

Ecore modeling

Diagrams and elements in Ecore models

There are no specific diagrams for editing Ecore models in MagicDraw. You can use the same Class diagrams as you use for your UML models. Since Ecore is almost a subset of UML (with a few additions), familiar UML elements are used for the modeling. You can also develop Ecore models without using the Ecore profile at all. If your Ecore model uses only UML-specific information, you can develop it using plain UML and export it to Ecore without any problem.

Ecore is even more similar to EMOF. You can export the same model to both Ecore and EMOF.

Class, DataType, Enumeration, Package, Operation, Parameter have a direct one-to-one correspondence between UML and Ecore.

Ecore has two flavors of structural features, EAttribute and EReference, while UML has just one – Property. Fortunately a differentiation between an attribute and a reference is unambiguous and automatically resolved: the property, whose type is a data type is treated as EAttribute; the property, whose type is a class is treated as EReference. Hence the user does not need to worry about this - he/she can simply use properties.

There are no standalone Association and Generalization model elements in Ecore, but there is analogous information in Ecore: two EReferences, pointing to each other by their **opposite** property is equivalent to the association; the EClass::**eSuperTypes** property is equivalent to a generalization. Hence it is possible and meaningful to draw associations and generalizations in your model for exporting this information to Ecore.

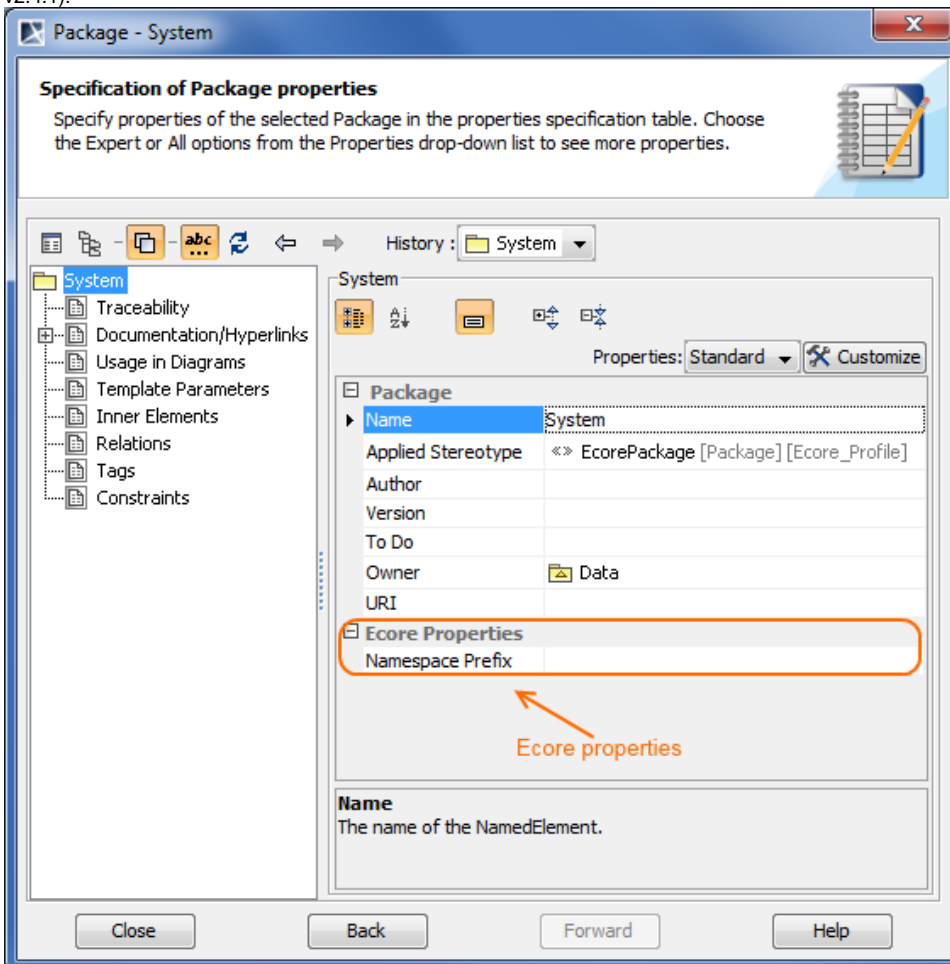
Ecore generics (templates) are also supported. You can use the UML template support to model Ecore generics. While the modeling is not trivial (and not one-to-one due to weak semantics of Ecore's EGenericType), it is possible to model all cases of template types, even ones with complexly nested type bounds like, for example, **SortedList<T extends Comparable<? super T>>**.

Your models can also contain any other UML elements, which are not present in Ecore. These elements are simply skipped during the export to Ecore. A warning is given about these elements.

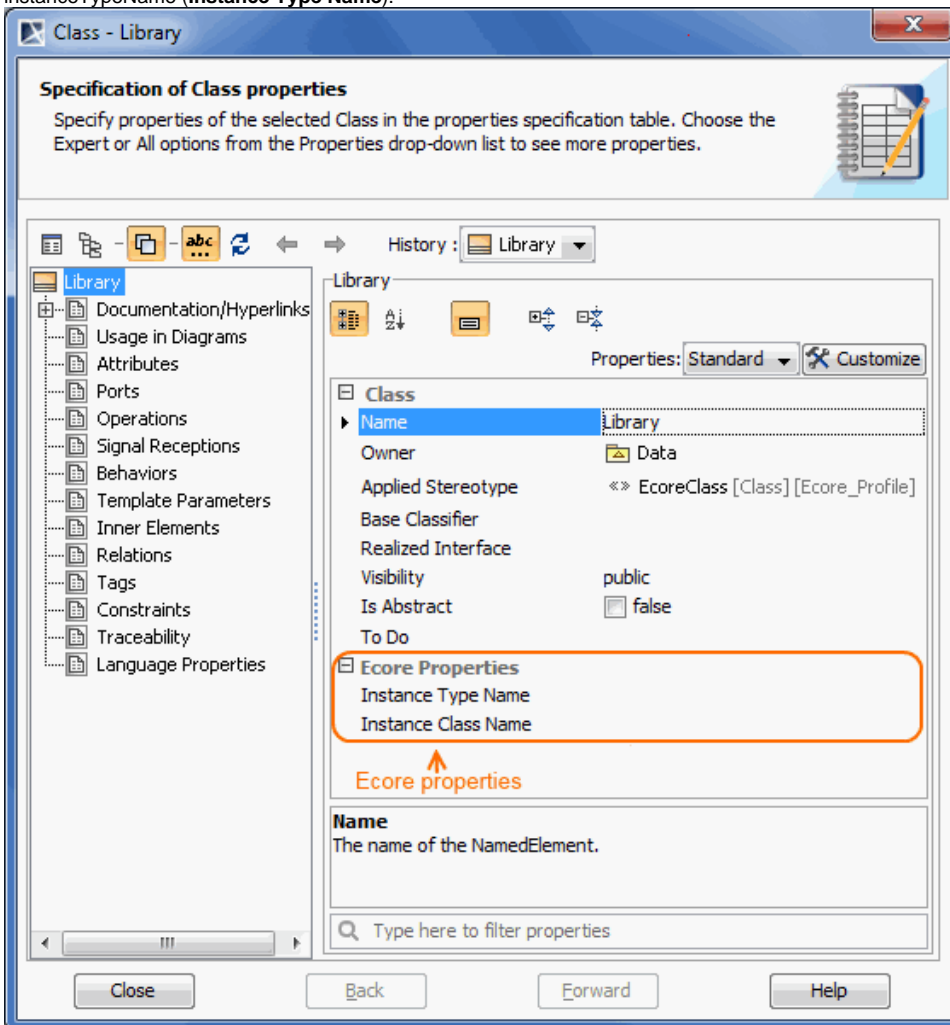
Element properties in Ecore models

There are few Ecore-specific properties, which are brought in when the Ecore profile is used. These properties are used to capture Ecore specific information, not existing in UML. MOF-specific properties are also relevant for Ecore. These special properties are as follows:

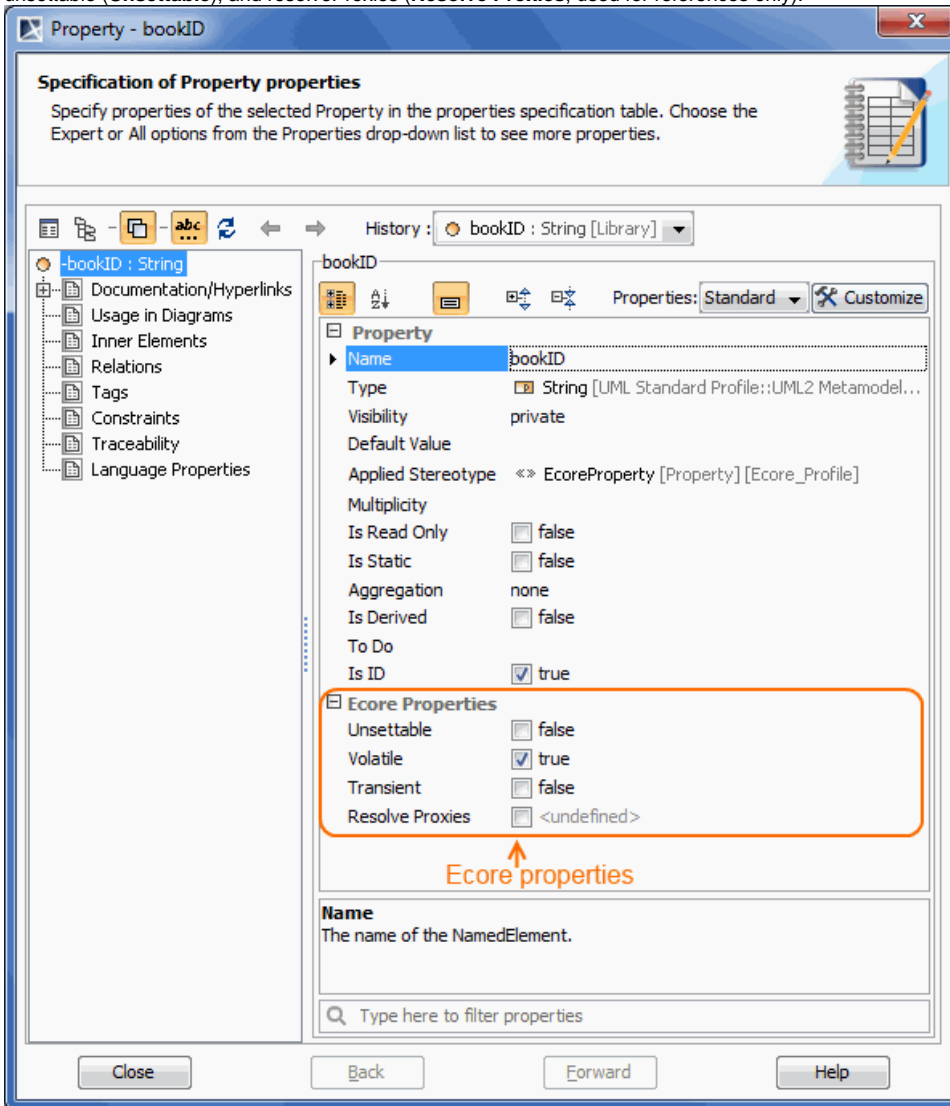
- The Ecore package has the additional properties: nsPrefix (**Namespace Prefix**) and nsURI (corresponds to the **URI** property specified in the UML v2.4.1).



- The Ecore classifier (class, data type, enumeration) has the additional properties: instanceClassName (**Instance Class Name**) and instanceTypeName (**Instance Type Name**).



- Ecore attributes and references (modeled as the UML property) have the additional properties: volatile (**Volatile**), transient (**Transient**), unsettable (**Unsettable**), and resolveProxies (**Resolve Proxies**, used for references only).



Annotation modeling

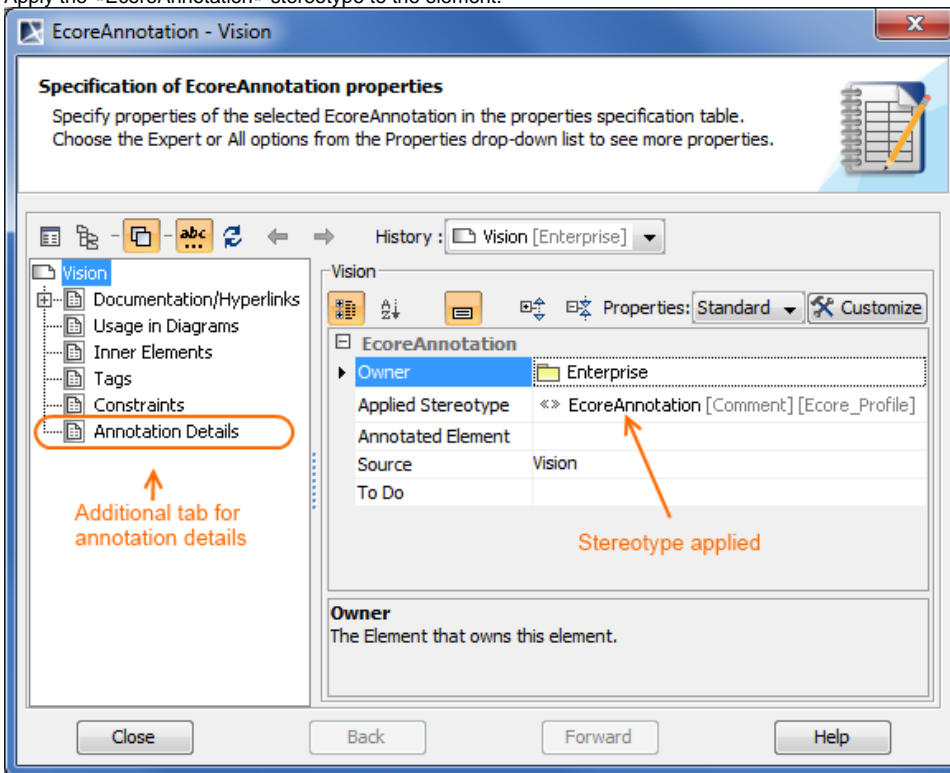
Ecore annotations are modeled as UML comments. For simple annotations no additional actions are necessary.

However, Ecore annotations have more powerful semantic than UML comments – they can have an internal substructure. In particular they can have an additional key-value map. For this additional information, there is a special «EcoreAnnotation» stereotype, that can be applied on an annotating comment. After applying the stereotype, the key-value map can be entered in a separate node of the annotating comment Specification window. Key-value pairs are stored as internal subcomment elements of the annotation.

To create an Ecore annotation

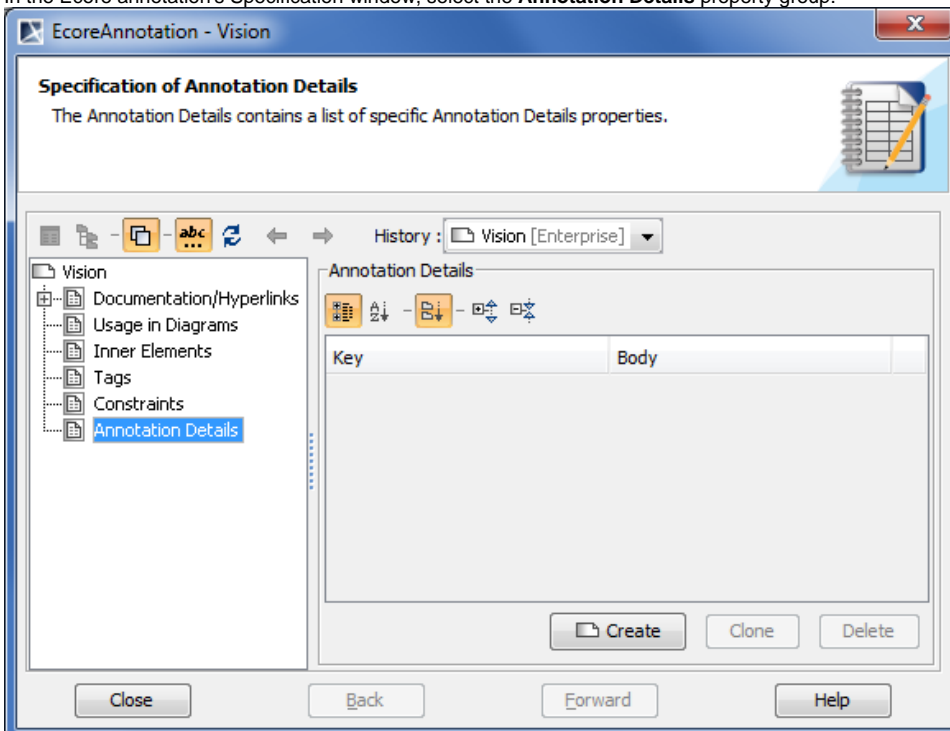
1. In the Ecore model, create a Comment element.

2. Apply the «EcoreAnnotation» stereotype to the element.

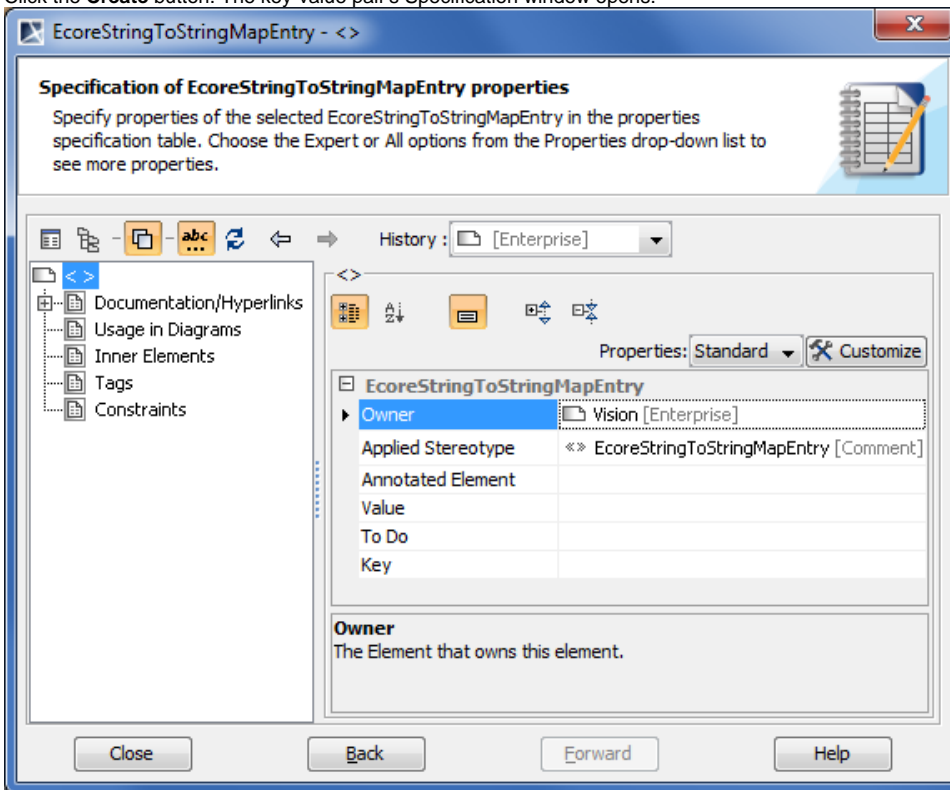


To create a key-value pair

1. In the Ecore annotation's Specification window, select the **Annotation Details** property group.



2. Click the **Create** button. The key-value pair's Specification window opens.



3. Enter values for both the **Value** and **Key** properties.
4. Click the **Back** button when you are done. You will see the key-value pair created.

