## SIMF Profile::SIMF Computation Rules

Computation rules define mappings that are implemented via external methods. As such the implementation is defined by implementations, not the specification.

### Diagram SIMF Computation Rules

1940672626.emf

**Figure 1 SIMF Computation Rules**

### Class ExistsRule

<<Exists Rule> is a rule to map the existence of an <element> to a boolean.<exists> is true iff <element> is not null.

Direct Supertypes

* [Rule Computation](#_7b2ad80ba8e04ff9ff1cb7b89d9ea1d3)

Attributes

-1258462637.emf element

-1258462637.emf exists : [Boolean](#_6119a00b0834641b9fe3f5ae9f58237f)

### Class List First

The <List First> rules will take the <list> property and place the first element into<first>. If <list> is empty, <first> will be empty.If there are more <list> elements than 1, all remaining elements are placed as a set in <remainder>.If <list> is an un-ordered set the order will be indeterminate but repeatable.<<List First>> is bidirectional and will compute <list> by appending <first> and <remainder>.Note that this will act like a LISP CDR/CAR pair

Direct Supertypes

* [Rule Computation](#_7b2ad80ba8e04ff9ff1cb7b89d9ea1d3)

Attributes

-1258462637.emf first [0..1]

-1258462637.emf list [\*]

-1258462637.emf remainder [\*]

### Class MapID

<<MapID>> is a rule where the source is an ID and the target is a class, maps an instance of the ID to an instance of the class.

Direct Supertypes

* [Rule Computation](#_7b2ad80ba8e04ff9ff1cb7b89d9ea1d3)

Attributes

-1258462637.emf id : [Classifier](#_5f9a87915e1e9718a1a1cc45af995a70)

-1258462637.emf identified : [Classifier](#_5f9a87915e1e9718a1a1cc45af995a70)

### Class Rule Computation

<<Rule Computation>> is an abstract supertype for a facade that includes external implementation. The implementation is outside of this specification.

### Class Summarize

<<Summarize>> is a rule that produces a natural language description of an element. Summarize may not be bi-directional and is expected to have information loss.<summary> is a summary of <element>. Content of summary is implementation specific.

Direct Supertypes

* [Rule Computation](#_7b2ad80ba8e04ff9ff1cb7b89d9ea1d3)

Attributes

-1258462637.emf element

-1258462637.emf summary : [String](#_e8a6ce315d976318da3ab784a645ea44)

## SIMF Profile::SIMF Concept Modeling Profile

The conceptual modeling profile defines the conceptual modeling capabilities of SIMF in UML.

### Diagram SIMF Conceptual Modeling Profile

430546471.emf

**Figure 1 SIMF Conceptual Modeling Profile**

### Stereotype Annotation

An <<Annotation>> comment provides a textual "body" as a "value for" one <<Annotation Property>> describing the annotatedElement(s).

Base Classes

* [Comment](#_f0cd78257a5f2650a07898e11268debe)

Tag Definitions

958301894.emf value for : [Annotation Property](#_78cccb8676a335918ea57bcbf344cfcc) [1]

<value for> is the property for which the <<Annotation>> is providing a value.

### Stereotype Annotation Property

An <<Annotation Property>> is a kind of <<Resource>> that asserts a property represents metadata rather than assertions about the subject domain.

Base Classes

* [Property](#_9624701f3c3fcd329e6dba1fc74db836)

Direct Supertypes

* [Resource](#_7eb9b01488b85cfcd94ce909e4efc52d)

### Stereotype Anything

<<Anything>> is a class that represents anything and is equivalent to all other classes of anything in any other model or logic. The defined class is equivalent to SIMF:Anything, OWL:Thing and other "top level" classes.Because of this equivalence, every class in every model virtually inherits from Anything, just as all OWL classes virtually inherit from owl:Thing.<<Anything>> classes may be used to define "global properties".

Base Classes

* [Classifier](#_5f9a87915e1e9718a1a1cc45af995a70)

### Stereotype Base Unit Type

<<Base Unit Type>> is a kind of <<Unit Type>> that marks one unit type of a quantity kind as the base unit type within a model. The base unit type provides the basis for conversions between units of the same quantity kind. The base unit always has a ratio of one and an offset of zero.

Base Classes

* [Classifier](#_5f9a87915e1e9718a1a1cc45af995a70)

Direct Supertypes

* [Unit Type](#_fff22704476e57332bbc8cc81106bf0e)

### Stereotype Classifies

A classification defined by a <<Classifies>> generalization or realization is a "mix in" or "non rigid" classification of an entity beyond any fundamental entity type. An instance must be typed by the classifies supertype for it to also be classified as the classifies subtype. A classification may be contextual, such as within a relation, situation and/or time frame. Instances may have any number of types and classifications may change over time.Classification is used in defining what a <<Role>> may be a role of, and for phases, what a <<Phase>> is a phase of.Classifications may be added to or removed from an individual over time and in different context.

Base Classes

* [Generalization](#_3596249272c463c4fc390555f5203d9e)

### Stereotype Concept Model

A <<Concept Model>> is a kind of <<Model>> that represents concepts in a real or possible world. Instances of elements in a concept model are "real world" things, not data about those things.

Base Classes

* [Package](#_657662b69a91b4b82844aca2e3465cd3)

Direct Supertypes

* [Model](#_86d8bba2473f751aaa57d08526f58037)

### Stereotype Disjoint With

A <<Disjoint With>> dependency is an assertion that two model elements do not and may not denote any of the same set of entities.When applied to a classifier, every element of the classifier's extent (set of instances) is included in the set of disjoint things.

Base Classes

* [Dependency](#_c8808432037fd32b0da9b016aeac1fa5)

### Stereotype Enumerates

An <<Enumerates>> dependency asserts that the supplier of the dependency is a type and the client of the dependency is a package containing a complete set of possible instance specifications. In this way, <<Enumerates>> is more general than a UML Enumeration because it can enumerate more than just UML data types.

Base Classes

* [Dependency](#_c8808432037fd32b0da9b016aeac1fa5)

### Stereotype Equivalent Class

A <<Equivalent Class>> generalization is an assertion that two classes have the same extents (set of instances). Unlike ontological languages it is not assumed that the two elements are consistent, as statements from different context may or may not agree.

Base Classes

* [Generalization](#_3596249272c463c4fc390555f5203d9e)

### Stereotype Equivalent Property

<<Equivalent Property>> is a declaration that a property is equivalent to one or more other properties (using "equivalent to") or is equivalent to a chain of other properties (using "chain"). <<Equivalent Property>> with at least one value for the "equivalent to" property is an alternative way of expressing <<Equivalent To>>, without introducing crossing lines on a diagram.Either "equivalent to" or "chain" must have a value.

Base Classes

* [Property](#_9624701f3c3fcd329e6dba1fc74db836)

Tag Definitions

958301894.emf chain : [Property](#_9624701f3c3fcd329e6dba1fc74db836) [\*]

An ordered set of properties forming a "property composition" expressing a traversal path that is equivalent to the stereotyped property. This is similar to a "property chain". (Note that in an OWL property chain, the property composition is not equivalent, it is a subproperty.)Due to potential "missing information" in creating a chain, a chain may or may not be able to be determined from asserting the chained property. Such a determination is defined in the mapping rules for that property in a particular context.Note that a chain may also be defined with mapping rules.

958301894.emf equivalent to : [Property](#_9624701f3c3fcd329e6dba1fc74db836) [\*]

A set of properties that the <<Equivalent Property>> is equivalent to. Note that equivalence can also be declared with a <<Equivalent To>> dependency.

### Stereotype Equivalent To

An <<Equivalent To>> dependency is an assertion that two model elements represent the same thing or the same set of things. Unlike ontological languages it is not assumed that the two elements are consistent, as statements from different contexts may or may not agree.

Base Classes

* [Dependency](#_c8808432037fd32b0da9b016aeac1fa5)

### Stereotype External Reference

<<External Reference>> provides traceability to the source of a "fact" in a model based on some external information resource. This references helps to facilitate provenance. Reference is a statement about the model data and has no semantic implication. Source reference may impact the trust in a statement but the evaluation of trust is outside of this specification. External reference is combined with the owned comment(s) to create SIMF descriptions as defined in the SIMF meta model..

Base Classes

* [Element](#_5d5cfd5b8453e2037ccdadc05f608c6d)

Tag Definitions

958301894.emf external reference : [String](#_e8a6ce315d976318da3ab784a645ea44)

Specifies the location URL of the external resource. The format must comply with [RFC3987].

958301894.emf external term : [String](#_e8a6ce315d976318da3ab784a645ea44)

The external term or location of the information in the source. The form of expression of the term or term path is dependent on the referenced technology.

### Stereotype Has Value

A <<Has Value>> dependency asserts that the client of the dependency is a type and the supplier of the dependency is an instance specification that defines acceptable values for one or more properties of that type. Each slot of the instance specification is a value for a corresponding property in the type. <<Has Value>> corresponds to one or more OWL property restrictions containing a "hasValue" constraint.

Base Classes

* [Dependency](#_c8808432037fd32b0da9b016aeac1fa5)

### Stereotype Information Model

An <<Information Model>> is a kind of <<Model>> that represents a model for some purpose, independent of technical implementation. An information model may contain logical models or data models, as well as other logical viewpoints.

Base Classes

* [Package](#_657662b69a91b4b82844aca2e3465cd3)

Direct Supertypes

* [Model](#_86d8bba2473f751aaa57d08526f58037)

### Stereotype Intersection

An <<Intersection>> is a class that has an extent (set of instances) equivalent to the intersection of the extents of all supertypes. Intersection is a stronger statement than a subtype, as a subtype may be a subset of the intersection. An instance of all the supertypes implies an instance is also an instance of the intersection type.For intersection, The SIMF profile considers UML generalization and UML realization equivalent. This is due to ownership and legacy considerations in UML. Generalization is the preferred representation.Note: Realizations are included to support unions across external models. UML generalization can not be used across external models due to the ownership of generalization.

Base Classes

* [Classifier](#_5f9a87915e1e9718a1a1cc45af995a70)

### Stereotype Is In Context

<<Is In context>> is an assertion that the client of the dependency is in the context of the supplier of the dependency. All assertions and rules defined in the supplier context apply to the client and everything in the context of the client (i.e., it is transitive). Packages, classes, situations and instances are typical contexts. Note that <<Is In Context>> is the default interpretation of a dependency, if no stereotype is specified it will be interpreted as <<Is In Context>>.

Base Classes

* [Dependency](#_c8808432037fd32b0da9b016aeac1fa5)

### Stereotype Model

<<Model>> is stereotype of package that may have an id (see <<Resource>>) and/or a namespace prefix (like the "dc" in "dc:title").

Base Classes

* [Package](#_657662b69a91b4b82844aca2e3465cd3)

Tag Definitions

958301894.emf namespace prefix : [String](#_e8a6ce315d976318da3ab784a645ea44)

A hint as to an appropriate abbreviation for a model that may be used in some technology mappings, such as XML. The prefix should be short and contain only letters and numbers and must start with a letter. e.g., "dc" in "dc:title".

Direct Supertypes

* [Resource](#_7eb9b01488b85cfcd94ce909e4efc52d)

### Stereotype Phase

A <<Phase>> (a.k.a. "State") is a classification of an entity based on change of that entity over time. A <<Phase>> <<Classifies>> the types that may have that phase (e.g., "Teenager").A phase is a "non rigid sortal", a type that may change over the lifetime of an entity.

Base Classes

* [Class](#_bdb071ba1da37974bd8d9639d911511b)

### Stereotype Quantity Kind

<<Quantity Kind>> is an aspect common to mutually comparable quantities represented by one or more units. Units with a common quantity kind may be algorithmically converted to any other unit of that quantity kind. e.g. temperature.[ JCGM 200:2008]. Units with a common quantity kind may be algorithmically converted to any other unit of that quantity kind. e.g. temperature. SIMF takes a wider view of quantity kinds to include conversions that may be contextual and time dependent, such as currencies.

Base Classes

* [Classifier](#_5f9a87915e1e9718a1a1cc45af995a70)

Direct Supertypes

* [Value Type](#_66c012755132097a23386c38f992f6b2)

### Stereotype Resource

A <<Resource>> is anything that can be referenced by an identifier in a model, ontology or vocabulary. This identifier is often an IRI.

Base Classes

* [NamedElement](#_0907197416161d85b05eb0e9c47d90e9)

Tag Definitions

958301894.emf id : [String](#_e8a6ce315d976318da3ab784a645ea44)

A unique identifier for any resource. When defined for a Package, id has the format defined in [RFC3987]. In this case, it is equivalent to UML:URI, and setting one will set the other.

### Stereotype Role

A <<Role>> is a classification of an entity based on that entity's behavior, participation in a situation, or capabilities. A <<Role>> <<Classifies>> the types that may play that role. e.g., "Teacher".A role is a "non rigid sortal", a type that may change over the lifetime of an entity.

Base Classes

* [Class](#_bdb071ba1da37974bd8d9639d911511b)

### Stereotype Sufficient

Specifying <<Sufficient>> for one or more of a type's properties means that an instance having an acceptable cardinality of values for all of those properties implies that the instance is an instance of that type.

Base Classes

* [Property](#_9624701f3c3fcd329e6dba1fc74db836)

### Stereotype Synonym

<<Synonym>> defines an alternate name for the annotated elements of the comment. The alternate name is the body of the comment.The alternate name will not be the "preferred name" of the element.

Base Classes

* [Comment](#_f0cd78257a5f2650a07898e11268debe)

### Stereotype Union

A <<Union>> is a class that has an extent (set of instances) which is equivalent to the union of the extents of all types that specialize the Union (Subclasses). Specializing types shall include subtypes and types that realize the union.Note: UML realizations are included to support unions across external models because UML generalization can not be used across external models due to the ownership of generalization.[MathWorld] Given two sets A and B, the union is the set that contains elements or objects that belong to either A or to B or to both.

Base Classes

* [Classifier](#_5f9a87915e1e9718a1a1cc45af995a70)

### Stereotype Unit Type

A <<Unit Type>> is a <<Value Type>> and an <<External Reference>> that represents a type of a quantity value referencing a specific unit. A Unit Type [?TBD] a required type of a property representing a quantity. [JCGM 200:2008] A Unit is a real scalar quantity, defined and adopted by convention, with which any other quantity of the same quantity kind can be compared to express the ratio of the two quantities as a number. e.g. Degrees Centigrade, Miles.Each unit type represents refinement of a quantity kind using generalization and is thus substitutable for that quantity kind. Typically quantity kinds are used in conceptual models and unit types in physical or logical models.Unit types may only subtype quantity kinds and numbers.Note that unit types are not units, but the type of quantity values expressed in a common unit as defined in [JCGM 200:2008].Each instance of a unit type shares a common unit (as defined by standards) with a reference defined by "external reference" and "external term".

Base Classes

* [Classifier](#_5f9a87915e1e9718a1a1cc45af995a70)

Tag Definitions

958301894.emf offset : [Real](#_aef4bcae5ebc35dd9653214547b3e3cc)

The difference between zero in the unit and zero in the base unit after the ratio is applied to the base unit as defined within the same model.

958301894.emf ratio : [Real](#_aef4bcae5ebc35dd9653214547b3e3cc)

The multiplier by which to multiply the unit to convert to the base unit as defined within the same model.

958301894.emf symbol : [String](#_e8a6ce315d976318da3ab784a645ea44)

The accepted symbol for a unit. e.g. "g" for "Gram".

Direct Supertypes

* [External Reference](#_57d5acead118af36f52c313850902bc4)
* [Value Type](#_66c012755132097a23386c38f992f6b2)

### Stereotype Value Type

A <<Value Type>> is a type representing an atomic unit of information without independent identity. Values include numbers, strings and enumerations. In some cases values may have internal structure.Quantity kinds and units are also values. Values may stereotype any classifier. UML data types, including primitives and enumerations, are implicitly values.

Base Classes

* [Classifier](#_5f9a87915e1e9718a1a1cc45af995a70)

## SIMF Profile::SIMF Rules Profile

The SIMF rules profile defines the way to model rules and mapping within and between data sources via a conceptual model.

### Diagram SIMF Rules Profile

603307669.emf

**Figure 1 SIMF Rules Profile**

Computation computes a value for the mapping end based on the expression applied to the mapped property or relationship. Where computation is used inverse mapping is not specified - any inverse mapping is implementation specific.

### Stereotype Facade

<<Facade>> defines a classifier as being a view of (facade of) one or more other classifiers. Facades usually define additional properties that match some external view of a conceptual model element. A facade will represent the classifier for which it is a facade. A Facade will use one of two methods to relate the facade properties to the conceptual Model:\* <<Rule>> using the facade.\* Applying the <<computation>> stereotype and Subclassing "Representation Computation"

Base Classes

* [Classifier](#_5f9a87915e1e9718a1a1cc45af995a70)

### Stereotype Map

<<Map>> defines an equality rule between two properties in a <<Rule>> - they must represent the same information, perhaps using different representations.Maps should be drawn <from> the representation <to> the more conceptual. <<Map>> may be used between models, as is common for a <<Mapping Rule>> or within one model to equate different representations for the same thing (e.g., property paths).

Base Classes

* [Connector](#_e427016101ac3f4ed3e1e48318bb5bbf)

Tag Definitions

958301894.emf coerce : [Boolean](#_6119a00b0834641b9fe3f5ae9f58237f)

Where <coerce> has a value of TRUE a map rule will be evaluated even if the <from> is not type compatible with the <to> type.Where <coerce> is FALSE or unstated a map rule will be evaluated only if the <from> is type compatible with the <to> type.Type compatible shall be defined as one of: Being the same type, <from> being a subtype of <to> (as defined by a type generalization rule), <from> being a representation of <to> (as defined by a representation rule).Representation rules applied to a supertype apply to a subtype.

958301894.emf condition : [ValueSpecification](#_a458ec73552c6b366443934fae939b88)

<condition> is an expression that must be true for the map rule to hold.

958301894.emf default : [Boolean](#_6119a00b0834641b9fe3f5ae9f58237f)

<default> is true if the map should be enforced only if no other maps apply.

958301894.emf type : [Classifier](#_5f9a87915e1e9718a1a1cc45af995a70)

<type> is a restriction on the type of a property or relation that a map represents. One "side" of the map connector must have this type.

### Stereotype Mapping Rule

<<Mapping Rule>> defines a pattern structure described by a structured classifier that shows how both "sides" of a representation (conceptual and logical) are related. Each "side" must match, including any traversals through structures defined with properties and connectors. Such traversals are links which may also have filters to more precisely define the pattern. The pattern is described using structured classifier properties and connectors.The mapping engine ensures that the patterns match, bidirectionally.

Base Classes

* [StructuredClassifier](#_b9e9bf5c5a181db8188a08fcb1e762bf)

Direct Supertypes

* [Rule](#_f91bc6f19e52cdd1d12a777a1f2c8c68)

### Stereotype Match

Match specifies an element in a structure that must match a model element for the pattern to match. The match is the starting point for the pattern from which all paths are computed.<<Match> is a shortcut for <<Pattern Element>> strength=Match

Base Classes

* [Connector](#_e427016101ac3f4ed3e1e48318bb5bbf)
* [ConnectorEnd](#_a78de45094d48ce6b529e91af177da33)
* [Property](#_9624701f3c3fcd329e6dba1fc74db836)

Direct Supertypes

* [Pattern Element](#_2ab083e7636c6f2e537ce538d10aef76)

### Stereotype Pattern Element

<<Pattern element>> further defines a connector, connector end or property within a pattern based on the tag values.Note that the UML default value may be used to set the initial value of a pattern element.

Base Classes

* [Connector](#_e427016101ac3f4ed3e1e48318bb5bbf)
* [ConnectorEnd](#_a78de45094d48ce6b529e91af177da33)
* [Property](#_9624701f3c3fcd329e6dba1fc74db836)

Tag Definitions

958301894.emf computation : [ValueSpecification](#_a458ec73552c6b366443934fae939b88)

<computation> computes a value for the pattern element based on the expression.Where computation is used inverse mapping is not specified - any inverse mapping is implementation specific.

958301894.emf condition : [ValueSpecification](#_a458ec73552c6b366443934fae939b88)

<condition> states a condition that must be true within the scope of the pattern element. This can be used for pattern matching, setting values or restriction of paths.

958301894.emf explicit : [Boolean](#_6119a00b0834641b9fe3f5ae9f58237f) [0..1]

If <explicit> is true, the pattern element must be explicitly asserted as the indicted type, not derived or inferred from a supertype or super property.

958301894.emf quantifier : [Quantifier](#_d25a58e40827d1a1b5e534e6ec96ec9e)

A property that defines a quantification within a pattern. The quantifier defines the set of things that will populate the pattern property for all instances of the pattern.Quantifiers operate over the type of a pattern element and define a set or subset that corresponds to the extend of the pattern elements type.e.g. for all people p: People is the context and P is the quantified property. In SIMF the quantified property would typically be named <quantifier> <type>. So the above quantified property would be named "all people". The quantified property will be asserted to have the quantified type.

958301894.emf strength : [Pattern Element Strength](#_8c74a7b021860f73d95790003eef7ac0)

<strength> defines the behavior of an element with respect to a pattern - how it impacts the selection, evaluation or assertion of the pattern.

958301894.emf type : [Classifier](#_5f9a87915e1e9718a1a1cc45af995a70)

<type> is a restriction on the type of a property or relation that a pattern element represents.

### Enumeration Pattern Element Strength

Pattern Element Strength defines a set of options for the mapping behavior of a pattern element.

Literals:

1297872297.emf Assert

The element will be asserted as required for a valid pattern. Assert is the default.

1297872297.emf Default

The element will be asserted only if no other values are asserted within the pattern or as pre-existing assertions.

1297872297.emf Exists

Existing element that will be used to compute other values but does not otherwise impact the pattern.

1297872297.emf Match

Match is used in query and mapping patterns, all elements of the classified type that match the pattern are selected as instances of the pattern.Match may be considered a qualified "All". Match does not assert the existence of something, it determines the existence of a pattern match such that other assertions may be made.Relationships between properties with <quantifier>=Match must hold between the matched properties for the pattern to match.

### Enumeration Quantifier

The set of quantifiers for pattern variables. Quantifiers operate over the type of a pattern element and define a set or subset that corresponds to the extend of the pattern elements type.

Literals:

1297872297.emf All

The universal quantifier - the quantified property is a stand-in for all elements of the extent of the quantified type

1297872297.emf Exactly One

The existential quantifier limited to exactly one of a potentially larger set

1297872297.emf Most

A stratified existential quantifier with a default for a "typical" value - example: <Most> people have 2 arms.For logics that do not support "most", most may be interpreted as "There Exists".

1297872297.emf None

A quantifier where no instance of the type may fill the role. E.g. "there may not exist".

1297872297.emf Some

A stratified existential quantifier for a common values - example: <Some> people like computers.For logics that do not support "some", some may be interpreted as "There Exists".

1297872297.emf There Exists

The existential quantifier - at least one element must exist.

### Stereotype Represents

<<Represents>> is an assertion that the source type or feature provides a more concrete way to represent the target type or feature. Represents may be used within conceptual models or from a physical model to a conceptual model.• A representation that is a dependency or realization makes no assumption that the types are substitutable.• A representation that is a generalization is substitutable for what it represents.

Base Classes

* [Dependency](#_c8808432037fd32b0da9b016aeac1fa5)
* [Generalization](#_3596249272c463c4fc390555f5203d9e)

Tag Definitions

958301894.emf condition : [ValueSpecification](#_a458ec73552c6b366443934fae939b88)

<condition> is an expression that must be true for the source to represent the target.

958301894.emf map-all : [Boolean](#_6119a00b0834641b9fe3f5ae9f58237f)

<map-all> implies a direct mapping between instances of the types in both directions.<map all> is equivalent to a mapping with a rule mapping properties of each type but is lower precedence than other mappings - if types have a more specific map it will apply first.

### Stereotype Rule

<<Rule>> defines a pattern that must hold true for the context of the rule.The pattern is described using structured classifier properties and connectors.A rule is a pattern structure described by a structured classifier that shows how elements are related. Each mapped element must match, including any traversals through structures defined with properties and connectors. Such traversals are links which may also have filters to more precisely define the pattern. The mapping engine ensures that the patterns match, bidirectionally.

Base Classes

* [StructuredClassifier](#_b9e9bf5c5a181db8188a08fcb1e762bf)

Tag Definitions

958301894.emf holds within : [Namespace](#_4ccbd6889d668da95205b752d8628552)

<holds within> is the context in which a rule is asserted (required to be true). Anything contextualized by the context is subject to the proposition.If not stated the rule is asserted by its owner.

### Stereotype Rule Model

A <<Rule Model>> defines a package as containing rule specifications and asserts those rules to be true.

Base Classes

* [Package](#_657662b69a91b4b82844aca2e3465cd3)

Direct Supertypes

* [Model](#_86d8bba2473f751aaa57d08526f58037)

### Stereotype Subset of

In a pattern or mapping rule, <Subset of> defines a pattern property that represents a subset of another property. The subset may be constrained by a more specific type, expressions, values or required cardinalities.Subset stereotypes the end of a connector that is the superset.

Base Classes

* [ConnectorEnd](#_a78de45094d48ce6b529e91af177da33)

Tag Definitions

958301894.emf default : [Boolean](#_6119a00b0834641b9fe3f5ae9f58237f) [0..1]

True if the subset should be populated only if no other subsets have been populated.

### Stereotype Subsumes

<<Subsumes>> is a dependency between rules. When a rule subsumes another the subsumed rule will not apply (fire). if the <subsumed by> rules applies (fires).Where rules are also patterns, a rule may specialize another which will subsume the specialized rule as well as include the generalized rule parts as parts of the specialized rule.

Base Classes

* [Dependency](#_c8808432037fd32b0da9b016aeac1fa5)