ODRL Version 2.1 Core Model

Final Specification: 5 March 2015

This Version: http://www.w3.org/community/odrl/model/2.1/

Latest Version: http://www.w3.org/community/odrl/model/2/

Editors:

Renato Iannella, Semantic Identity, ri@semanticidentity.com
Susanne Guth, Vodafone, susanne.guth@vodafone.com
Daniel Paehler, University of Koblenz, tulkas@uni-koblenz.de
Andreas Kasten, University of Koblenz, andreas.kasten@uni-koblenz.de

Status of this document

This specification was published by the W3C ODRL Community Group. It is not a W3C Standard nor is it on the W3C Standards Track. Please note that under the W3C Community Contributor License Agreement (CLA) there is a limited opt-out and other conditions apply. Learn more about W3C Community and Business Groups.

The W3C ODRL Community Group publishes a Final Specification to indicate that the document is believed to be mature and stable for implementation by the wider community. This Final Specification is now endorsed by the W3C ODRL Community Group as appropriate for widespread deployment and that promotes the Community Groups's mission.

Discussion and feedback of this document takes place on the W3C ODRL Community Group mailing list (see Contributor Archive). Feedback from the public is encouraged and can be send to public-odrl@w3.org (see Public Archive).

Table of contents

1. Overview

The W3C ODRL Community Group's aim is to develop and promote an open international specification for Policy Language expressions. The ODRL Policy Language provides a flexible and interoperable information model to support transparent and innovative use of digital assets in the publishing, distribution and consumption of content, applications, and services across all sectors and communities. The ODRL Policy model is targeted to support the business models of open, educational, government, and commercial communities through Profiles that enhance the model to align to their requirements whilst providing a common semantic layer for interoperability.

ODRL Version 2.1 is a major update for ODRL and supersedes Version 2.0 and Version 1.1.[ODRL11]

The ODRL Core Model is designed to be independent from implementation mechanisms and is focussed on the optimal interoperable model and semantics to represent policy-based information.

The following documents are part of the ODRL Version 2 series:

- ODRL V2 – Requirements [ODRL-REQ]. The Requirements document represents requirements for the language that have been gathered since ODRL Version 1.1 has been released. Not all requirements are aimed to be met.
- ODRL V2.1 – Core Model *(this document)*
- ODRL V2.1 – Common Vocabulary. This document specifies the terms (vocabulary) used by the Core Model for policy expression needs across communities [ODRL-VOCAB]. (This was called the "data dictionary" previously.)
- ODRL V2.1 – XML Encoding. The XML Encoding document specifies the serialisation of the Core Model in XML [ODRL-XML].
- ODRL V2.1 – Ontology. The ODRL Ontology document will specify the serialisation of the Core Model in the W3C RDF/OWL Semantic Web languages [ODRL-ONTO].
- ODRL V2.1 – JSON Encoding. The JSON Encoding document specifies the serialisation of the Core Model in JSON [ODRL-JSON].

The requirements for Version 2 are documented [ODRL-REQ] and will be directly referenced in this document to ensure that they have been adequately addressed (where applicable).

The model shall be formally specified using UML notation [UML] [ODRL-REQ-6] and shall utilise the key words "MUST", "MAY", "REQUIRED", and "OPTIONAL" in accordance to [RFC2119].

To make keywords easily distinguishable, this document uses syntax highlighting (formatted coded typeface) to indicate Core Model entities, the entities' attributes, and concrete values for the attributes.

2. ODRL Core Model

The basic context of an ODRL Policy is that only an explicitly permitted use may be executed. Any use not explicitly permitted is prohibited by default. Based on that context, an ODRL Policy must contain at least one Permission and may contain Prohibitions. An ODRL Policy only permits the action explicitly specified in a Permission and all other actions are implicitly prohibited. An action defined in a Prohibition should only refine (or directly relate to) the semantics of an action defined in one of the Permissions in the ODRL Policy.

For example, an ODRL Policy that has the action "present" Permission and may also have the action "print" Prohibition (as these actions are related hierarchically in the ODRL Common Vocabulary).

Note that ODRL Profiles can be developed and used to refine the basic context of an ODRL Policy. Hence, the application of an ODRL Profile must be understood by the consuming community and systems.

Figure 2.1 below shows the ODRL Core Model. The `Policy` is the central entity that holds an ODRL policy together.

A `Permission` *allows* a particular `Action` to be executed on a related `Asset`, e.g. "play the audio file abc.mp3". A `Constraint` like "at most 10 times" might be added to specify the `Permission` more precisely. The `Party` that grants this `Permission` is linked to it with the `Role` `assigner`, the `Party` that is granted the `Permission` is linked to it with the `Role` `assignee`, e.g. "`assigner` VirtualMusicShop grants the `Permission` to `assignee` Alice". Additionally, a `Permission` MAY be linked to `Duty` entities.

Similar to `Permissions`, a `Duty` states that a certain `Action` MAY be executed by the `Party` with the `Role` `assignee` for the `Permission` to be valid, e.g. "Alice must pay 5 EUR in order to get the `Permission` to play abc.mp3".

The `Prohibition` entity is used in the same way as `Permission`, with the two differences that it does not refer to `Duties` and (obviously) that it *forbids* the `Action`, e.g. "Alice is forbidden to use abc.mp3 commercially".

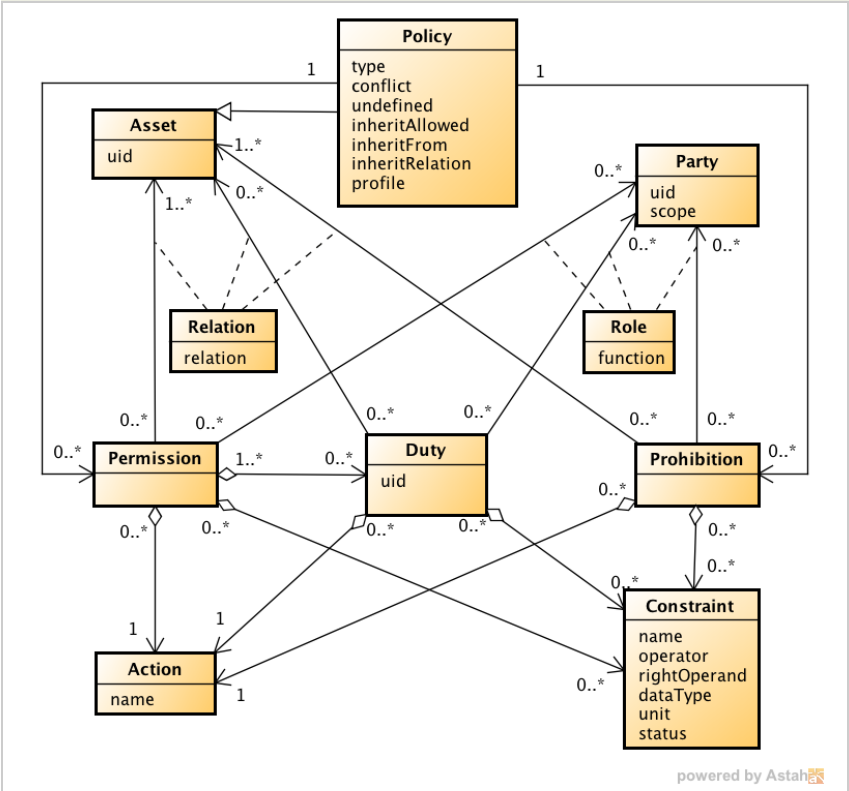

Figure 2.1 – ODRL Core Model Version 2.1

The following sections describes each entity of the Core Model in greater detail.

2.1 Policy

The `Policy` entity is the top-level entity and contains the following attributes:

- `uid`: the unique identification of the `Policy` entity (REQUIRED)
- `type`: indicates the semantics of the `Policy` entity (REQUIRED). These are further described in the Common Vocabulary and ODRL Profiles.
- `conflict`: indicates the precedence between `Permissions` and `Prohibitions` (OPTIONAL)
- `undefined`: indicates how to handle undefined `Actions` (OPTIONAL)
- `inheritAllowed`: indicates if the `Policy` entity can be inherited (OPTIONAL)
- `inheritFrom`: the identifier from which this `Policy` inherits from it's parent `Policy` (OPTIONAL)
- `inheritRelation`: the identifier for the relationship type of this inheritance structure (OPTIONAL)
- `profile`: the identifier of the ODRL Profile that this `Policy` conforms to (OPTIONAL)

The `uid` attribute MUST be a unique identifier.

The range of values for the `Policy` entity's `type` attribute will be described in the Common Vocabulary document or in community profiles. This value MAY also impose further constraints on the Core Model, such as are exemplified in the Scenarios for types `Offer` and `Agreement`. It is important that the `type` attribute be clearly understood in policy expressions as the semantics MAY impose restrictions on the expression language constructs such as cardinalities between entities.

The `conflict` attribute is used to resolve conflicts arising from the merging of policies, specifically when there are conflicting `Actions` in the `Permissions` and `Prohibitions`. If present, the `conflict` attribute MUST take one of the following values:

- `perm`: the `Permissions` will always takes precedence
- `prohibit`: the `Prohibitions` will always takes precedence
- `invalid`: the policy is not valid

If the `conflict` attribute is not explicitly set, its default value will be used instead. The default value of the `conflict` attribute is `invalid`.

The `undefined` attribute is used to indicate how to support `Actions` that are not part of any profile in the policy expression system. If present, the `undefined` attribute MUST take one of
the following values:

- `support`: the `Action` is to be supported as part of the policy – and the policy remains valid
- `ignore`: the `Action` is to be ignored and not part of the policy – and the policy remains valid
- `invalid`: the `Action` is unknown – and the policy is invalid

In the `support` case, even though the `Action` is unknown, the policy still is valid and the consuming parties or system of the policy MUST be made aware of the unknown `Action`. This MAY be via a user interface that displays the unknown `Action` for human readability.

In the `ignore` case, even though the `Action` is unknown, the policy still is valid and the consuming parties or system of the policy MAY be made aware of the unknown `Action`.

In the `invalid` case with the unknown `Action`, the policy is invalid and the consuming parties or system of the policy MUST be made aware of this.

If the `undefined` attribute is not explicitly set, its default value will be used instead. The default value of the `undefined` attribute is `invalid`.

Other attributes MAY be added to the `Policy` entity to support additional functions and requirements. Typically, these will be from different community vocabularies. For example, to indicate the issued date or valid dates of the `Policy` entity, use of the Dublin Core Metadata Terms would be recommended.

2.1.1 Inheritance

The `inheritAllowed` attribute in the `Policy` entity is used to indicate if the `Policy` expression can be used in any inheritance relationship [ODRL-REQ-1.20]. If present, the value of the `inheritAllowed` attribute MUST take one of the following values:

- `true`: the `Policy` expression can be used for inheritance
- `false`: the `Policy` expression can not be used for inheritance

If the `inheritAllowed` attribute is not explicitly set, its default value will be used instead. The default value of the `inherit` attribute is `true`.

Only if the `inheritAllowed` attribute has the value `true` can the `inheritFrom` and `inheritRelation` attributes be specified.

The `inheritFrom` attribute in the (child) `Policy` will uniquely identify (via a UID) the (parent) `Policy` from which the inheritance will be performed.

The `inheritRelation` attribute in the (child) `Policy` will uniquely identify (via a UID) the type of inheritance from the (parent) `Policy`. For example, this may indicate the business scenario, such as *subscription*, or prior arrangements between the parties (that are not machine representable). Such terms MAY be defined in the Common Vocabulary or Community Profiles. For example, an Assigner and Assignee may have a historical arrangement related to the specific use of content they make available to each other. The business model (identified with a URI) is used in the `inheritRelation` attribute in their subsequent ODRL policies they exchange. This will require the ODRL policy to be interpreted with the additional information identified by the URI. For example, this may include additional permission actions or constraints (etc) that is documented in their business model arrangement.

Both the `inheritFrom` and `inheritRelation` attribute MAY be used independently.

The following restrictions apply when using inheritance:

- Single inheritance is only supported. (One Parent `Policy` to one or more Child `Policy` entities. No Child `Policy` can inherit from two or more Parent `Policy` entities.)
- Inheritance can be to any depth. (Multiple levels of Children `Policy` entities.)
- Inheritance cannot be circular.
- The Child `Policy` MUST override the Parent `Policy`. i.e.: If the same `Action` appears in the Parent, then it is replaced by the Child version, otherwise the Parent `Actions` are added to the Child's `Actions`.
- No state information is transferred from the policy in the Parent `Policy` to the Child `Policy`

2.1.2 Profile

The `profile` attribute in the `Policy` entity is used to indicate the identifier (URI) of the ODRL Profile for which the policy expression conforms to. This attribute is OPTIONAL, but if the attribute appears, then any consuming system MUST understand the identified ODRL Profile – and all the rules from the Profile MUST apply to the policy expression. If multiple ODRL Profiles are required, then it is recommended that a new identifier be created to identify the combination of Profiles (and document the combined Profiles).

Since the ODRL Core Model and Common Vocabulary [ODRL-VOCAB] represents broad needs for policy expressibility, different communities will require less or more concepts from the Core Model and terms from the Common Vocabulary. Community profiles of the ODRL model are expected to be developed that adequately document these requirements in respect to the Core Model and Vocabulary. Some requirements for communities developing ODRL Profiles include:

- Document any additions to the Core Model
- Document any aspects of the Core Model that will have different default values or interpretations
- Document which aspects of the Core Model are not being used (deprecated)
- Document new Vocabulary terms
- Document deprecated Vocabulary terms
- Declare the ODRL Profile identifier (URI)
- Declare your communities namespace URI (see the various Encoding specifications)
- Share the Community ODRL Profile with the W3C ODRL community for feedback and comments

It is recommended that the ODRL Profile URI be the same as the Namespace URI, but this is not mandatory.

2.2 Asset

The `Asset` entity is the subject of an ODRL policy expression that permissions and prohibitions are applied to. The Asset entity can be any form of identifiable resource, such as data/information, content/media, applications, or services. Furthermore, it can be used to represent other `Asset` entities that are needed to undertake the Policy expression, such as with the `Duty` entity. The `Asset` entity is referred to by the `Permission` and/or `Prohibition` entities, and also by the `Duty` entity.

The `Asset` entity contains the following attribute:

- `uid`: the unique identification of the `Asset` (REQUIRED)

The identification of the `Asset` entity is a key foundation of the ODRL Policy language. However, there are some use cases where the ODRL Policy expression MAY be embedded inside the target `Asset`. In these cases, it MAY be more appropriate to provide, or infer, a link to the `Asset` entity (as the complete `Asset` `uid` may not be known at the time) through the local context. Use of such inference and context MUST be documented in the relevant ODRL community Profile.

Since ODRL policies could deal with any kind of asset, the ODRL Core Model does not provide additional metadata to describe `Asset` entities of particular media types. It is recommended to use already existing metadata standards, such as Dublin Core Metadata Terms that are appropriate to the `Asset` type or purpose.

The `Relation` entity is used to associate the `Asset` entity with the relevant `Permission`, `Prohibition`, and `Duty` entities

2.2.1 Relation

The `Relation` entity is an association class and can be used to link to an `Asset` from either `Permission`, `Duty` or `Prohibition`, indicating how the `Asset` MAY be utilised in respect to the entity that links to it.

The `Relation` entity contains the following attribute:

- `relation`: indicates the relationship of the `Asset` to the linked entity (REQUIRED)

The default value for the `relation` attribute is `target` which indicates that the `Asset` is the primary object to which the `Permission`, `Duty` or `Prohibition` actions apply.

Other values for the `Relation` entity MAY be defined in the Common Vocabulary and community Profiles.

2.3 Party

The `Party` entity is the object of an ODRL policy that performs (or not performs) actions or has a role in a Duty [ODRL-REQ-1.5]. The `Party` entity can be any form of identifiable entity, such as a person, group of people, organisation, or agent. An agent is a person or thing that takes an active role or produces a specified effect.

The `Party` entity contains the following attribute:

- `uid`: the unique identification of the party (REQUIRED)
- `scope`: defines how the role shall be interpreted under different contexts. (OPTIONAL)

The ODRL Core Model does not provide additional metadata for the `Party` element. It is recommended to use already existing metadata standards, such as IETF vCard that are appropriate to the `Party` type or purpose.

The `Role` entity is used to associate the `Party` entity with the relevant `Permission`, `Prohibition`, and `Duty` entities.

2.3.1 Role

The `Role` entity is an association class and can be used to link to a `Party` from either `Permission`, `Duty` or `Prohibition`, indicating which role the `Party` takes with respect to the entity that links to it.

The `Role` entity contains the following attributes:

- `function`: the functional role the `Party` takes (REQUIRED)

The `function` attribute MUST support the following values:

- `assigner`: indicates that the `Party` has assigned the associated `Permission`, `Duty`, or `Prohibition`. In other words, the `Party` grants a `Permission` or requires a `Duty` to be performed or states a `Prohibition`.
- `assignee`: indicates that the `Party` has been assigned the associated entity, i.e. they are granted a `Permission` or required to perform a `Duty` or have to adhere to a `Prohibition`.

Other values for the `function` attribute MAY be defined in the Common Vocabulary and community Profiles.

The scope attribute MAY be used to indicate the context under which to interpret the Party entity. If present, the scope attribute MAY take one of the following values:

- individual: indicates that the Party entity is a single individual. The linked Permission, Duty or Prohibition is applicable for that individual only.
- group: indicates that the Party entity represents a group. The group consisting of many individual members. The linked Permission, Duty or Prohibition is applicable for each member of that group. For example, a (constrained) Permission to play a movie 5 times is valid for each Party member or the Duty to pay 3 EUR has to be fulfilled by each Party member.

Other values for the scope attribute MAY be defined in the Common Vocabulary and community Profiles.

2.4 Permission

The Permission entity indicates the Actions that the assignee is permitted to perform on the associated Asset. In other words, what the assigner (supplier) has granted to the assignee (consumer).

An ODRL policy expression MAY contain at least one Permission. It is important to verify the semantics of the Policy type attribute as this MAY indicate additional constraints on the Policy expression structure.

If several Permission entities are referred to by a Policy, then *all of them* are valid.

The Permission entity has the following relations:

- Asset: the Permission entity MUST refer to an Asset (where at least one, and only one, relation value is target) on which the linked Action MAY be performed (REQUIRED)
- Action: the Permission entity MUST refer to *exactly one* Action that indicates the granted operation on the target Asset (REQUIRED)
- Party: the Permission MUST refer to one or more Party entities linked via the Role entity (see Section 2.3.1) (OPTIONAL)
- Constraint: the Permission MAY refer to one or more Constraints which affect the validity of the Permission, e.g. if the Action play is only permitted for a certain period of time (OPTIONAL)
- Duty: the Permission MAY refer to one or more Duty entities that indicate a requirement that MAY be fulfilled in return for receiving the Permission (OPTIONAL)

2.5 Duty

The Duty entity indicates a requirement that MUST be fulfilled in return for being entitled to the referring Permission entity [ODRL-REQ-1.8]. While implying different semantics, the Duty entity is similar to Permission in that it is an Action that can be undertaken. If a Permission refers to several Duty entities, *all of them* have to be fulfilled for the Permission to become valid. If several Permission entities refer to one Duty, then the Duty only has to be fulfilled *once* for all the Permission entities to become valid.

The Duty entity contains the following attributes:

- uid: a unique identification for this Duty. Used to refer a single Duty to multiple Permission entities (OPTIONAL)

The Duty entity has the following relations:

- Action: indicates the operation that MUST be performed (REQUIRED). Note: It is assumed that the assigned Party has the appropriate permissions to perform this action.
- Party: a Duty MAY refer to Party entities with different Roles (see Section 2.3.1). If no explicit Party is linked to as assignee or assigner, the Parties with the respective Roles are taken from the referring Permission. (OPTIONAL)
- Asset: a Duty entity MAY refer to an Asset (where at least one, and only one, relation value is target) related to fulfilling the Duty.
  For example, a nextPolicy Action must be linked to the identifier of a target policy Asset. (OPTIONAL)
- Constraint: a Duty MAY link to one or more Constraints [ODRL-REQ-1.10] (OPTIONAL)

A Duty entity does not, by itself, specify any conditions on when the Duty Action MUST or MAY be performed, such as to compensate before viewing the movie. Such conditions MAY be expressed through Constraint entities.

To support cases where the Duty MAY be performed for each Action on an Asset (for example, pay-per-view) then the use of a Constraint (e.g. count=1) on the Permission (e.g. play) can express these semantics.

2.6 Prohibition

The Prohibition entity indicates the Actions that the assignee is prohibited to perform on the related Asset [ODRL-REQ-1.7]. Prohibitions are issued by the supplier of the Asset – the Party with the Role assigner. If several Prohibition entities are referred to by a Policy, *all of them* must be satisfied.

The Prohibition entity has the following relations:

- Asset: the Prohibition entity MUST refer to an Asset (where at least one, and only one, relation value is target) on which the Action is prohibited (REQUIRED)
- Action: the Prohibition entity MUST refer to *exactly one* Action that is prohibited (REQUIRED)
- Party: the Prohibition MAY refer to one or more Party entities linked via the Role entity (see Section 2.3.1) (OPTIONAL)
- Constraint: the Prohibition MAY refer to one or more Constraint entities (OPTIONAL)

2.7 Action

The Action entity (when related to a Permission entity) indicates the operations (e.g. play, copy, etc.) that the assignee (i.e. the consumer) is *permitted* to perform on the related Asset linked to by Permission. When related to a Prohibition, the Action entity indicates the operations that the assignee (again the consumer) is *prohibited* to perform on the Asset linked to by Prohibition. Analogously, when related to a Duty, it indicates the operation to be performed.

Action contains the following attribute:

- name: indicates the Action entity term (REQUIRED)

As its value, the name attribute MAY take one of a set of Action *names* which are formally defined in profiles. The ODRL Common Vocabulary defines a standard set of potential terms that MAY be used. Communities will develop new (or extend existing) profiles to capture additional and refined semantics.

2.8 Constraint

The `Constraint` entity indicates limits and restrictions to the `Permission`, the `Prohibition` and the `Duty` entity [ODRL-REQ-1.9]. `Constraints` express mathematical terms with two operands and one operator. For example, the "number of usages" (`name`) must be "smaller than" (`operator`) the "number 10" (`rightOperand`).

If multiple `Constraint` entities are linked to the same `Permission`, `Prohibition`, or `Duty` entity, then all of the `Constraint` entities MUST be satisfied. That is, all the `Constraint` entities are (boolean) *and*ed. In the case where the same `Constraint` is repeated, then these MUST be represented as a single `Constraint` entity using an appropriate `operator` value (for example, `isAnyOf`).

The `Constraint` entity contains the following attributes:

- `name`: a name that identifies the the left operand of the operation (REQUIRED)
- `operator`: an operator function (REQUIRED)
- `rightOperand`: the right operand of the operation (REQUIRED)
- `dataType`: the datatype of the rightOperand (OPTIONAL)
- `unit`: the units of the rightOperand (OPTIONAL)
- `status`: the current value of the left operand (OPTIONAL)

The `name` identifies the left operand of the mathematical operation for the `Constraint` such as "Number of Usages" and "Expiration Date" etc. The `operator` identifies the comparative operation such as "greater than" or "equal to". The `rightOperand` identifies the value that is being compared. The `dataType` indicates the type of the `rightOperand`, such as "decimal" or "datetime" and the `unit` indicates the unit value of the `rightOperand`, such as "EU dollars".

When processing policy expressions, these `Constraint` names MAY be directly linked to a procedure that can determine the outcome of the operations, such as the number of already performed usages and the current date. The `name` and `operator` are defined in the ODRL Common Vocabulary or community profiles.

The `status` provides the current value of the `Constraint` variable (i.e. current value of `name`) [ODRL-REQ-1.3]. This is useful in cases where the current status of `Constraints` needs to be captured and expressed in the ODRL Core Model.

3. Scenarios (Informative)

This section shows a number of policy expression scenarios. In these examples, the different policy expression `types` that are used are for illustrative purposes only and are not part of this normative specification. Also, the specific `Action` and `Constraint` names (etc.) used in these examples are for illustrative purposes only. Please note that formal policy expression `types` and other entities are defined in the ODRL Common Vocabulary specification, or in community profiles.

3.1 Set

The following shows an instance of a `set Policy`. The `Set` shows a policy expression, stating that the `Asset http//example.com/asset:9898` is the `target` of the `Permission reproduce` and the `Prohibition` to `modify`. No parties or other elements are involved. This `set` could be used, for example, as a template or an *instant license*.

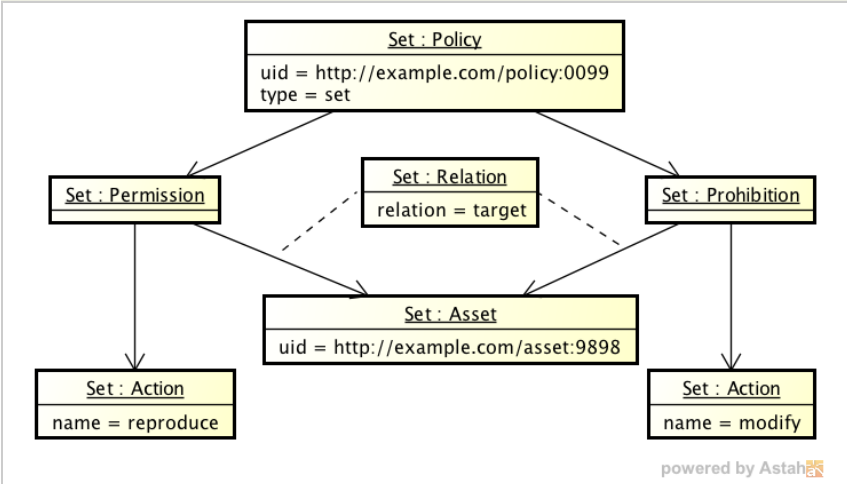

Figure 3.1 – An instance of a Set Policy

3.2 Offer

The following shows the instance of an `offer Policy`. The `offer` contains the music file `http//example.com/music:4545` that is offered by the `Party http//example.com/sony:10` with the `Permissions` to `play` and `copy` the file. The `Permission copy` is only granted once. The two `Permissions` are offered for a payment of AUD$0.50.

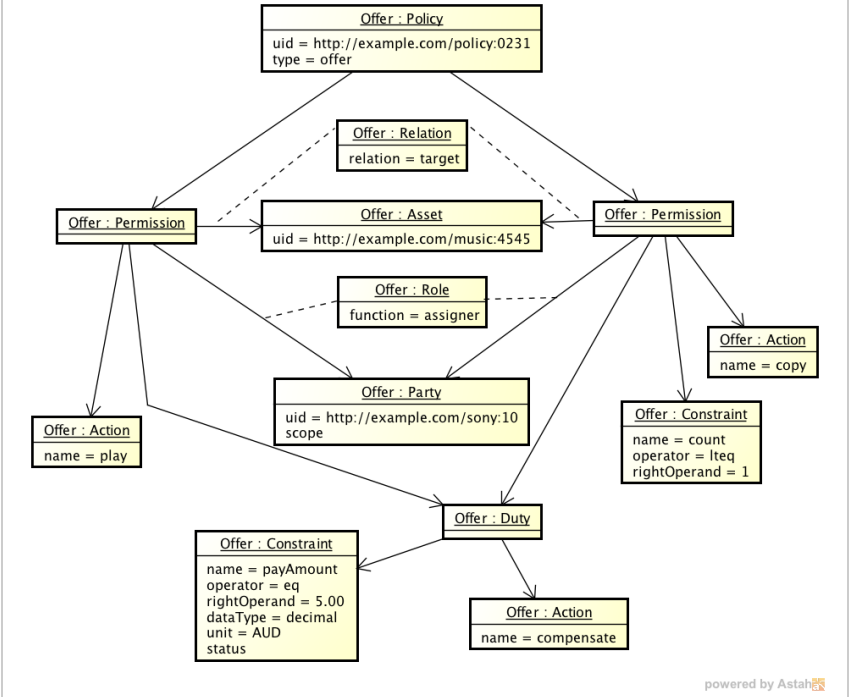

Figure 3.2 – An instance of an Offer Policy

3.3 Agreement

The following shows the instance of an `agreement` `Policy`. The `agreement` contains all entities shown in the `offer` scenario above. A new `Party` element `http//example.com/billie:888` has been added. This `Party` accepted the previous `offer` and thus is now the buyer of the `Permission` `play` and `copy`, i.e. is now linked as `assignee` of the `Permission`s and `Duty` entities.

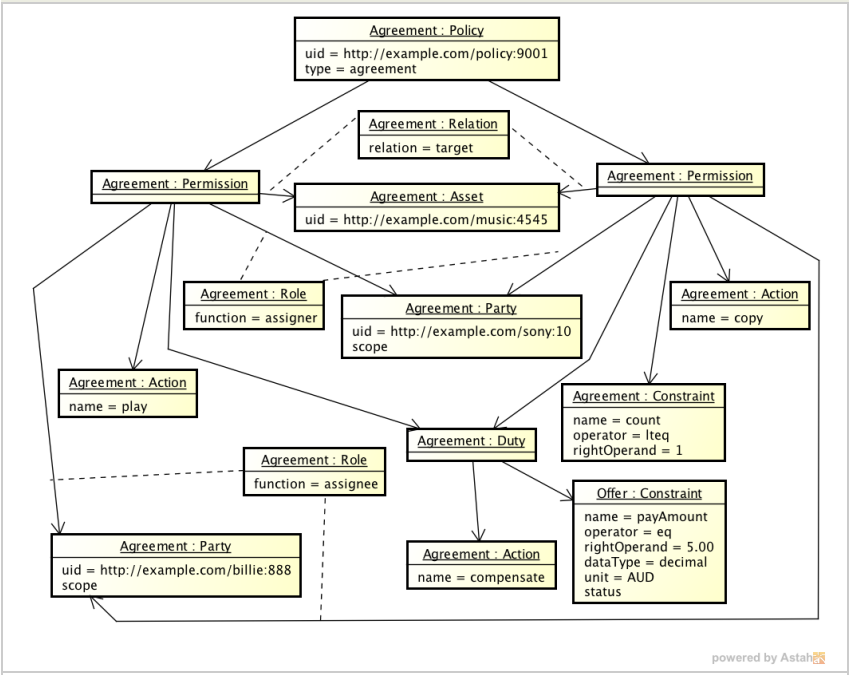

Figure 3.3 – An instance of an Agreement Policy

3.4 Request

The following shows the instance of a `request` `Policy`. The `Party` `http//example.com/guest:0589` has requested the `Permission` to `display` the `target` `Asset` `http//example.com/news:0099`.

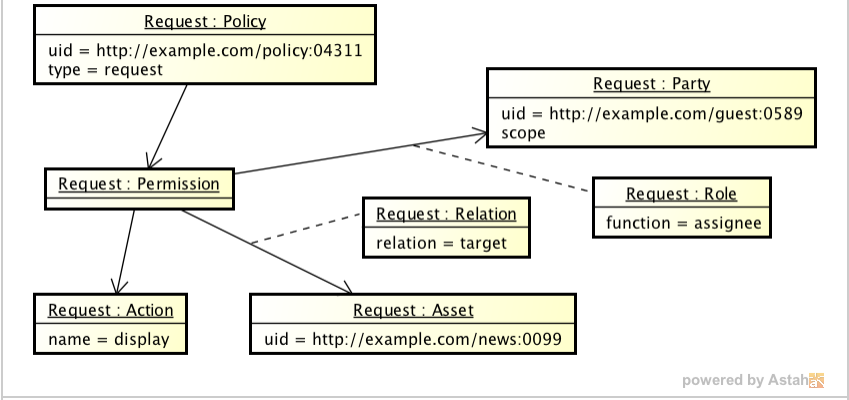

Figure 3.4 – An instance of a Request Policy

3.5 Ticket

The following shows the instance of a `ticket` `Policy`. The `ticket` expresses the `play` `Permission` for the `target` `Asset` `http//example.com/game:4589`. The `Ticket` is valid until the end of the year 2010. Any valid holder of this ticket may exercise this `Permission`.

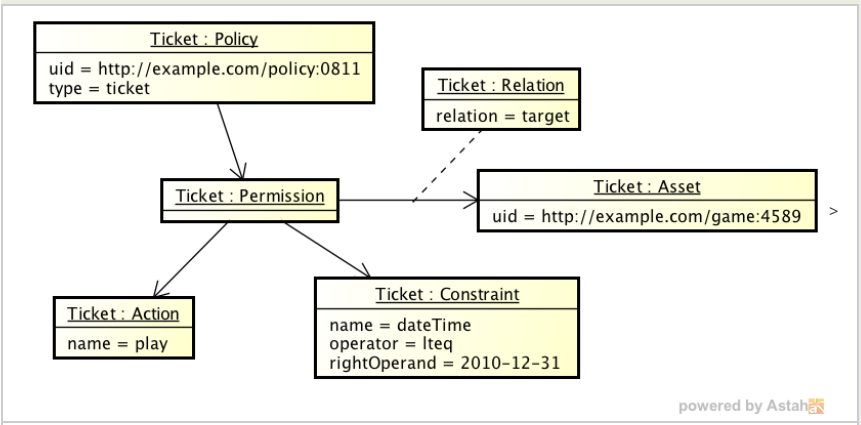

Figure 3.5 – An instance of a Ticket Policy

3.6 Offer and Next Policy

The following shows the instance of an `offer` `Policy` showing the `nextPolicy` structure. The party `http//example.com/sony:99` assigns the `Permission` `distribute` directly to the potential buyer of the permission who will pay $EU1,000. The `distribute` `Permission` is also constrained to the country Italy. The potential `assignee` may then `distribute` the `target` `Asset` according to the `nextPolicy` `target` `Asset` linked directly from this `Duty`. In this case, the next Policy `Asset` stipulates that the potential `assignee` may only offer the `display` `Permission` to downstream consumers.

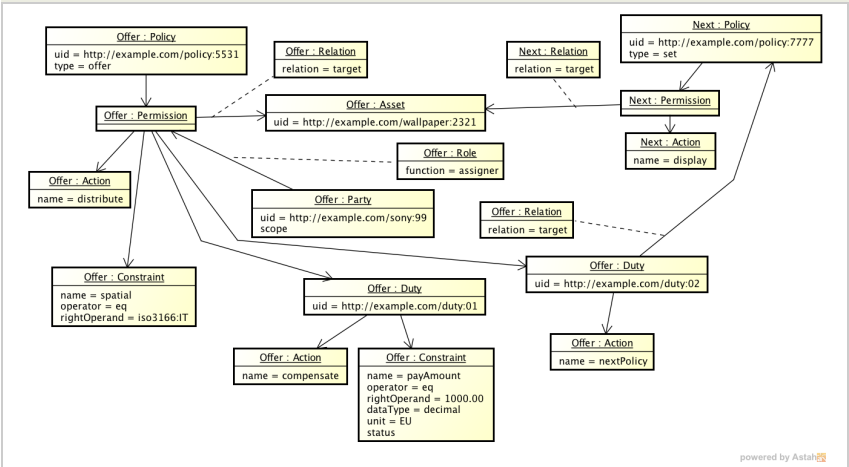

Figure 3.6 – An instance of an Offer and Next Policy Policy

3.7 Privacy

The following shows the instance of an `privacy` `Policy`.

The `target` `Asset` is Personal Data and the `assignee` is allowed to `distribute` the `Asset` only for the `purpose` of contacting the subject of the Personal Data. The purpose value is taken from the P3P privacy purpose vocabulary.

Additionally, the `assigner` (the `Party` who the personal data is about) has stipulated that the `assignee` must `delete` the `Asset` after a 30 day period (retention policy).

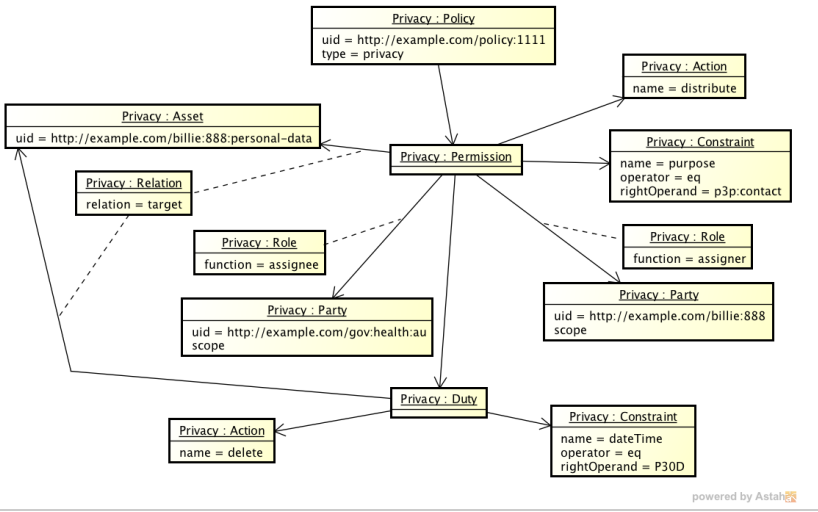

Figure 3.7 – An instance of an Privacy Policy

3.8 Permission and Prohibition

The following shows the instance of an `agreement` `Policy` with both a `Permission` and a `Prohibition`. The party `http//example.com/sony:10` assigns the `Permission` `play` to the `Party` `http//example.com/billie:888` at the same time they are *prohibited* from utilising the `target` `Asset` as a `mobile:ringtone`. Additionally, in case of any conflict, the `conflict` attribute is set to `perm` indicating that the `Permission` entity will take precedence.

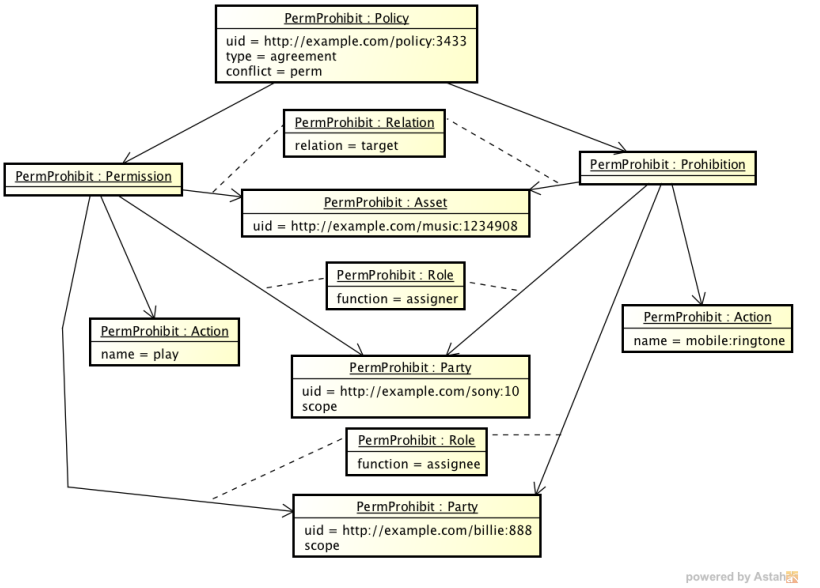

Figure 3.8 – An instance of an Permission and Prohibition Policy

3.9 Inheritance

The following shows the instance of a (child) `Policy` `http//example.com/policy:9999` inheriting from another (parent) `Policy` `http//example.com/policy:5531`. The `inheritFrom` attribute of the (child) `Policy` has the same identifier as the (parent) `Policy`. In this inheritance example, the (parent) `Policy` allows the `Party` `http//example.com/billie:888` to `print` the (parent's) `target` `Asset`. The (child) `Policy` allows the `Party` `http//example.com/class:IT01` (a group of people) to `display` the (child's) `target` `Asset`. Since the (child) `Policy` also inherits from the (parent) `Policy`, then the `Party` `http//example.com/class:IT01` can also `print` the (parent's) `target` `Asset`.

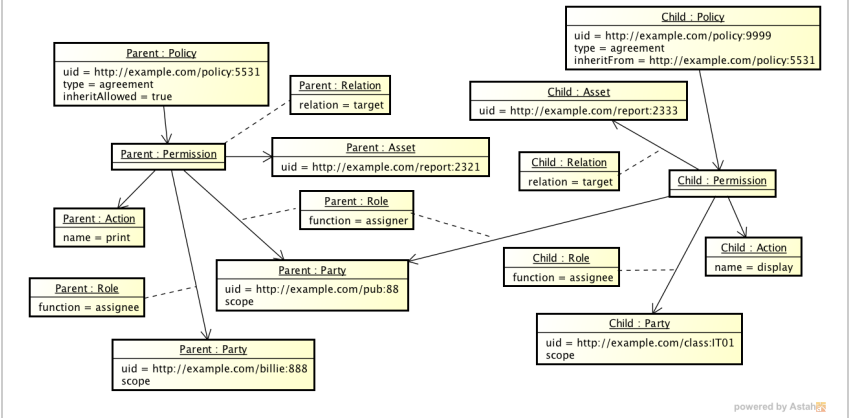

Figure 3.9 – An instance of an Inheritance Policy

3.10 Social Network

The following shows the instance of an `agreement` `Policy` for a Social Network scenario.

The `target` `Asset` are photos posted to a Social Network site and the `assigner` is the owner of the photos. The `assignee` is a `Party` `group` and represents the *football network members* on the social network, who are each allowed to `display` the photos.

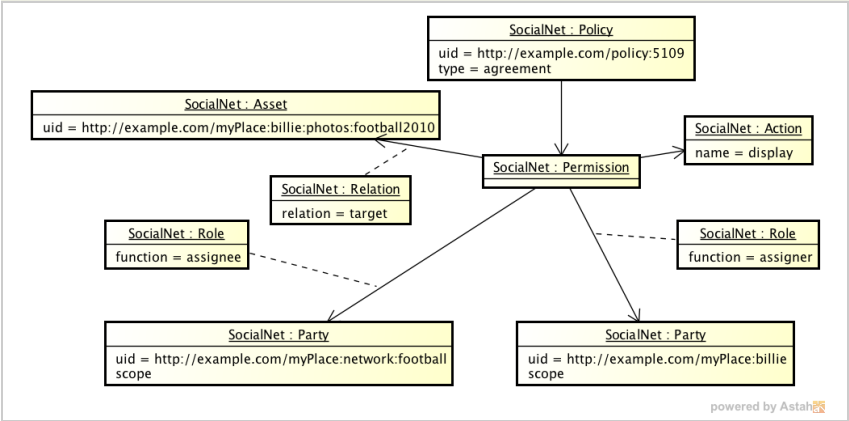

Figure 3.10 – An instance of an Social Network Policy

3.11 Multiple Assets

The following shows an instance of a `set` `Policy` utilising multiple `Asset` entities.

The `index` `Permission` is granted to the `target` `Asset`. As well, the `x:collection` `Asset` specifies which database the `index` outcome should be stored in.

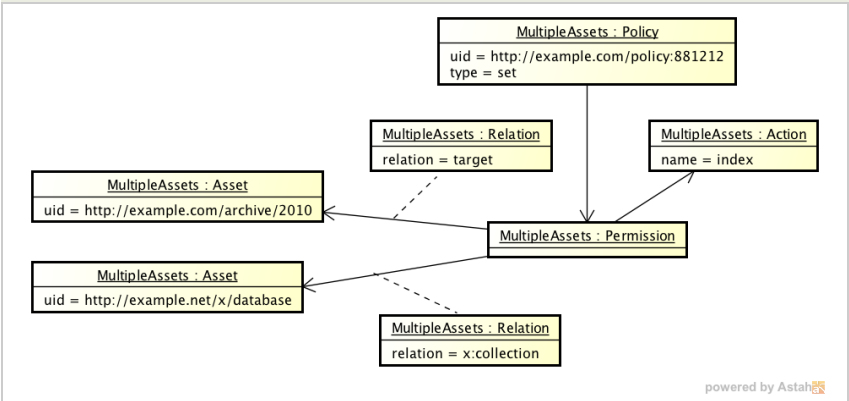

Figure 3.11 – An instance of an Multiple Assets Policy

4. Experimental Features (Informative)

This section contains advanced ODRL features. Although not part of the normative specification, they provide an opportunity for communities to experiment with and provide feedback on experiences that may be included in future ODRL versions.

4.1 Extended Relations

Extended Relations may tie Permission, Prohibition, Duty, and Constraint entities together with an AND, OR or XOR relationship. Only entities of the same type can be linked with this model. For example, a Permission and Prohibition cannot be linked together within this model. The Extended Relations model supports the following attribute:

- operation: may be set with *one* of the mathematical values AND, OR and XOR. (OR is the default if not specified.)

The following table outlines the semantics of Extended Relations with respect to each of the main entity types.

	Permission	Prohibition	Duty	Constraint
OR	The related party may perform *any* (*at least*) one of the Actions	The related party MAY NOT perform *at least* one of the Actions	The related party MUST perform *at least* one of the Actions	The related Permission/Prohibition/Duty is restricted by *at least* one of the Constraints
AND	The related party MUST perform *all* of the Actions	The related party MAY NOT perform *all* of the Actions	The related party MUST perform *all* of the Actions	The related Permission/Prohibition/Duty is restricted by *all* of the Constraints
XOR	The related party MAY perform only *one* of the Actions	The related party MAY NOT perform only *one* of the Actions	The related party MUST perform only *one* of the Actions	The related Permission/Prohibition/Duty is restricted by only *one* of the Constraints.

Note that Extended Relations are not needed to assign two or more Permissions to a Party entity. In this case simply use as many Assignee relations between Party and Permission as needed.

4.2 Abstract Policy Expression

As the Core Model diagram shows (see Figure 4.1), the key Permission, Prohibition and Duty entities are very similar since they have (more or less) the same relationships to the other entities. They core difference is in their semantics:

- Permission says that the assignee *may* do something,
- Duty says that the assignee *should* do something, and
- Prohibition says that they *should not* do it.

In an implementation that interprets ODRL, it may make sense to introduce a common superclass *Rule*, as shown in the (abbreviated) Model in Figure 5.1.

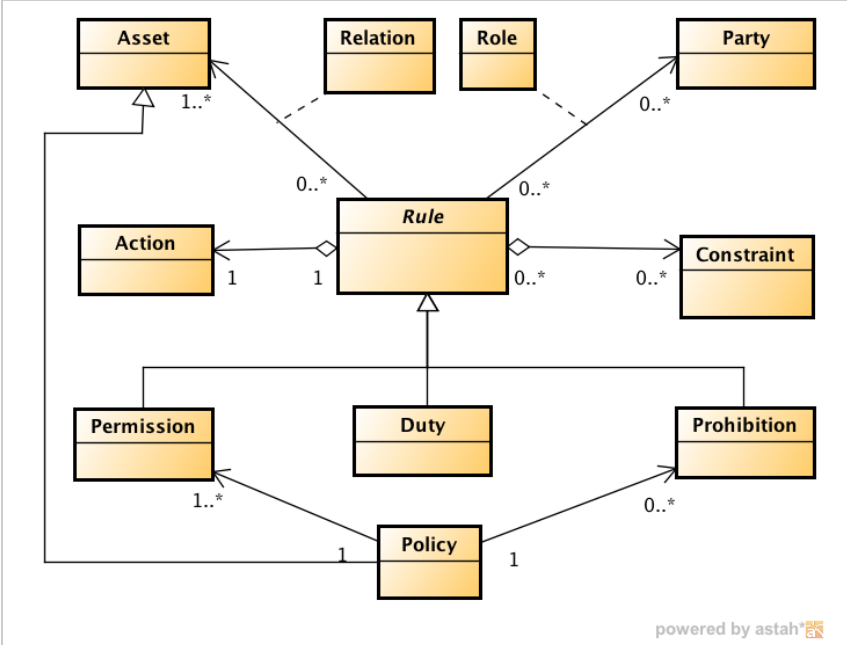

Figure 4.1 – ODRL Abstract Policy Model

By implementing Permission, Prohibition and Duty as subclasses of *Rule*, the redundancy of having very similar, but separately developed classes in an application's source code can be avoided. Furthermore, *Rule* makes it possible to easily extend the Core Model in Profiles by adding policy expressions (as subclasses of *Rule*) that are not possible by default.

4.3 Remedies

In the ODRL Core Model, Duties are only directly related to Permissions, meaning that for a Permission to become effective, the related Duty should be performed. For some use cases though, it might be useful to attach a Duty to a Prohibition, meaning that if a Prohibition is violated, the Duty has to be performed as a kind of remedy or consequence for the violation.

Not only can a Prohibition have a Duty attached to it as a remedy, even Duties themselves may have remedies, e.g. "For the Permission to play audio file xyz to become effective, you have to perform the Duty 'pay 2€'. If you don't perform this Duty (even though you've played yxz), you have to remedy this by performing the Duty 'pay 5€'".

In order to distinguish between a Duty that has to be fulfilled as a requirement and one that has to be fulfilled as a remedy, different relation names are introduced as shown in the Figure 4.2.

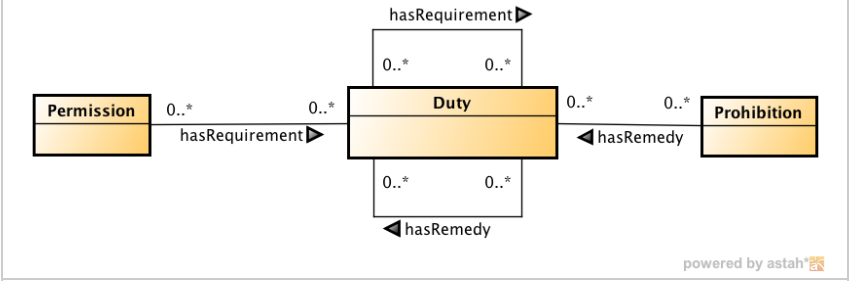

Figure 4.2 – Remedy Model

The relation between `Permission` and `Duty`, which was unnamed before, is now named `hasRequirement`. This is needed not only to make the different semantics clearer, but also because a `Duty` can refer to yet another `Duty` as a requirement, e.g. "If you want to print this written article, you have the `Duty` to attach a particular image of the author, and if you do that, you have the `Duty` to attribute the image to the photographer".

Change History

The major changes from Version 2.0 include:

- Added the `profile` attribute (section 2.1.2)
- Added `dataType` and `unit` to the Constraint entity (section 2.8)
- Added the basic context of ODRL (section 2.0)
- Updated the inheritRelation attribute (section 2.1.1)
- Updated Asset definition (section 2.2)
- Updated Party definition (section 2.3)

Acknowledgements

The editors gratefully acknowledge feedback and contributions to this document from:

- Michael Steidl, Vicky Weissman, Mark Strembeck, Alapan Arnab, Steven Rowat, Eetu Luoma, Jaime Delgado, Ruediger Grimm, Helge Hundacker, Stuart Myles, Francis Cave, Rigo Wenning, Hassan Abdel-Rahman, Jonas Zitz
- Members of the W3C ODRL Community Group

References

[ODRL-11]	R. Iannella (ed). Open Digital Rights Language (ODRL) Version 1.1. W3C Note, 19 Sept 2002. http://www.w3.org/TR/odrl/
[ODRL-REQ]	S. Guth, R. Iannella (eds). Open Digital Rights Language (ODRL) Version 2.0 Requirements (Working Draft), ODRL Initiative, http://www.w3.org/2012/09/odrl/archive/odrl.net/2.0/v2req.html, 24 November 2004.
[ODRL-VOCAB]	R. Iannella, M. Steidl, S. Guth (eds). Open Digital Rights Language (ODRL) Version 2.1 – Common Vocabulary. Final Specification, W3C ODRL Community Group http://www.w3.org/community/odrl/vocab/2.1/
[ODRL-XML]	R. Iannella (ed). Open Digital Rights Language (ODRL) Version 2.1 – XML Encoding. Final Specification, W3C ODRL Community Group http://www.w3.org/community/odrl/xml/2.1/
[ODRL-ONTO]	M. McRoberts, V. Rodríguez Doncel. Open Digital Rights Language (ODRL) Version 2.1 – Ontology. Final Specification, W3C ODRL Community Group http://www.w3.org/ns/odrl/2/
[ODRL-JSON]	J. Öberg, S. Myles, L. Ai . Open Digital Rights Language (ODRL) Version 2.1 – JSON Encoding. Final Specification, W3C ODRL Community Group http://www.w3.org/community/odrl/json/2.1/
[RFC2119]	Key words for use in RFCs to Indicate Requirement Levels, S. Bradner. The Internet Society, March 1997. http://tools.ietf.org/html/rfc2119
[UML]	Unified Modeling Language (UML), Object Management Group, 2003. http://www.omg.org/technology/documents/formal/uml.htm