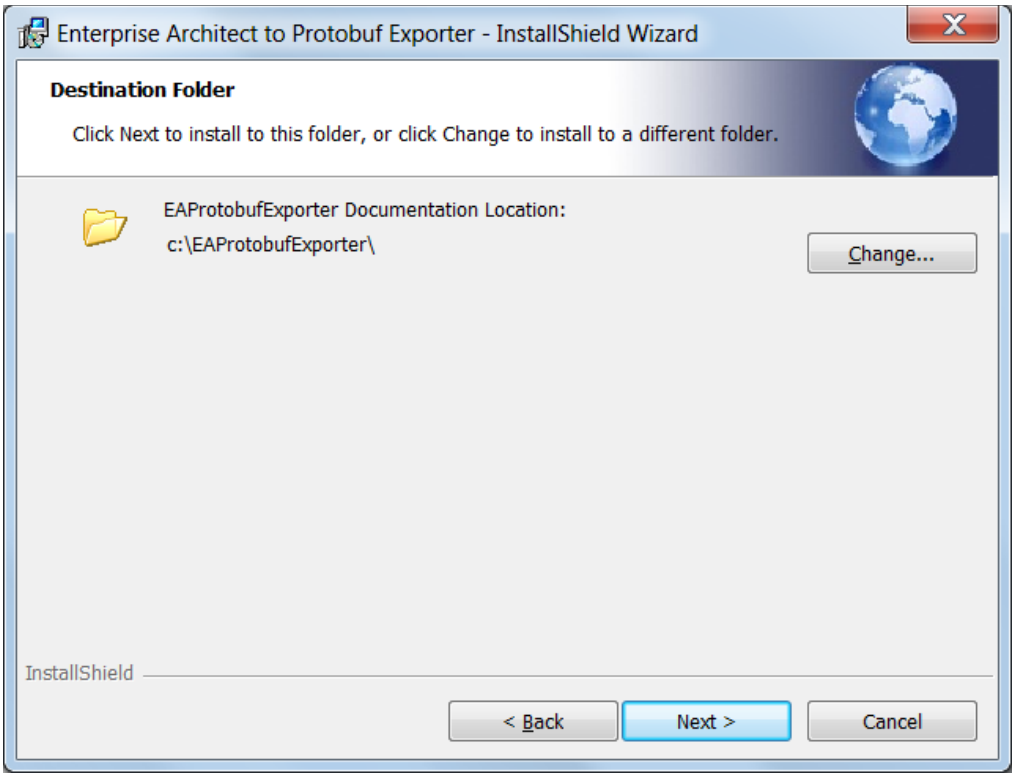
2) Press Next to continue



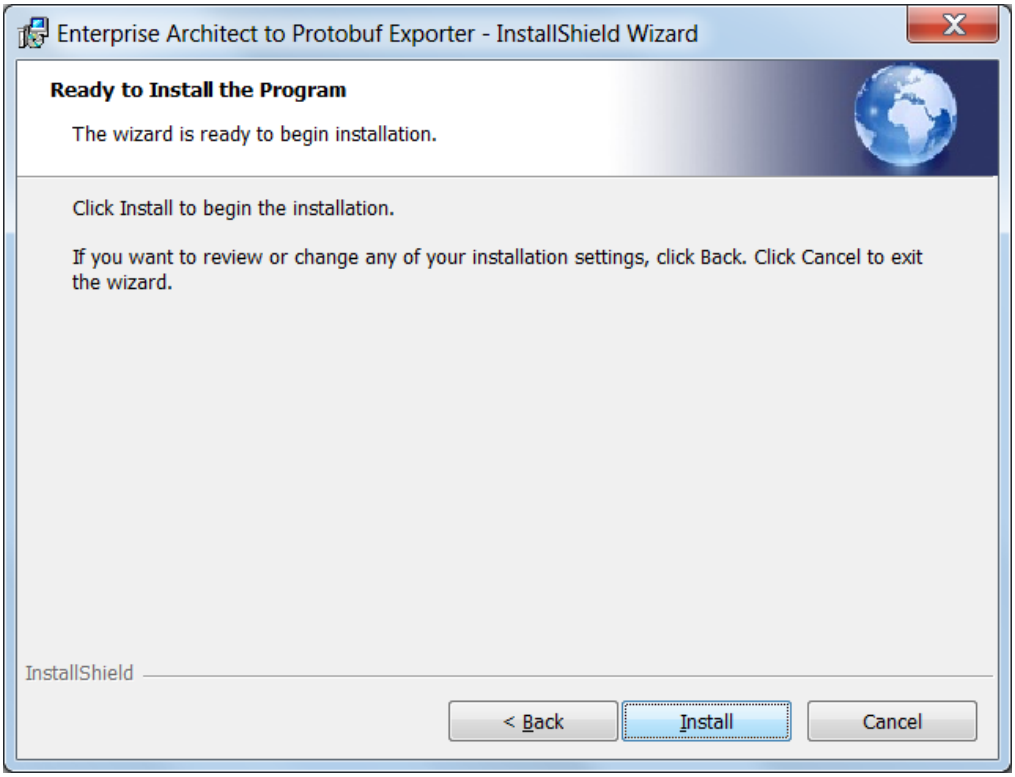
3) Accept the default program installation directory or select another directory in which to install the DLL



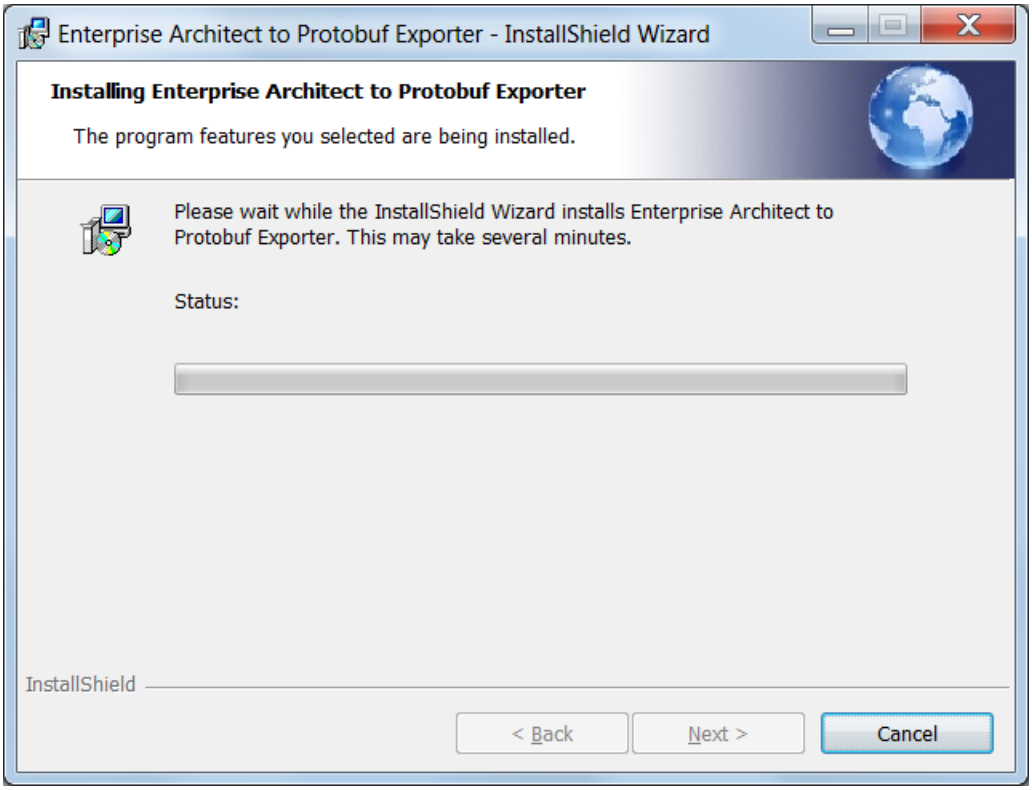
4) Accept the default installation directory or select another directory in which to install the documentation



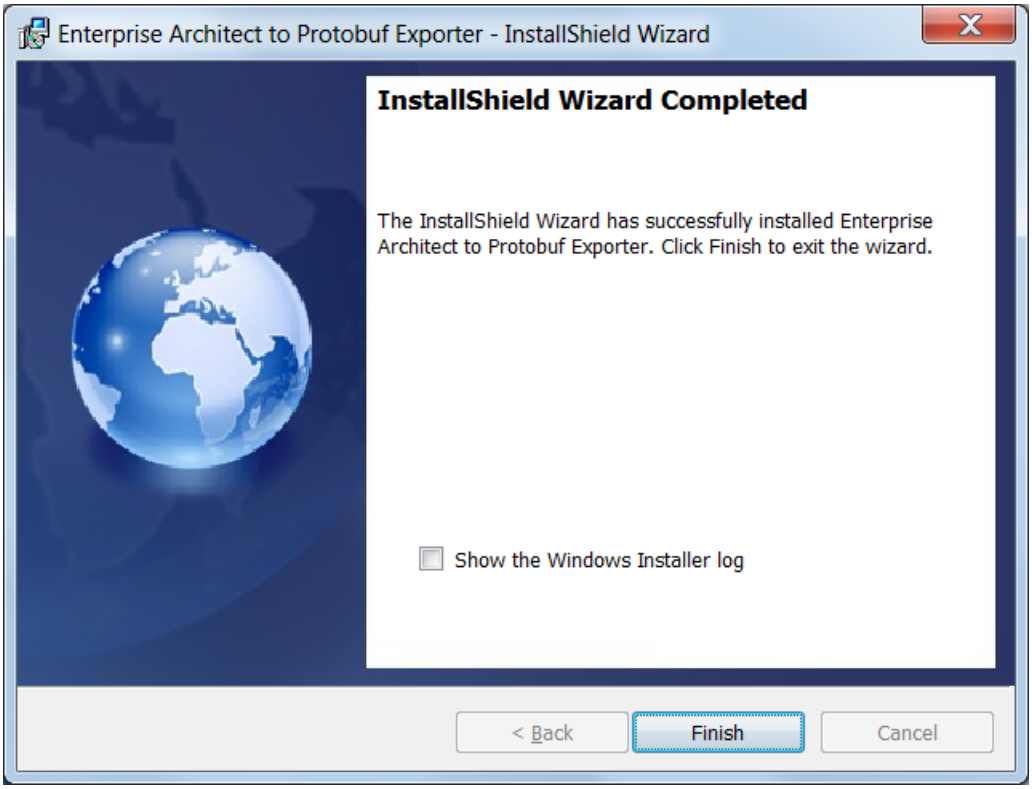
5) Press Install to begin installation



5) Wait for the installation to complete

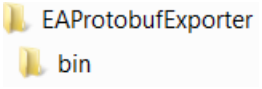


6) Press Finish

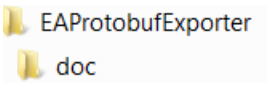


If you accept the default directories during installation you will see the following directory structures

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



- The documentation will be installed under C:\



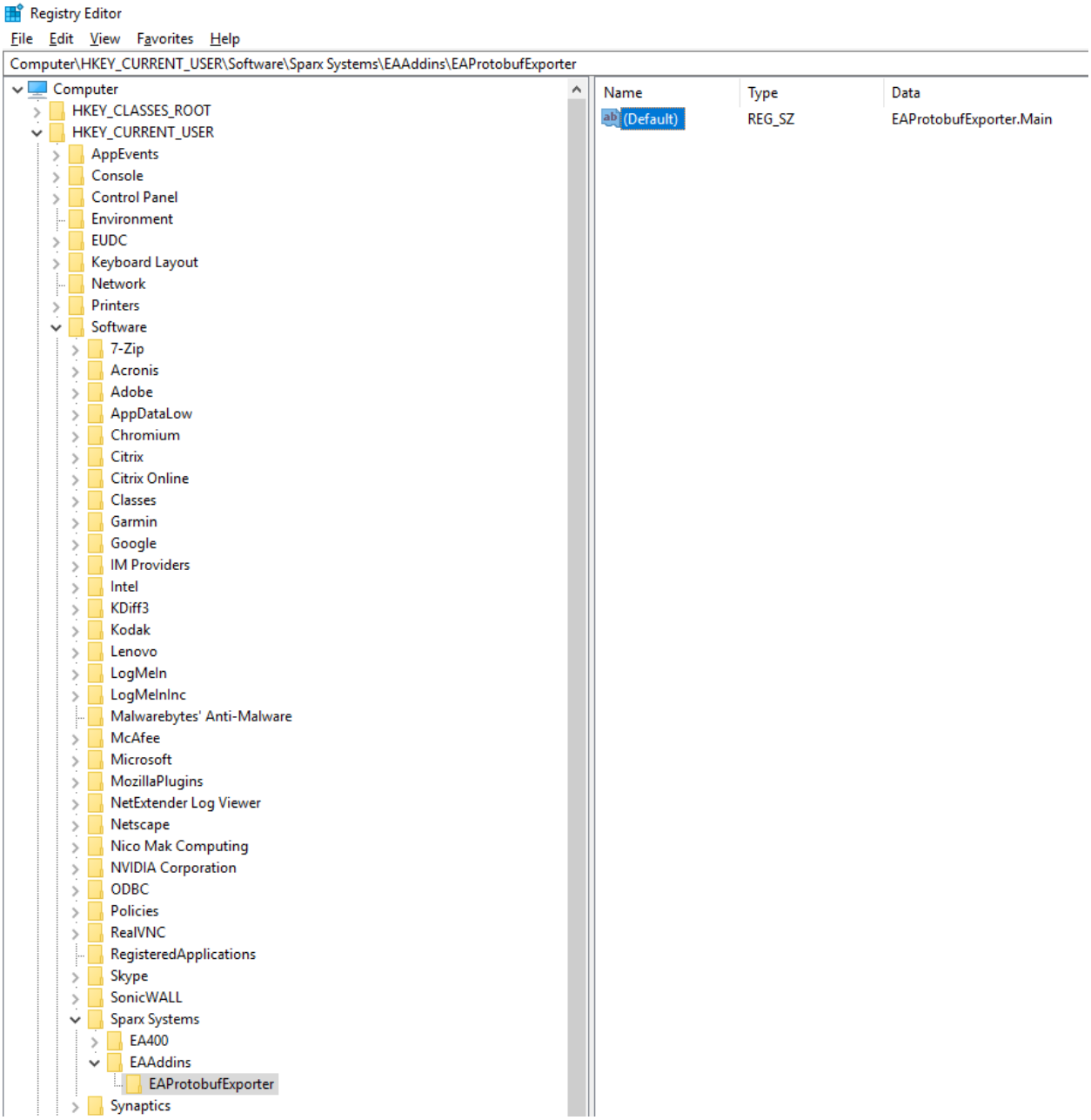
2.2 Optional Manual Installation

If not using the installer program, manually install the Enterprise Architect to Protobuf Exporter by following 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

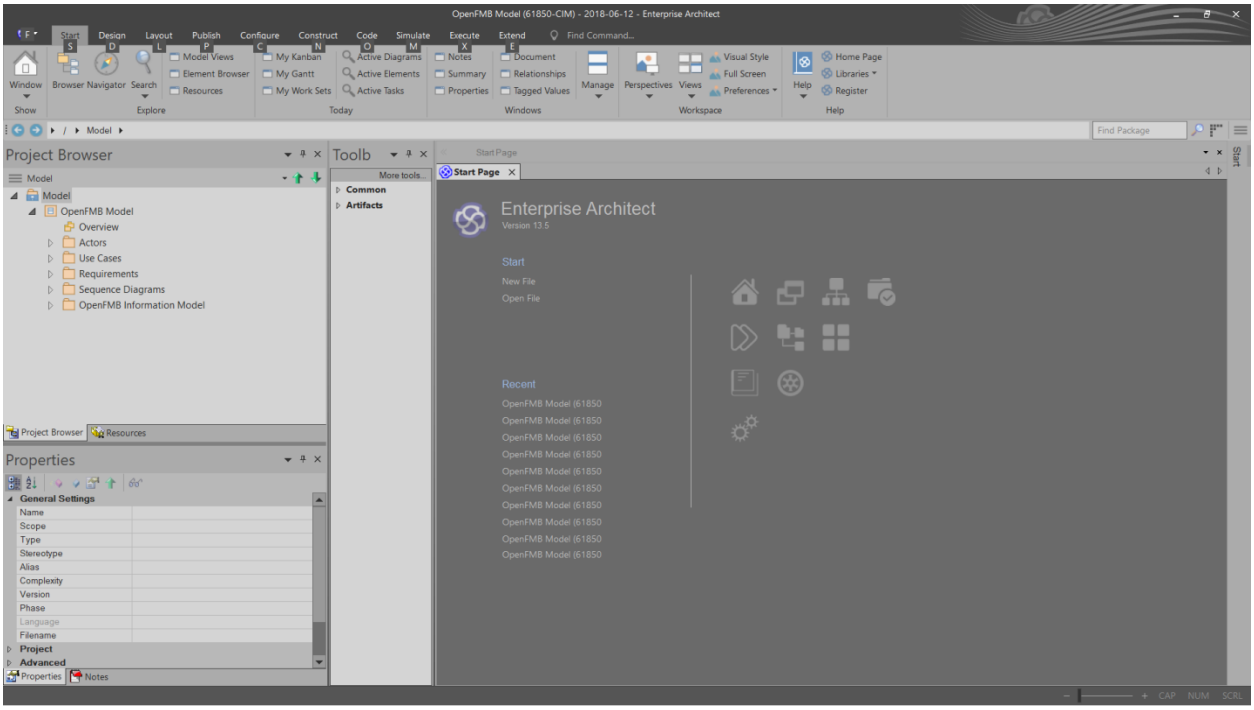
3. Create a new entry in the registry by running regedit. This will allow Enterprise Architect to recognize the presence of the Enterprise Architect to Protobuf Exporter Add-In. Add the new key value "*EAAddIns*" under the appropriate location:
   a. For single users: "*HKEY_CURRENT_USER\Software\Sparx Systems*"
   b. For multiple users: "*HKEY_LOCAL_MACHINE\Software\Sparx Systems*"
4. Under the "*EAAddIns*" key, add a new key value using the project name "*EAProtobufExporter*" of the Enterprise Architect to Protobuf Exporter Add-In
5. 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*"

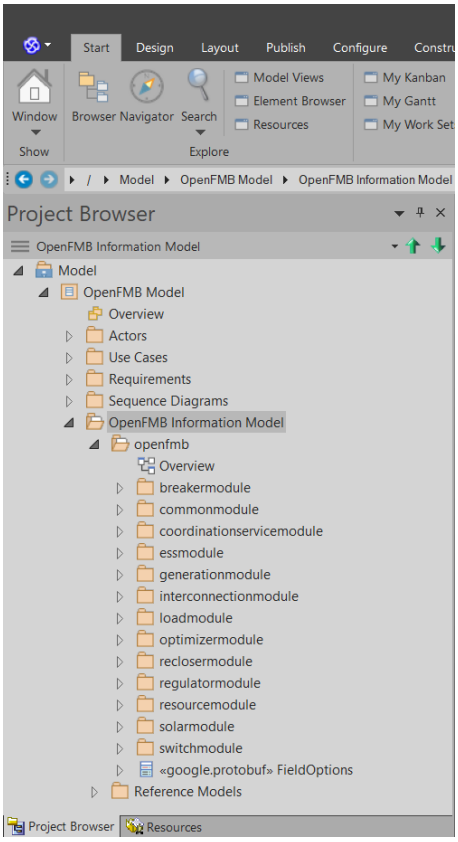

Registry Editor
File Edit View Favorites Help
Computer\HKEY_CURRENT_USER\Software\Sparx Systems\EAAddins\EAProtobufExporter
Computer
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
AppEvents
Console
Control Panel
Environment
EUDC
Keyboard Layout
Network
Printers
Software
7-Zip
Acronis
Adobe
AppDataLow
Chromium
Citrix
Citrix Online
Classes
Garmin
Google
IM Providers
Intel
KDiff3
Kodak
Lenovo
LogMeIn
LogMeInInc
Malwarebytes' Anti-Malware
McAfee
Microsoft
MozillaPlugins
NetExtender Log Viewer
Netscape
Nico Mak Computing
NVIDIA Corporation
ODBC
Policies
RealVNC
RegisteredApplications
Skype
SonicWALL
Sparx Systems
EA400
EAAddins
EAProtobufExporter
Synaptics
Name
Type
Data
(Default)
REG_SZ
EAProtobufExporter.Main

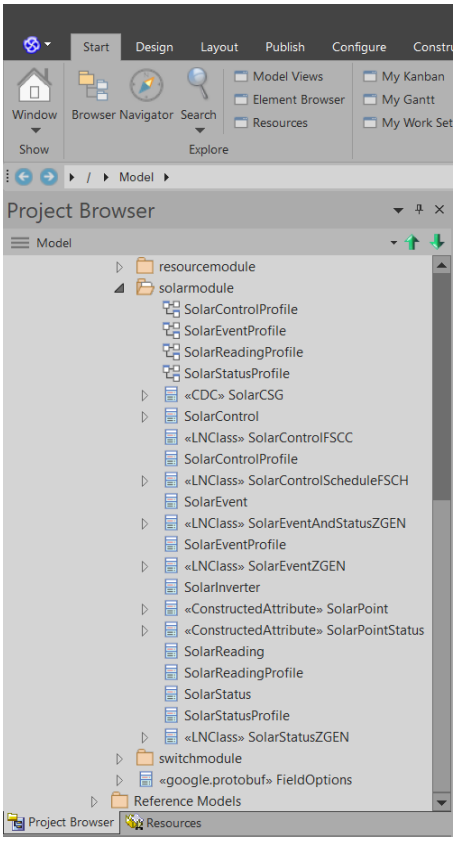
3 Running the Enterprise Architect to Protobuf Exporter Add-In

3.1 Start Enterprise Architect

Start Enterprise Architect and load the appropriate OpenFMB model.

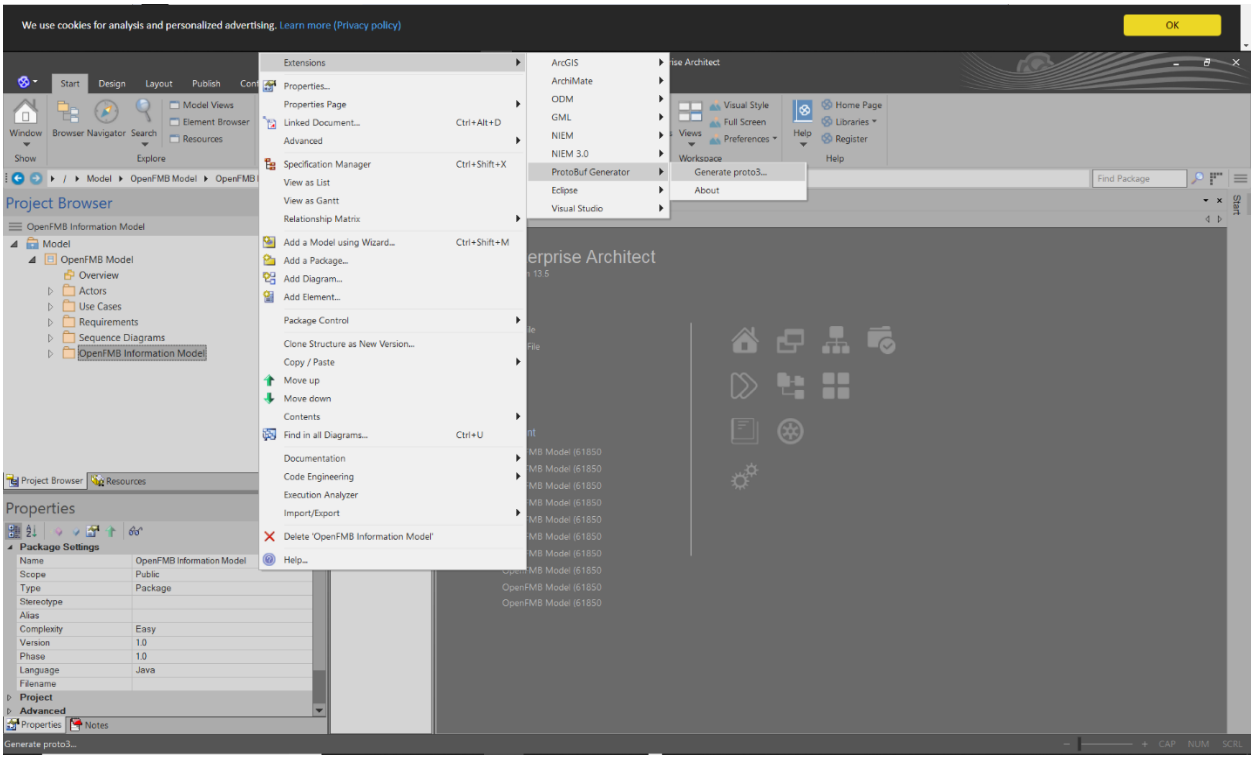



Start
Design
Layout
Publish
Configure
Window
Browser
Navigator
Search
Model Views
Element Browser
Resources
My Kanban
My Gantt
My Work Set
Show
Explore
Model
OpenFMB Model
OpenFMB Information Model
Project Browser
OpenFMB Information Model
Model
OpenFMB Model
Overview
Actors
Use Cases
Requirements
Sequence Diagrams
OpenFMB Information Model
openfmb
Overview
breakermodule
commonmodule
coordinationservicemodule
essmodule
generationmodule
interconnectionmodule
loadmodule
optimizermodule
reclosermodule
regulatormodule
resourcemodule
solarmodule
switchmodule
«google.protobuf» FieldOptions
Reference Models
Project Browser
Resources


Start
Design
Layout
Publish
Configure
Window
Browser
Navigator
Search
Model Views
Element Browser
Resources
My Kanban
My Gantt
Show
Explore
Model
Project Browser
Model
resourcemodule
solarmodule
SolarControlProfile
SolarEventProfile
SolarReadingProfile
SolarStatusProfile
«CDC» SolarCSG
SolarControl
«LNClass» SolarControlFSCC
SolarControlProfile
«LNClass» SolarControlScheduleFSCH
SolarEvent
«LNClass» SolarEventAndStatusZGEN
SolarEventProfile
«LNClass» SolarEventZGEN
SolarInverter
«ConstructedAttribute» SolarPoint
«ConstructedAttribute» SolarPointStatus
SolarReading
SolarReadingProfile
SolarStatus
SolarStatusProfile
«LNClass» SolarStatusZGEN
switchmodule
«google.protobuf» FieldOptions
Reference Models
Project Browser
Resources

3.2 Run the OpenFMB Protobuf Generator

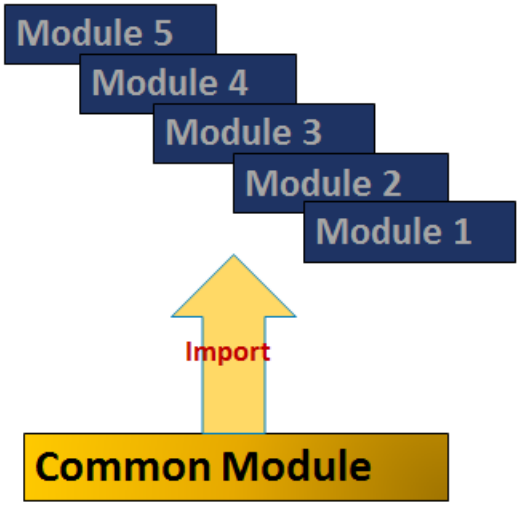
Right click in the "*Project Browser*" window and highlight the "*Extensions*" menu to show a list of the available extensions or add-ins. Highlight the "*OpenFMB Protobuf Generator*" and select "*Generate Protobuf Definitions…*".

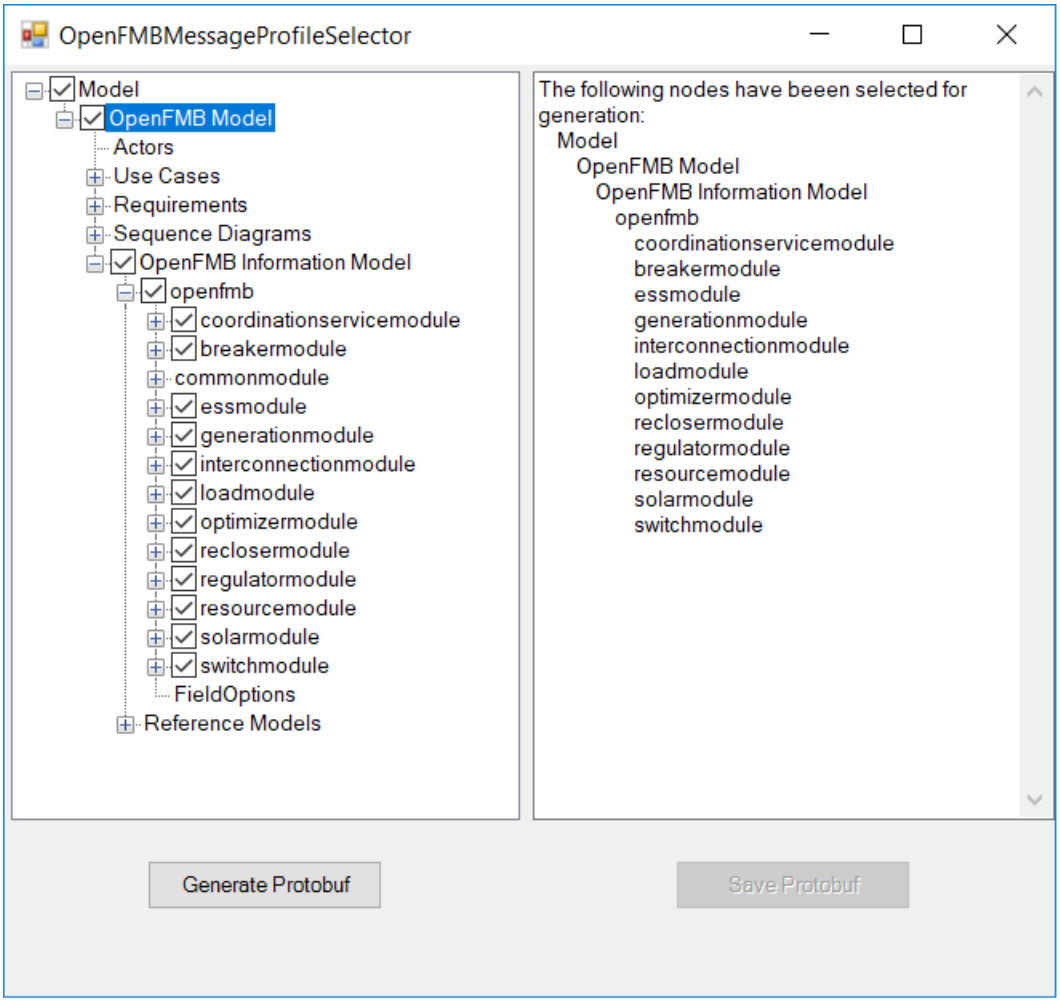


The OpenFMB Protobuf Generator will build a Tree View containing the model information with check boxes next to the Tree Nodes that are specific to the generation of the Protobuf files in the left pane and processing information in the right pane. Since it is used by the other modules, the Common is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

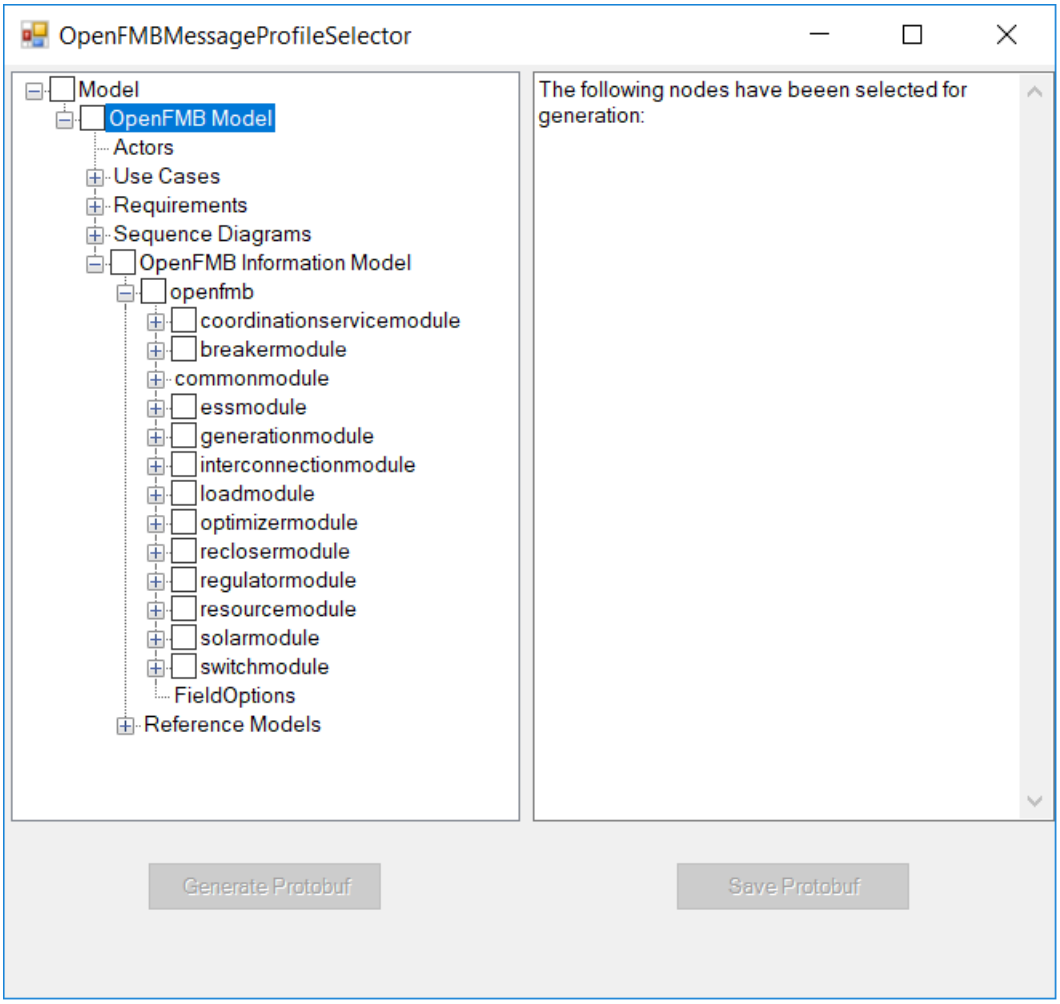
Common Module contains reusable classes shared (imported) across other modules

Each module may contain multiple profiles





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

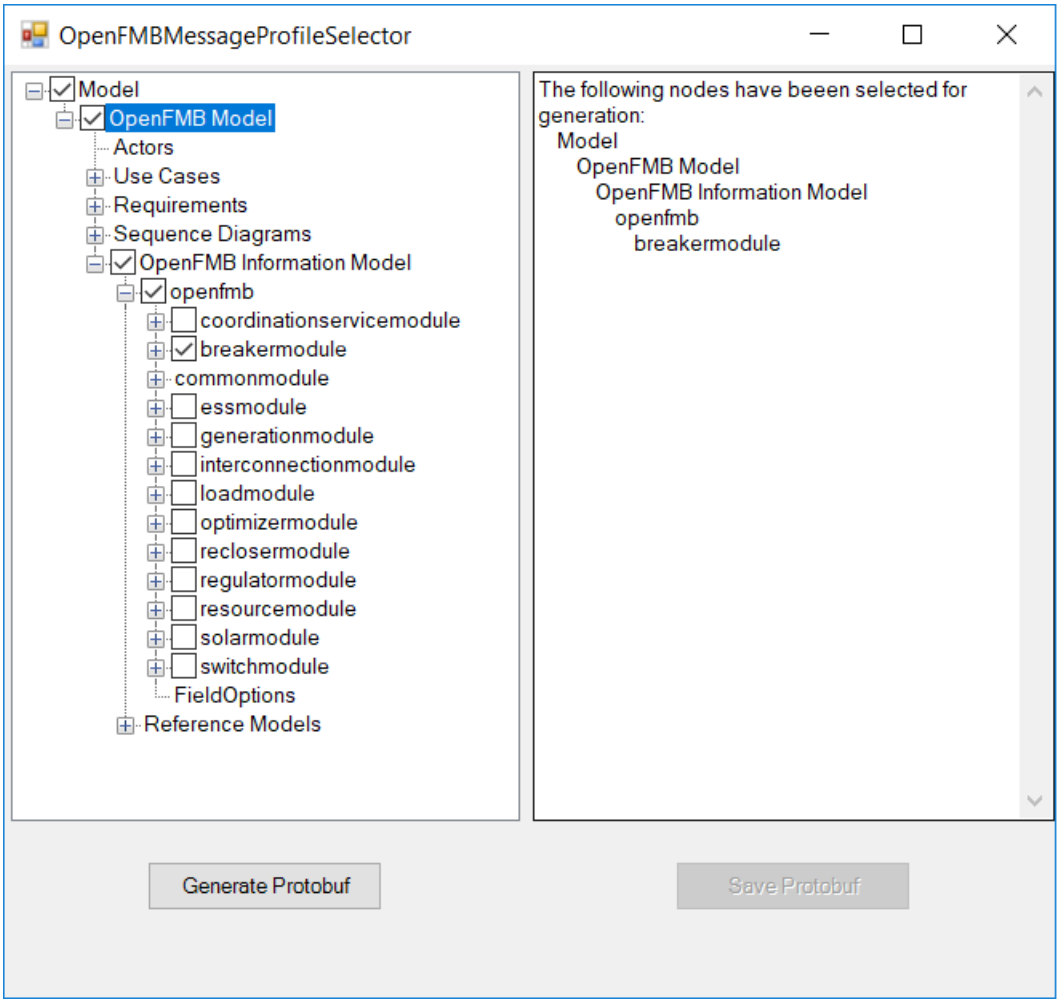


Reselecting nodes will result in the right pane updating to reflect the nodes that are selected.

3.3 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.

3.4 Errors During Generate Protobuf

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

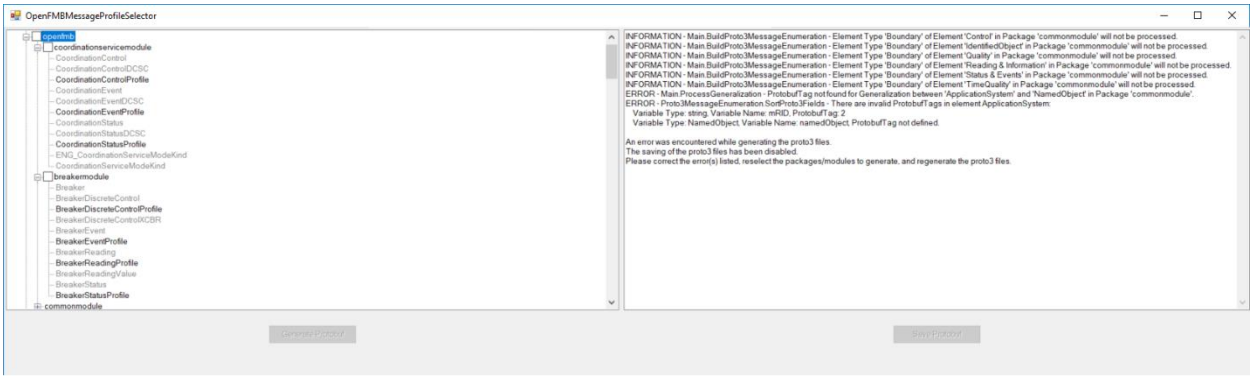
An error was encountered while generating the proto3 files.

The saving of the proto3 files has not been disabled.

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

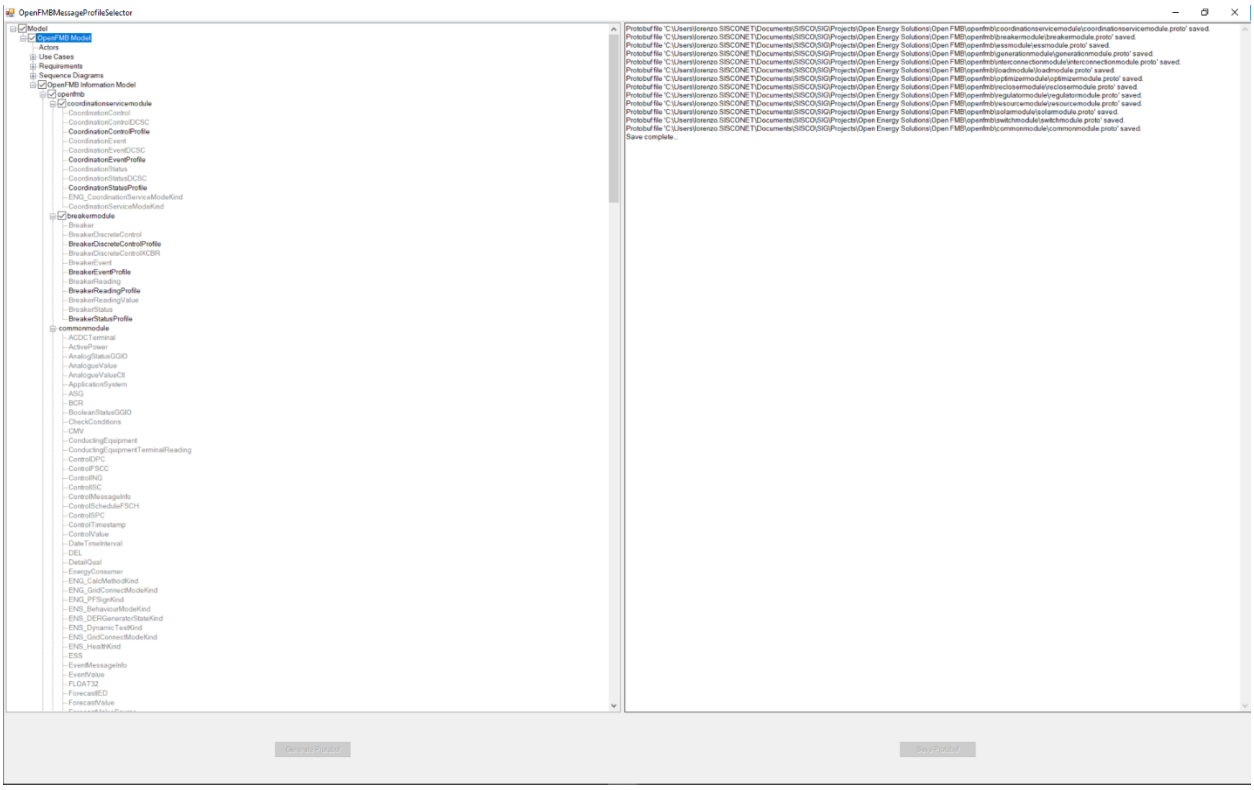
At this point the user has the following options:

- 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

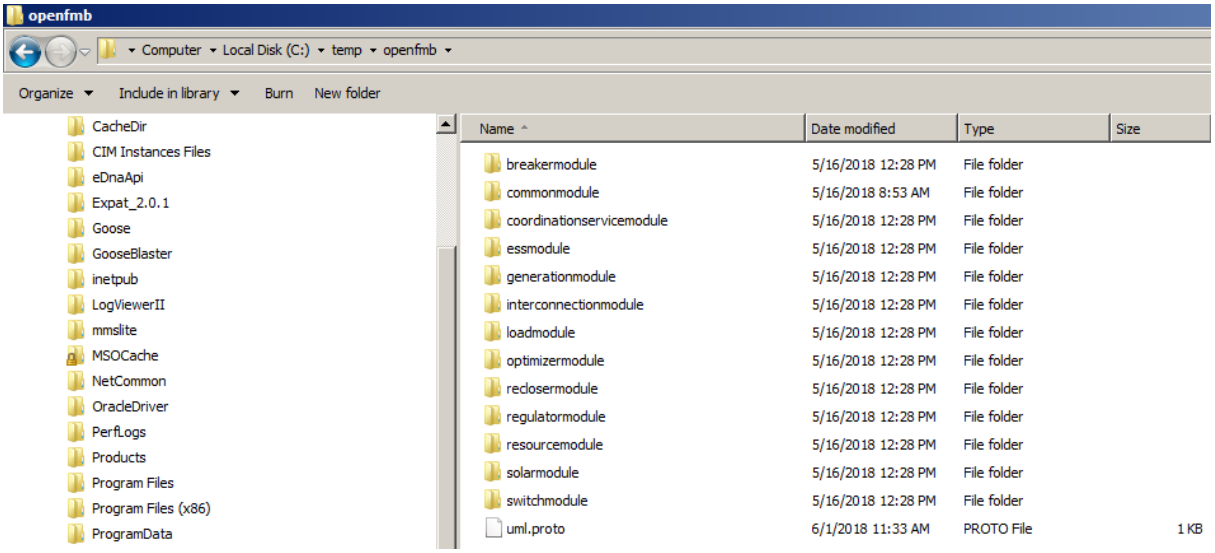


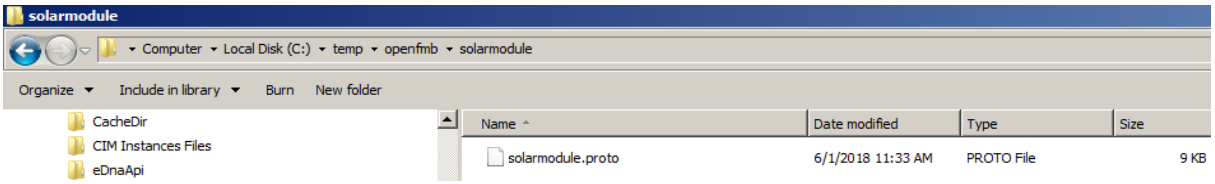
3.5 Save Protobuf

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*>.


openfmb
Computer ▸ Local Disk (C:) ▸ temp ▸ openfmb ▸
Organize
Include in library
Burn
New folder
CacheDir
CIM Instances Files
eDnaApi
Expat_2.0.1
Goose
GooseBlaster
inetpub
LogViewerII
mmslite
MSOCache
NetCommon
OracleDriver
PerfLogs
Products
Program Files
Program Files (x86)
ProgramData
Name
Date modified
Type
Size
breakermodule 5/16/2018 12:28 PM File folder
commonmodule 5/16/2018 8:53 AM File folder
coordinationservicemodule 5/16/2018 12:28 PM File folder
essmodule 5/16/2018 12:28 PM File folder
generationmodule 5/16/2018 12:28 PM File folder
interconnectionmodule 5/16/2018 12:28 PM File folder
loadmodule 5/16/2018 12:28 PM File folder
optimizermodule 5/16/2018 12:28 PM File folder
reclosermodule 5/16/2018 12:28 PM File folder
regulatormodule 5/16/2018 12:28 PM File folder
resourcemodule 5/16/2018 12:28 PM File folder
solarmodule 5/16/2018 12:28 PM File folder
switchmodule 5/16/2018 12:28 PM File folder
uml.proto 6/1/2018 11:33 AM PROTO File 1 KB

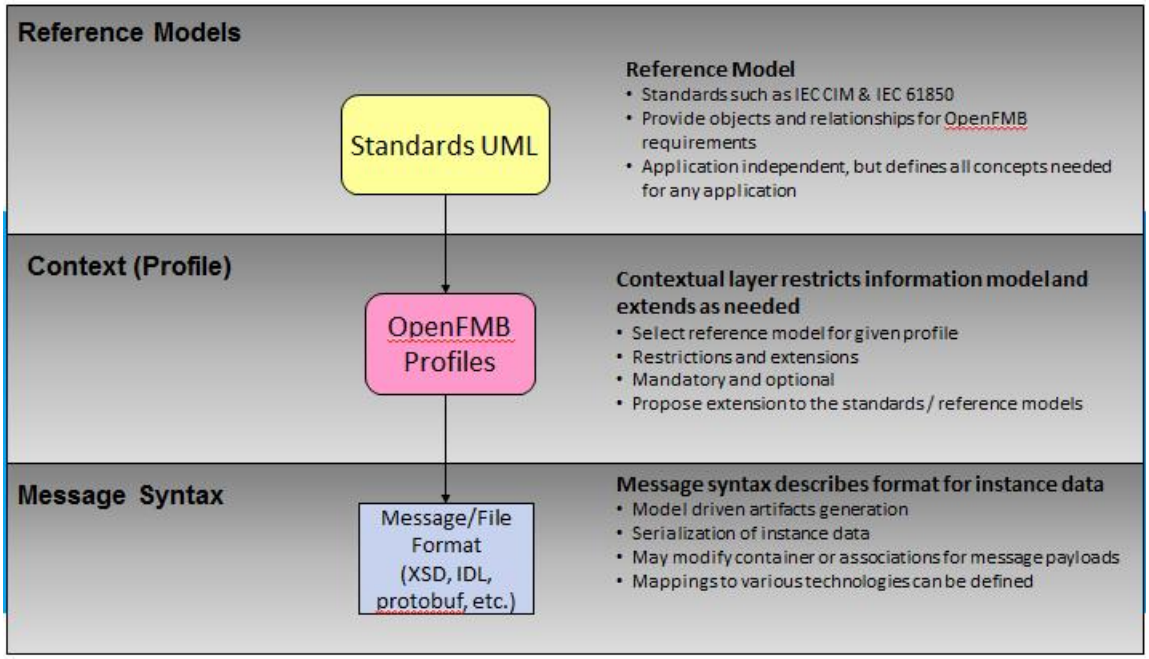

solarmodule
Computer ▸ Local Disk (C:) ▸ temp ▸ openfmb ▸ solarmodule
Organize
Include in library
Burn
New folder
CacheDir
CIM Instances Files
eDnaApi
Name
Date modified
Type
Size
solarmodule.proto 6/1/2018 11:33 AM PROTO File 9 KB

3.6 The OpenFMB Protobuf Generator 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 "*OpenFMB Protobuf Generator*" and select "*About*".

4 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.



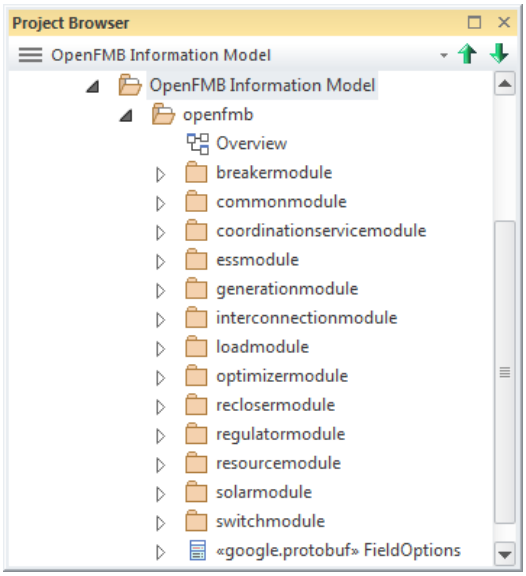
Protocol Buffers require additional metadata beyond that available in standard UML models. To express this information OpenFMB has defined specific UML Tagged Values. In general, the UML tagged values (Sparx Enterprise Architect users guide on tagged values) are used to extend an existing UML model for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

4.1 What are UML tagged values?

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 has more information.

4.2 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.



4.3 How to add UML tags

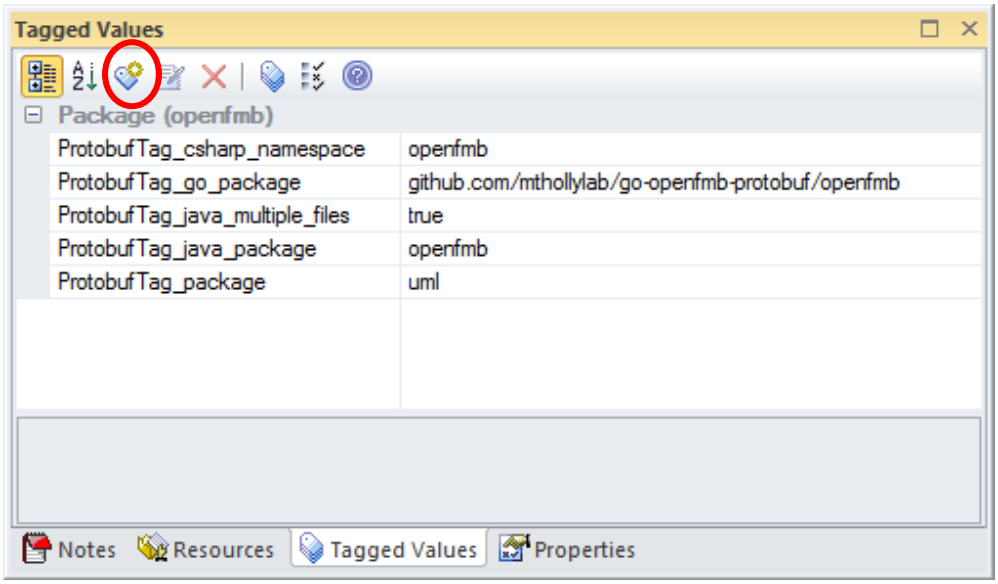
As mentioned, the Protobuf related UML tagged values are used for UML package and individual UML model element such as class and association.

4.3.1 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.



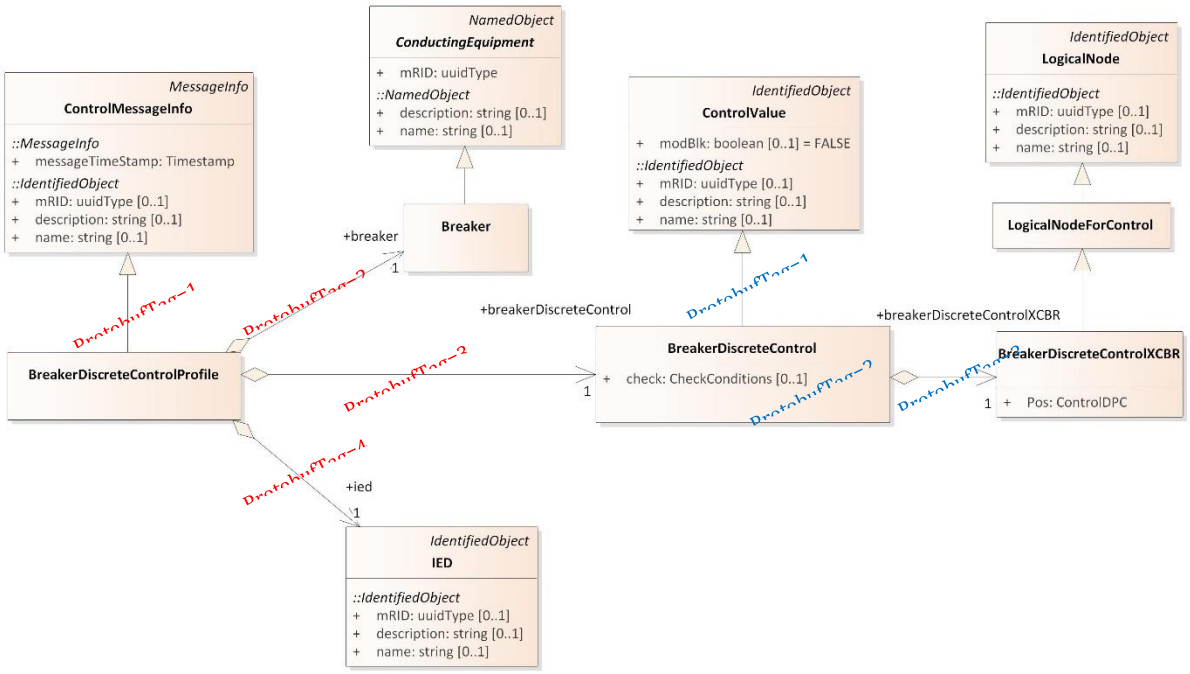
- ProtobufTag_csharp_namespace – Specifies the namespace during Java code generation
- ProtobufTag_go_package – Specifies the package name during Go code generation
- ProtobufTag_java_multiple_files – Causes top-level messages, enums, and services to be defined at the package level, rather than inside 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

4.3.2 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:

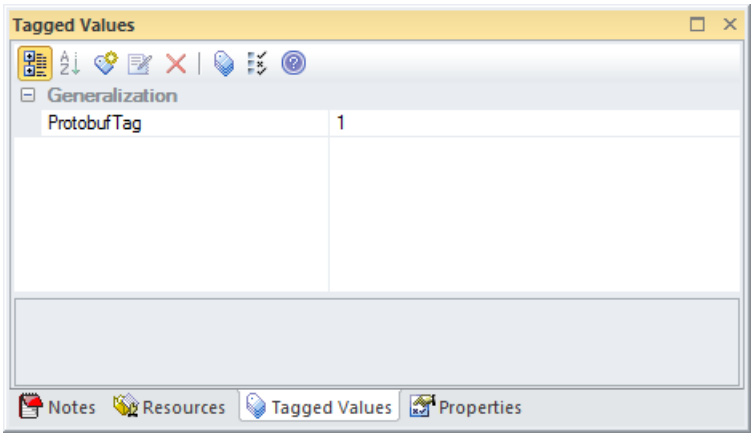
1) Generalization (if exist)
2) Attribute (if exist)
3) Association (if exist)

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.



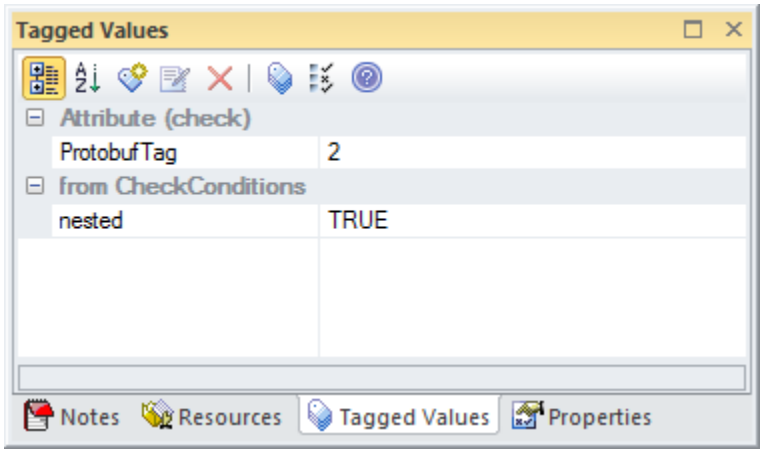
4.3.3 Steps to add a tagged value for Generalization:

1) Select a generalization and bring up its "Tagged Values" window
2) Select Generalization row and add ProtobufTag with corresponding value



4.3.4 Steps to add a tagged value for attribute:

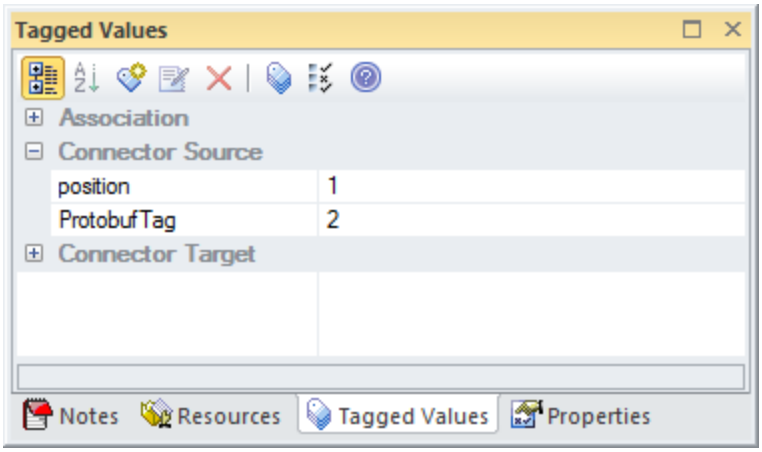
1) Select an attribute and bring up its "Tagged Values" window
2) Select Attribute row and add ProtobufTag with corresponding value



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

4.3.5 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

4.3.6 Top Package Level Field and Message Options

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