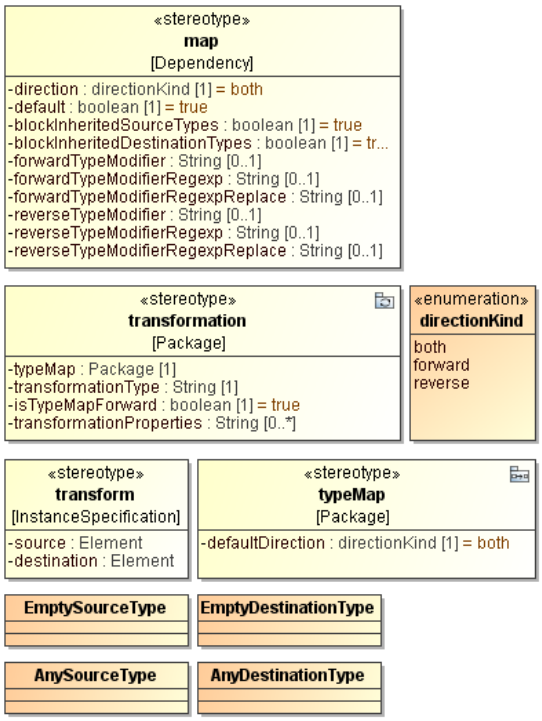
Transformation Type Mapping

During the transformation between the different modeling domains, such as UML and SQL, it is necessary to go through data types used in the source model and change the types from the source domain into the equivalent types in the target domain, for example, changing String type usages in the UML model into varchar type

usages in the SQL model. This is achieved by establishing a type map and then supplying it for the transformation (many transformations have default, predefined type maps).

A type map can be regarded as a collection of rules of the form “Replace the usage of type X with the usage of type Y”. A type map is created by modeling means and is a model object, hence all model manipulation operations can be performed on it. In particular - it can be refactored into a module and mounted into any project, that needs it. It can be a simple package in your project as well, if you need a custom, one-off type map. Predefined type map can be taken from the MagicDraw module and edited.

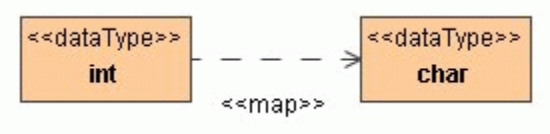
A type map is a stereotyped package, holding a collection of stereotyped dependencies. Stereotypes for building type maps are stored in the *Model Transformation Profile*.



To create a transformation type map

---

1. Use or import *Model_Transformation_Profile.xml.zip*.
2. Create a package, which will represent your type map.
3. Apply a stereotype *«typeMap»* to the created package.
4. Choose types (data types, classes, enumerations) in your source domain and their corresponding types in your target domain.
5. Create the desired dependency relationships between the corresponding types.
6. Apply a stereotype named *«map»* to these dependencies.



Be sure to place dependencies in the type map package (MagicDraw is prone to placing dependencies in or near the dependent model element, so you may need to relocate them).

In the example above, after the transformation, all *int* types will be transformed to *char*.

Each of thus created dependencies represents one type remapping rule. The package represents the complete type map.

Type mapping rule behavior can be further customized by setting various tags on the rules.

Transitive type mapping (of the form type1 > type2 > type3) is not supported.