Open Digital Rights Language (ODRL)

Version: 1.1
Date: 2002-08-08

Available at: <http://odrl.net/1.1/ODRL-11.pdf>

Editor: Renato Iannella <renato@iprsystems.com>

Status

The Open Digital Rights Language (***ODRL***) is a proposed language for the Digital Rights Management (DRM) community for the standardisation of expressing rights information over content. The ***ODRL*** is intended to provide flexible and interoperable mechanisms to support transparent and innovative use of digital resources in publishing, distributing and consuming of electronic publications, digital images, audio and movies, learning objects, computer software and other creations in digital form.

The ***ODRL*** has been submitted to appropriate standards body for formal adoption and ratification.

The ***ODRL*** has no license requirements and is available in the spirit of "open source" software.

The change-bars (in the PDF version only) indicates differences with between Version 1.0 and Version 1.1.

Feedback and Comments

Please send feedback to <feedback@odrl.net> and general comments to <info@odrl.net>. Interested parties wishing to support the ***ODRL*** Initiative, please send email to <support@odrl.net>.

Further information on ***ODRL*** can be found at <http://odrl.net>

Table of Contents

1 Overview

Digital Rights Management (DRM) involves the description, layering, analysis, valuation, trading and monitoring of the rights over an enterprise's tangible and intangible assets. DRM covers the digital management of rights - be they rights in a physical manifestation of a work (eg a book), or be they rights in a digital manifestation of a work (eg an ebook). Current methods of managing, trading and protecting such assets are inefficient, proprietary, or else often require the information to be wrapped or embedded in a physical format.

A key feature of digitally managing rights will be the substantial increase in re-use of digital material on the Internet as well as the increased efficiency for physical material. The pervasive Internet is changing the nature of distribution of digital media from a passive one way flow (from Publisher to the End User) to a much more interactive cycle where creations are re-used, combined and extended ad infinitum. At all stages, the rights need to be managed and honoured with trusted services.

Current DRM technologies include languages for describing the terms and conditions, tracking asset usages by enforcing controlled environments or encoded asset manifestations, and closed architectures for the overall management of rights.

The Open Digital Rights Language (***ODRL***) provides the semantics for DRM expressions in open and trusted environments whilst being agnostic to mechanisms to achieve the secure architectures.

1.1 The Bigger Picture

It is envisaged that ***ODRL*** will "plug into" an open framework that enables peer-to-peer interoperability for DRM services. However, ***ODRL*** can also be used as an mechanism to express rights statements on its own and to plug into existing DRM architectures and frameworks.

DRM has traditionally been flavoured with security and encryption as a means to solve these issues, that is, Digital Rights Enforcement. This is the first-generation of DRM and represents a substantial narrowing of its real and broader capabilities. Second-generation DRM systems will provide these additional functions and support end-to-end supply chain services.

DRM systems should be based on and designed around open functional architectures and a robust and extensible information model. See [IANNELLA] for an overview of DRM Architectures and [ERICKSON] for DRM-enabled digital object models. The layers and relationships of rights can also quickly become very complex [HIGGS] and DRM systems must be designed to cater for this intricacy.

DRM systemts must also address and be consistent with the non-technical issues, including legal, business and social aspects of rights management. A recent W3C DRM Workshop [W3C-DRM] produced a number of position papers highlighting these important issues.

1.2 Scope

ODRL complements existing analogue rights management standards by providing digital equivalents, and supports an expandible range of new services that can be afforded by the digital nature of the assets in the Web environment. In the physical environment, ***ODRL*** can also be used to enable machine-based processing for rights management.

ODRL is a standard language and vocabulary for the expression of terms and conditions over assets. ***ODRL*** covers a core set of semantics for these purposes including the rights holders and

the expression of permissible usages for asset manifestations. Rights can be specified for a specific asset manifestation (ie format) or could be applied to a range of manifestations of the asset.

ODRL is focused on the semantics of expressing rights languages and definitions of elements in the data dictionary. ***ODRL*** can be used within trusted or untrusted systems for both digital and physical assets. However, ***ODRL*** does not determine the capabilities nor requirements of any trusted services (eg for content protection, digital/physical delivery, and payment negotiation) that utilises its language. Clearly, however, ***ODRL*** will benefit transactions over digital assets as these can be captured and managed as a single rights transaction. In the physical world, ***ODRL*** expressions would need an accompanying system with the distribution of the physical asset.

ODRL defines a core set of semantics. Additional semantics can be layered on top of ***ODRL*** for third-party value added services with additional data dictionaries.

ODRL does not enforce or mandate any policies for DRM, but provides the mechanisms to express such policies. Communities or organisations, that establish such policies based on ***ODRL***, do so based on their specific business or public access requirements.

ODRL depends on the use of unique identification of assets and parties. Common identification is a very difficult problem to have agreement across sectors and is why identification mechanisms and policies are outside the scope of ***ODRL***. Sector-specific versions of ***ODRL*** may address the need to infer information about asset and party identifiers.

The ***ODRL*** model is based on an analysis and survey of sector specific requirements (including models and semantics), and as such, aims to be compatible with a broad community base. ***ODRL*** intends to meet the common requirements for many sectors and has been influenced by the ongoing work and specifications/models of the following groups:

- The <indecs> Project [INDECS]
- Electronic Book Exchange Working Group [EBX]
- International Federation of Library Associations [IFLA]
- DOI Foundation [DOI]
- ONIX International [ONIX]
- Moving Pictures Expert Group [MPEG]
- IMS Global Learning Consortium [IMS]
- Dublin Core Metadata Initiative [DCMI]
- Propagate Project [PROPAGATE]
- OpenEBook Forum [OEBF]
- Publisher Requirements for Industry Standard Metadata [PRISM]
- Association of American Publishers [AAP]
- Digital Imaging [DIG35]

ODRL proposes to be compatible with the above groups by defining an independent and extensible set of semantics. ***ODRL*** does not depend on any media types as it is aimed for cross-sector interoperability.

1.3 Specification Overview

This document, along with its normative references, includes all the specification necessary for the implementation of interoperable ***ODRL*** applications.

The key words *must, must not, required, shall, shall not, should, should not, recommended, may,* and *optional* in this specification are to be interpreted as described in [RFC2119] which defines the significance of each particular requirement.

Examples used in this document are for demonstration purposes only.

The ***ODRL*** specification contains the following main sections:

- Section 2 describes the model for the ***ODRL*** expression language.
- Section 3 describes the semantics of the ***ODRL*** data dictionary elements.
- Section 4 describes the XML syntax used to encode the ***ODRL*** expressions and elements.
- Section 5 describes how additional ***ODRL*** data dictionaries can be defined.

The formal Expression Language and Data Dictionary elements are defined in:

- Appendix A: ODRL Expression Language XML Schema (Normative)
- Appendix B: ODRL Data Dictionary XML Schema (Normative)

2 ODRL Expression Language

The models for the ***ODRL*** language and data dictionary contain the structure and core semantics for the expressions. These models provide the overall framework for the expressions into which elements (from DRM data dictionaries) can be applied.

Note: The figures in this section cover both the Expression Language entities (shown as shadowed rectangles with "EX" in the top left corner) and the Data Dictionary elements (shown as rectangles with "DD" in the top left corner). Other namespaces ("DS" for XML Digital Signatures) are also shown.

2.1 ODRL Foundation Model

ODRL is based on an extensible model for rights expressions which involves a number of core entities and their relationships. This ***ODRL*** Foundation Model is shown in Figure 1.

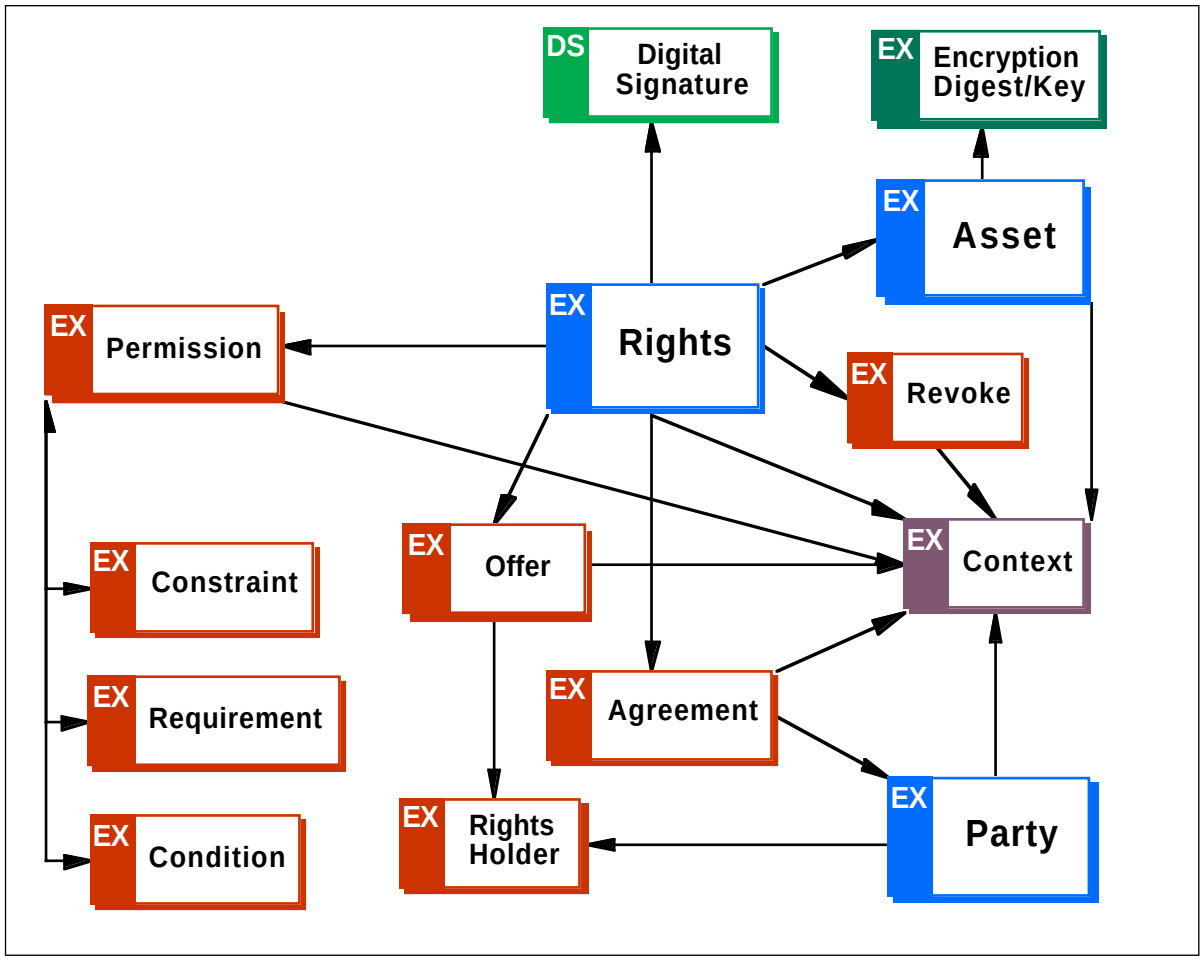

Figure 1. ODRL Foundation Model

The model, as shown in Figure 1, consists of the following three core entities:

- Assets
- Rights
- Parties

The Assets include any physical or digital content. The Assets must be uniquely identified and may consist of many subparts and be in many different formats. Assets can also be non-

tangible expressions of works and/or manifested in particular renditions. Assets may also be encrypted to enable secure distribution of content.

The Rights include Permissions which can then contain Constraints, Requirements, and Conditions. Permissions are the actual usages or activities allowed over the Assets (eg Play a video Asset). Constraints are limits to these Permissions (eg Play the video for a maximum of 5 times). Requirements are the obligations needed to exercise the Permission (eg Pay $5 each time you Play the video). Conditions specify exceptions that, if become true, expire the Permissions and renegotiation may be required (eg If Credit Card expires then all Permissions are withdrawn to Play the video).

The Parties include end users and Rights Holders. Parties can be humans, organisations, and defined roles. End users are usually the asset consumers. Rights Holders are usually parties that have played some role in the creation, production, distribution of the Asset and can assert some form of ownership over the Asset and/or its Permissions. Rights Holders may also receive royalties.

With these three core entities, the foundation model can then express Offers and Agreements. Offers are proposals from Rights Holders for specific Rights over their Assets. Agreements are when Parties enter into contracts or deals with specific Offers. The model can also then express revoking of any Offers or Agreements.

The representation of Offers and Agreements is important core aspect of ***ODRL***. This makes clear what the purpose of the rights expressions is achieving. Many different Offers can be created to meet various business models for assets. Offers can be linked, creating a hierarchy of options for end users. Agreements are the transformation of an Offer into a license for rights over an asset by parties. Also, there is no requirement that Offers be made prior to Agreements. After human interactions, Agreements can be created to express the accepted terms and conditions.

Most entities in the model can support a specific Context. A Context, which is relative to the entity, can describe further information about that entity or the relationship between entities. For example, the Context of an Agreement may specify the date of the transaction, the Context of a Party may specify their role. Although not mandatory, the use of Context to assign unique identifiers to the entire rights expression is highly recommended.

The Context also plays a very important role in identifying the entity (using a unique number/code from a standard identification scheme). This ability to uniquely reference any entity can be utilised in providing linkages between entities. For example, an Agreement can be linked to the original Offer by the latter's unique id number.

The general description of the Party and Asset entities is outside the scope of ***ODRL***. What is in scope is that these entities must be referenced by using a Uniform Resource Identifier [URI]. The URI is a unique identification system and may also include a resolution mechanism to the actual entity. Both of these entities contain a mechanism to refer to their unique identifiers, as well as a Context description.

The Asset entity (sometimes referred to as a Work, Content, Creation, or Intellectual Property), is viewed as a whole entity. If the Rights are assigned at the Asset's subpart level, then such parts would require to also be uniquely identifiable. However, ***ODRL*** can specify constraints on subparts of the asset. Additionally, Assets can be identified as to their layer of intellectual property as defined by the IFLA model [IFLA]. These include Work, Expression,

Manifestation, and Item. This features also allows rights to be expressed over non-tangible assets and individual instances.

These core Entities together allow for a wide and flexible range of ***ODRL*** expressions to be declared. Additionally, the expressions can be digitally signed.

The core entities (except for Asset and Party) are further explained in the following sections:

- Permissions (see Section 2.2 "ODRL Permission Model")
- Constraints (see Section 2.3 "ODRL Constraint Model")
- Requirements (see Section 2.4 "ODRL Requirement Model")
- Conditions (see Section 2.5 "ODRL Condition Model")
- Rights Holders (see Section 2.6 "ODRL Rights Holder Model")
- Contexts (see Section 2.7 "ODRL Context Model")
- Offers (see Section 2.8 "ODRL Offer Model")
- Agreements (see Section 2.9 "ODRL Agreement Model")
- Revoking Rights (see Section 2.10 "ODRL Revoke Model")

The security-based entities (Signature and Encryption) are further explained in the following section:

- Security (see Section 2.11 "ODRL Security Model")

Additional features of the ODRL expression language are explained in the following sections:

- Containers of entities (see Section 2.12 "Expression Containers")
- Sequence of entities (see Section 2.13 "Expression Sequence")
- Linking between entities (see Section 2.14 "Expression Linking")
- Inheritance between entities (see Section 2.15 "Expression Inheritance")

2.1.1 Foundation XML Example

The ***ODRL*** Foundation Model can be expressed using an XML binding. An example is shown in Example 1.

```
<rights>
    <context>.
        <uid> ... </uid>
    </context>
    <offer>
        <asset> ... </asset>
        <permission>
            <permission-type>
                <requirement> ... </requirement>
                <constraint> ... </constraint>
            </permission-type>
            <condition> ... </condition>
        </permission>
        <party>
            <context> ... </context>
            <rightsholder> ... </rightsholder>
        </party>
    </offer>
    <agreement>
        <context> ... </context>
        <party> ... </party>
        <permission> ... </permission>
        <asset> ... </asset>
    </agreement>
</rights>
```

Example 1. Foundation Model XML Syntax

Complete and formal syntactical examples are given in Section 4 "ODRL XML Syntax".

2.2 ODRL Permission Model

ODRL supports the expression of Permissions for both Offers and Agreements. This is the recognised set of activities allowed over the Asset. The ***ODRL*** Permission Model is shown in Figure 2.

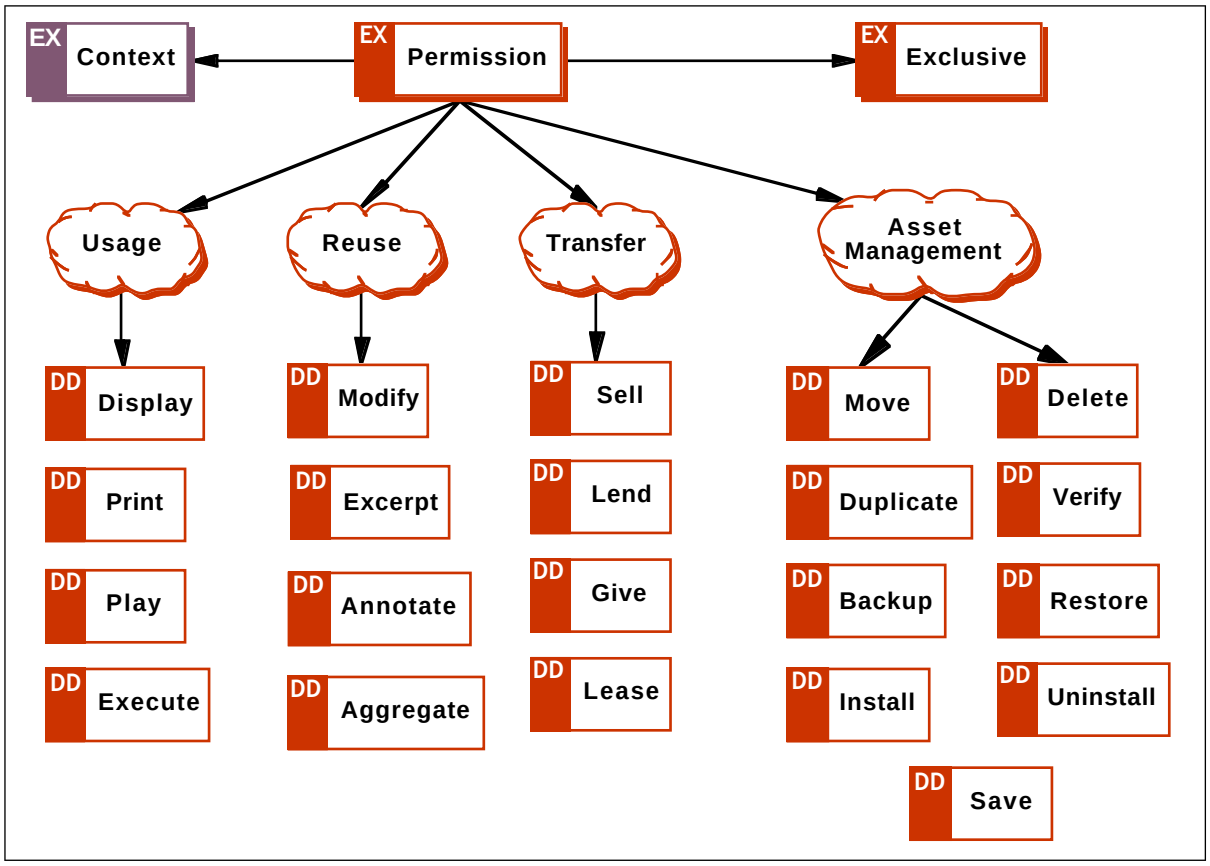

Figure 2. ODRL Permission Model

The Permission entity consists of an aggregation of four abstract entities. The abstract entities (depicted as clouds in Figure 2) are solely used to group similar permissions, and include:

- Usage - indicates a set of methods in which the Asset can be consumed (realised with: Display, Print, Play, Execute).
- Reuse - indicates a set of operations in which the Asset (or portions of it) can be re-utilised (realised with: Modify, Excerpt, Annotate, Aggregate).
- Transfer - indicates a set of procedures in which the rights over the Asset can be transferred (realised with: Sell, Lend, Give, Lease).
- Asset Management - indicates a set of digital asset management operations (realised with: Move, Duplicate, Delete, Verify, Backup, Restore, Save, Install, Uninstall).

Note: The listed elements above (eg Display, Print etc) are defined in full in Section 3 "ODRL Data Dictionary Semantics" and form the basis of the ***ODRL*** Data Dictionary.

Additionally, the Permissions support:

- an “Exclusivity” attribute that indicates if the granted permission is confined to nominated Parties (as per an Agreement).
- The Context entity for the purposes of assigned unique identifiers to specific sets or groups of Permissions.

A Permission must be associated with one or more Assets via an Offer or Agreement. This association can be explict (ie Permission is a child element of Offer or Agreement) or implict

via a reference from an Offer/Agreement). A Permission can be associated with zero or more Constraints, Conditions, and Requirements.

Important Note:

A Permission that is not specified in any Rights Expressions is not granted. That is, no assumptions should be made with regard to Permissions if they are not explicitly mentioned in the ***ODRL*** expression.

2.2.1 Permission XML Example

The ***ODRL*** Permission Model can be expressed using an XML binding. An example is shown in Example 2 in which permissions for display, print (with a constraint), and annotate have been granted.

```
<permission>
    <display/>
    <print>
        <constraint> ... </constraint>
    </print>
    <annotate/>
</permission>
```

Example 2. Permission Model XML Syntax

Complete and formal syntactical examples are given in Section 4 "ODRL XML Syntax".

2.3 ODRL Constraint Model

ODRL supports the expression of Rights Constraints. This is the recognised set of restrictions on the Permissions over the Asset. The ***ODRL*** Constraint Model is shown in Figure 3.

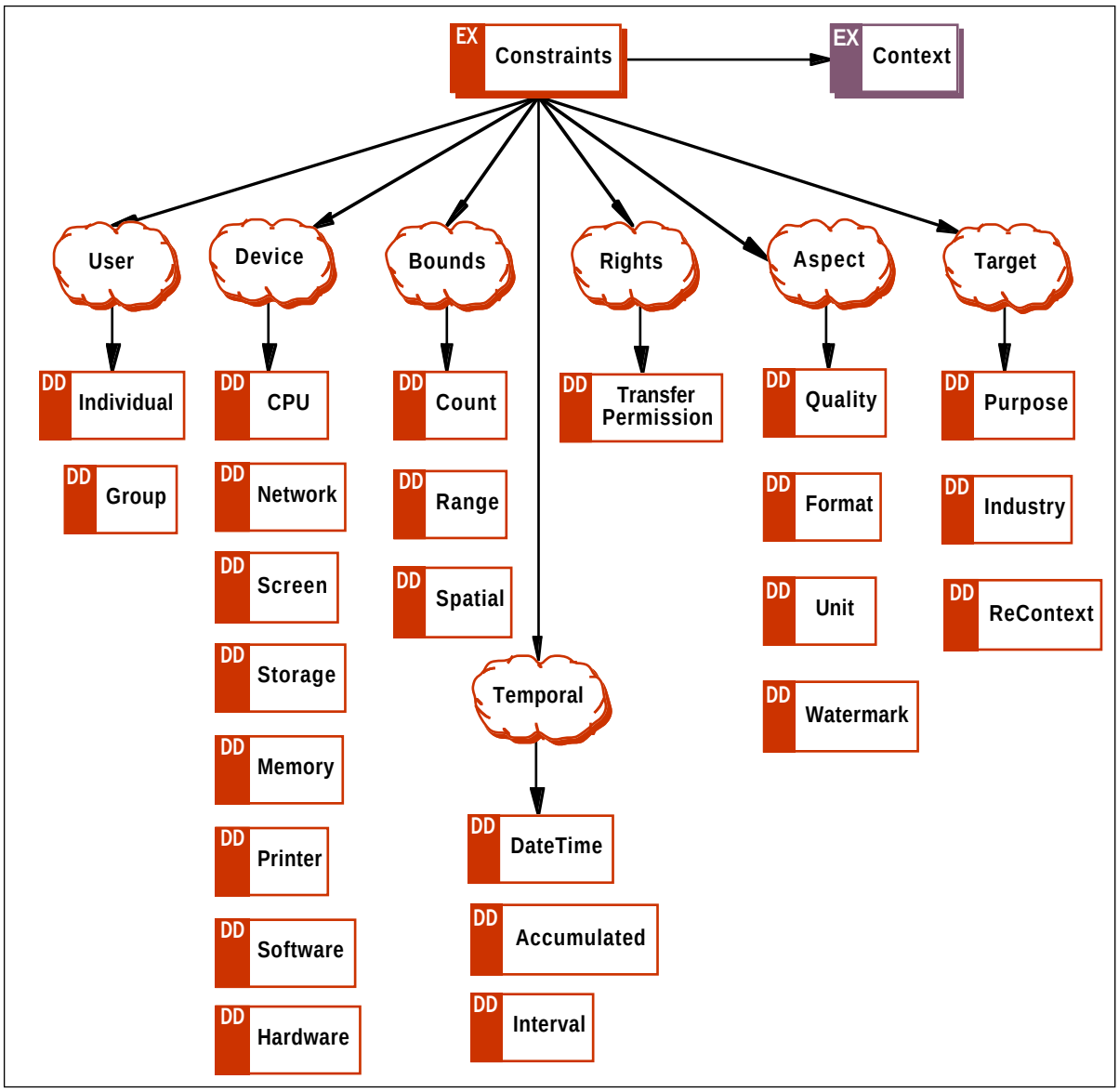

Figure 3. ODRL Constraint Model

The Constraints entity consists of an aggregation of several abstract entities. The abstract entities (depicted as clouds in Figure 3) are solely used to group similar constraints, and include:

- User - indicates a set of constraints which limits usage to identified user(s) (realised with: Individual, Group).
- Device - indicates a set of constraints which limits usage to physical devices or systems (realised with: CPU, Network, Screen, Storage, Memory, Printer, Software, Hardware).
- Bounds - indicates a set of constraints which limits usage to a fixed number or extent/ coverage (realised with: Count, Range, Spatial).
- Temporal - indicates a set of constraints which limits usage to temporal boundaries (realised with: Date Time, Accumulated, Interval).
- Aspect - indicates a set of constraints which limits usage to distinct features or expressions of the asset (realised with: Quality, Format, Unit, Watermark).

- Target - indicates a set of constraints which limits usage to where and how the asset is used (realised with: Purpose, Industry, ReContext).
- Rights - indicates a set of constraints which only applies to asset with a Transfer Permissions and enables the specification (and constraints) on downstream permissions (realised with: Transfer Permission).

Note: The listed elements above (eg Individual, Group etc) are defined in full in Section 3 "ODRL Data Dictionary Semantics" and form the basis of the ***ODRL*** Data Dictionary.

A Constraint is associated with one Permission. If a Constraint appears at the same level as a number of Permissions, then the Constraint applies to all of the Permissions.

Constraints can also have zero or more other Constraints. In this case, the child constraint applies to the parent constraint. As an example of this, consider the <unit> constraint and it's meanings when used with the <count> and <range> constraints:

- A <count> constraint that is outside a <unit> constraint means the number of times the right can be exercised (eg, a count of 5 inside a print element means “print 5 times”).
- A <count> constraint that is inside a <unit> constraint means the number of units (eg a count of 5 inside a *NumberOfPages type* unit element that is itself inside a print element means “print 5 pages”).
- A <range> constraint that is inside a <unit> constraint means the minimum or maximum ordinal position of the units (eg a range with min of 1, max of 100, inside a *NumberOfPages type* unit element that is itself inside a print element means “print pages numbered (ie positioned) between 1 and 100”).
- A meaning of a <range> constraint that is outside a <unit> constraint depends on the containing element.

Additionally all Constraint elements may have a Context element (to support the use of uid) and a “type” attribute (to refer to additional information).

Important Note:

Any Constraint that is expressed but cannot be performed by the consuming system, must not be granted. That is, if a system does not understand how to guarantee that a specified constraint be honoured it must not grant the permission at all.

For Permissions with multiple constraints, all constraints must be honoured with no conflicts arising. An error must be generated if the latter is true.

2.3.1 Constraint XML Example

The ***ODRL*** Constraint Model can be expressed using an XML binding. An example is shown in Example 3 in which the display permission is constrained to a particular CPU. The print

permission is limited to printing up to five times. The play permission is limited to a recurring period of seven days which itself can only occur up to ten times.

```
<display>
    <constraint>
        <cpu/>
    </constraint>
</display>
<print>
    <constraint>
        <count>5</count>
    </constraint>
</print>
<play>
    </constraint>
        <interval>P7D</interval>
        <constraint>
            <count>10 </count>
        </constraint>
    </constraint>
 </play>
```

Example 3. Constraint Model XML Syntax

Complete and formal syntactical examples are given in Section 4 "ODRL XML Syntax".

2.3.2 Transfer Permission Constraint Example

The Transfer Permission constraint is used to enable the downstream distribution of rights over assets. It applies only to assets that have one of the Transfer Permission applied as this ensures that the authorised party can transfer rights to other users. (This is sometimes called “narrow” rights.)

The Transfer Permission constraint may contain a list of other Permissions that either must be or may be passed along when the asset is transferred. If a Permission is not listed, then those permissions may not be passed along when the asset is transferred. In other words, the default is that no Permissions may be passed along when the asset is transferred.

When a Permission is passed along, the party that transfers the asset may or may not be allowed to alter the Permissions (typically by narrowing it). The Transfer Permission entity supports the following “downstream” attribute values to indicate this:

- equal
- less
- notgreater (less than or equal to)

If the downstream attribute is “equal”, the Permissions must be passed along without change when the asset is transferred. This is the default situation.

If the downstream attribute is “less”, a smaller subset of Permissions must be passed along when the asset is transferred.

If the downstream attribute is “notgreater”, the Permissions may be narrowed when the asset is transferred, but it must not be expanded.

The following Example 4 shows how the Transfer Permission constraint (applied to the Sell transfer permission) contains two options. The first option contains two other permissions (print, display) and the downstream attribute is set to “equal”. The means that when the asset is sold to other users, the seller (authorised under the agreement) must provide these

Permissions. The second option also contains two other permissions (aggregate, annotate) and the downstream attribute is set to “notgreater”. The means that when the asset is sold to other users, the seller may provide these two Permissions, one of them, or none of them.

```
<permission>
    <sell>
        <constraint>
            <transferPerm downstream=”equal”>
                <print/>
                <display/>
            </transferPerm>
            <transferPerm downstream=”notgreater”>
                <aggregate/>
                <annotate/>
            </transferPerm>
        </constraint>
    </sell>
</permission>
```

Example 4. Constraint Model - Transfer Permission - XML Syntax

2.3.3 For Each Member Constraint

Some constraints are based on entities that could contain a number of sub-entities (ie members).

For example, a Group constraint would normally refer to a collection of individuals. Under normal conditions, a Print permission with a Count constraint of “1” that appeared with a Group constraint would indicate that the asset can only be printed once. However, we may wish the Count constraint of “1” to apply to each member of the Group constraint, thus enabling each individual to print the asset once.

In cases where the constraint needs to apply at the level of each member of another constraint then we can support this feature with a “forEachMember” mechanism. This mechanism enables any member-based constraint to be identified and the referring constraint will apply to each member of that constraint.

ODRL defines a URI to represent these semantics:

http://odrl.net/1.1/#forEachMember

This URI value is used in the “type” attribute of the constraint element. The constraint elements refer to each other using id/idref (see Section 2.14).

Consider Example 4B (below) where an asset has been acquired for teaching a class (eg JAVA101). The teacher has purchased the rights to an asset and the first constraint (with id of “student01”) enables any member of the identified group to display the asset. The print permission also has a constraint of “1” but this constraint refers to the first constraint (using idref) and has the “forEachMember” type attribute. This means that each member of the Group can print the asset once (and not that the asset can be printed once in total).

```
<display>
    <constraint id="student01">
        <group>
            <context>
                <uid>ldap://dir.uni.au/class=JAVA101;o=A-UNI;c=AU</uid >
            </context>
        </group>
    </constraint>
</display>
<print>
    <constraint idref="student01" type="http://odrl.net/1.1/#forEachMember">
        <count>1</count>
    </constraint>
</print>
```

Example 4B: Constraint Model - For Each Member - XML Syntax

2.4 ODRL Requirement Model

ODRL supports the expression of Rights Requirements. This is the recognised set of preconditions that must be met in order to obtain the related permissons. The ***ODRL*** Requirements Model is shown in Figure 4.

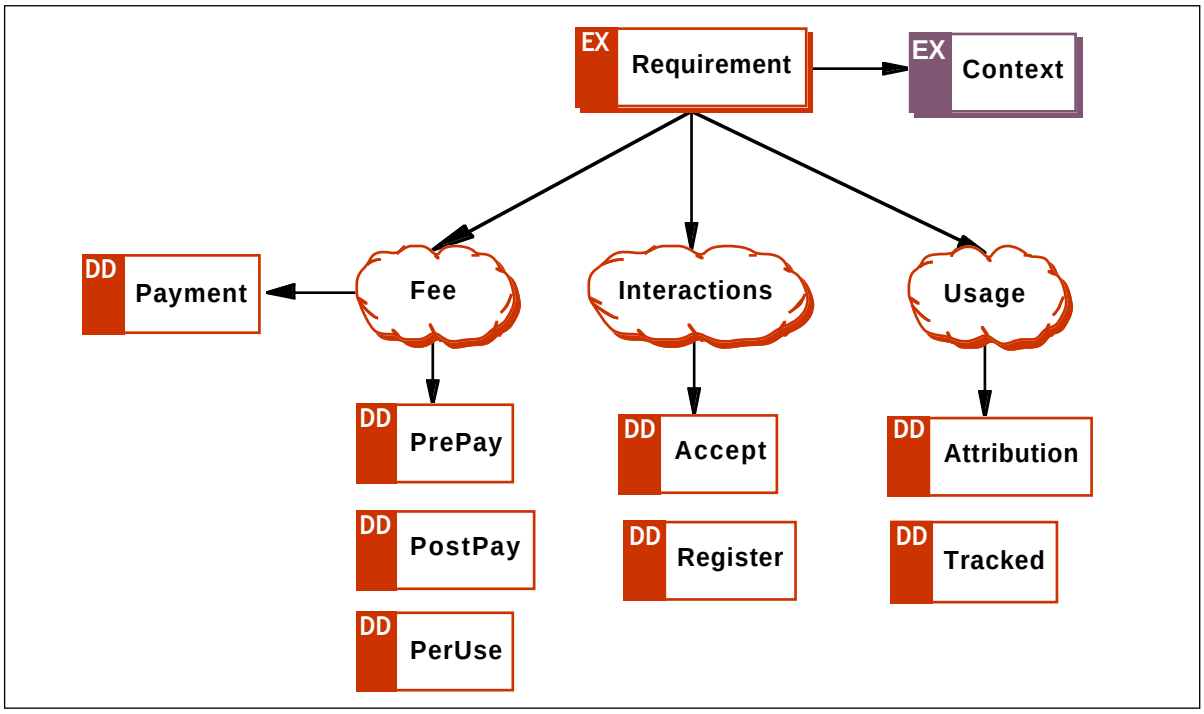

Figure 4. ODRL Requirement Model

The Requirement entity consists of three abstract entities. The abstract entities (depicted as a cloud in Figure 4) are solely used to group similar requirements, and includes:

- Fee - indicates a set of requirements for payments for usage (realised with: PrePay, PostPay, PerUse).
- Interactions - indicates a set of requirements on user interactions (realised with: Accept, Register).

- Usage - indicates a set of requirements on the use of the asset (realised with: Attribution, Tracked).

Note: The listed elements above (eg PrePay etc) are defined in full in Section 3 "ODRL Data Dictionary Semantics" and form the basis of the ***ODRL*** Data Dictionary.

A Requirement can be associated with one or many Permissions. If a Requirement appears at the same level as a number of Permissions, then the Requirement applies once to all of the Permissions. For Permissions with multiple Requirements, all Requirements must be honoured and no conflicts should arise. An error must be generated if the latter is true.

Additionally all Requirement elements may have a Context element.

The ***ODRL*** Data Dictionary also defines the common reusable Payment entity, A payment may consist of the fixed amount to pay, the currency of the amount, the taxation due (as a percentage) and a taxation code.

Important Note:

Any Requirement that is expressed but cannot be performed by the consuming system, must not be granted. That is, if a system does not understand how to guarantee that a specified Requirement has been met, then it must not grant the permission at all. Users of the system must be informed of the situation.

2.4.1 Requirement XML Example

The ***ODRL*** Requirement Model can be expressed using an XML binding. An example is shown in Example 5 in which the play permission requires a $AUD20 fee (plus 10% tax) to be paid per use.

```
<play>
    <requirement>
        <peruse>
            <payment>
                <amount currency="AUD">20.00</amount>
                <taxpercent code="GST">10.0</taxpercent>
            </payment>
        </peruse>
    </requirement>
</play>
```

Example 5. Requirement Model XML Syntax

Complete and formal syntactical examples are given in Section 4 "ODRL XML Syntax".

2.5 ODRL Condition Model

ODRL supports the expression of Rights Conditions. These are exceptions that are conditional events that, if become true (or occur), render the Permissions as no longer valid. The ***ODRL*** Condition Model is shown in Figure 5.

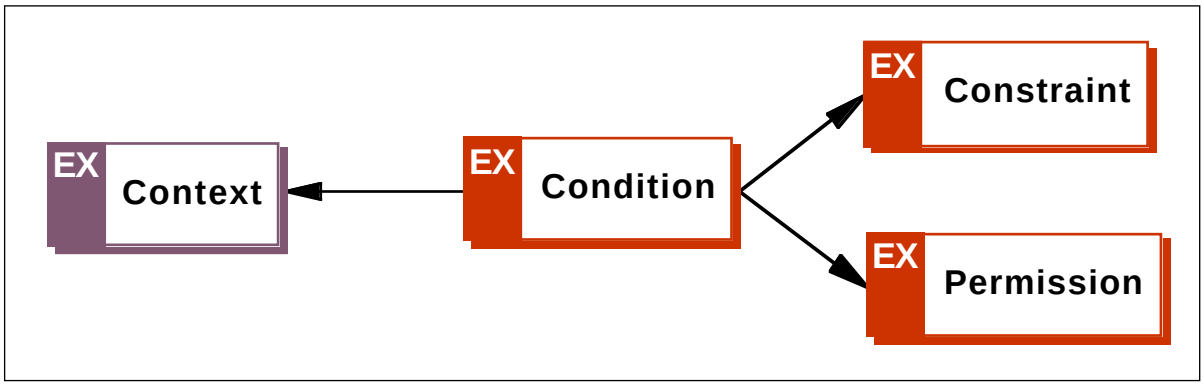

Figure 5. ODRL Condition Model

The Condition entity reuses two existing entities:

- Permission - indicates the set of permissions that will trigger the event
- Constraint - indicates a set of constraints. that will trigger the event

Note: The elements above are defined in full in Section 3 "ODRL Data Dictionary Semantics" and form the basis of the ***ODRL*** Data Dictionary.

If a Condition becomes true (ie the specified event occurs), then the system must cease granting the Permission to the user. The system may inform the user of the situation and offer information about renegotiating a new agreement.

A Condition can be associated with one or many Permissions. If a Condition appears at the same level as a number of Permissions, then the Condition applies once to all of the Permissions. For Permissions with multiple Conditions, all Conditions must be honoured and no conflicts should arise. An error must be generated if the latter is true.

Additionally all Condition elements may have a Context element.

Important Note:

Any Condition that is expressed but cannot be met or understood by the consuming system, must not grant the related Permissions. That is, if a system does not understand how to guarantee that a specified Condition has been met, then it must not grant the Permissions at all.

It is also important to note that Conditions and Constraints/Permissions behave in a similar ways but with opposite effect. A right can be expression with a Permission (what you are allowed to do) and a Constraint (limiting your permission). However, a Condition expresses the same semantics but as exceptions. If those conditions are met, the right is extinguished.

2.5.1 Condition XML Example

The ***ODRL*** Condition Model can be expressed using an XML binding. An example is shown in Example 6 in which there are two permissions; sell and play. The play permission has a constraint on a particular type of software meaning that the play permission must be expired if and when that software is used to play the video. Additionally, there is a constraint that applies to all of the permissions (both play and sell). This constraint is on the spatial area

(Australia) meaning that the permissions must be expired if and when the permissions are carried out in Australia.

```
<permission>
    <sell/>
    <play>
        <condition>
            <constraint>
                <software> ... </software>
            <constraint/>
        </condition>
    </play>
</permission>
<condition>
    <constraint>
        <spatial>
            <context>
                <uid>iso3166:AU</uid>
            </context>
        </spatial>
    <constraint/>
</condition>
```

Example 6. Condition Model XML Syntax

Complete and formal syntactical examples are given in Section 4 "ODRL XML Syntax".

2.6 ODRL Rights Holder Model

ODRL supports the identification of Rights Holders. The Rights Holder is a recognised Party and any set of entitlements that they are due for the use of their Asset. The ***ODRL*** Rights Holder Model is shown in Figure 6.

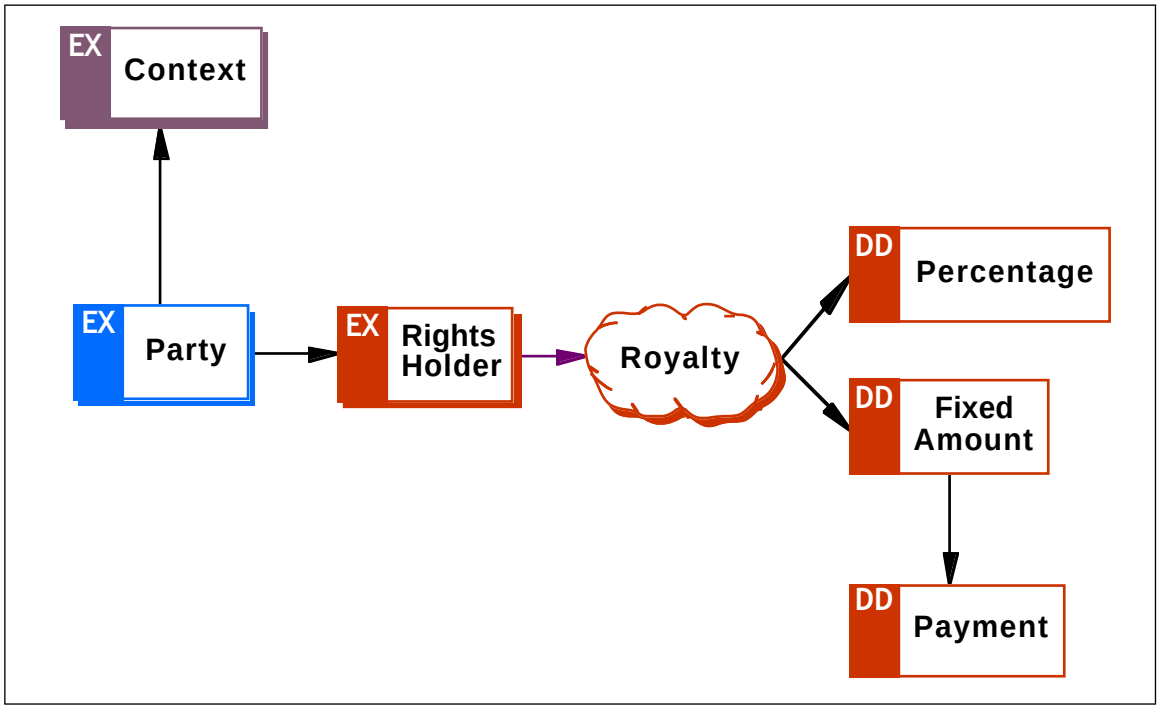

Figure 6. ODRL Rights Holder Model

The Rights Holder entity consists of one abstract entity: The abstract entity (depicted as a cloud in Figure 6) is solely used to group similar Rights Holder royalty entitlements, and include:

- Percentage - indicates a payment due to the indicated party for each transaction over the asset as a percentage of the value of the net transaction.
- Fixed Amount - indicates a payment due to the indicated party for each transaction over the asset as a fixed value of the net transaction.

Note: The listed elements above (eg Percentage etc) are defined in full in Section 3 "ODRL Data Dictionary Semantics" and form the basis of the ***ODRL*** Data Dictionary.

A Rights Holder can contain one or more Parties, and can be associated with one or many Assets. Parties can also be nested to indicate dependencies. For example; to support Rights Holder Sonya receiving 10% of Rights Holder Fiona's royalty (who may receive a fixed amount).

2.6.1 Rights Holder XML Example

The ***ODRL*** RightsHolder Model can be expressed using XML. An example is shown in Example 7 in which two identified parties share royalties with 90% to one and 10% to the other for each net transaction over their asset.

```
<party>
    <context> ... </context>
    <rightsholder>
        <percentage>90</percentage>
    </rightsholder>
</party>
<party>
    <context> ... </context>
    <rightsholder>
        <percentage>10</percentage>
    </rightsholder>
</party>
```

Example 7. RightsHolder Model XML Syntax

Complete and formal syntactical examples are given in Section 4 "ODRL XML Syntax".

2.7 ODRL Context Model

ODRL supports expressing additional information about entities and related entities. The ***ODRL*** Context Model is shown in Figure 7.

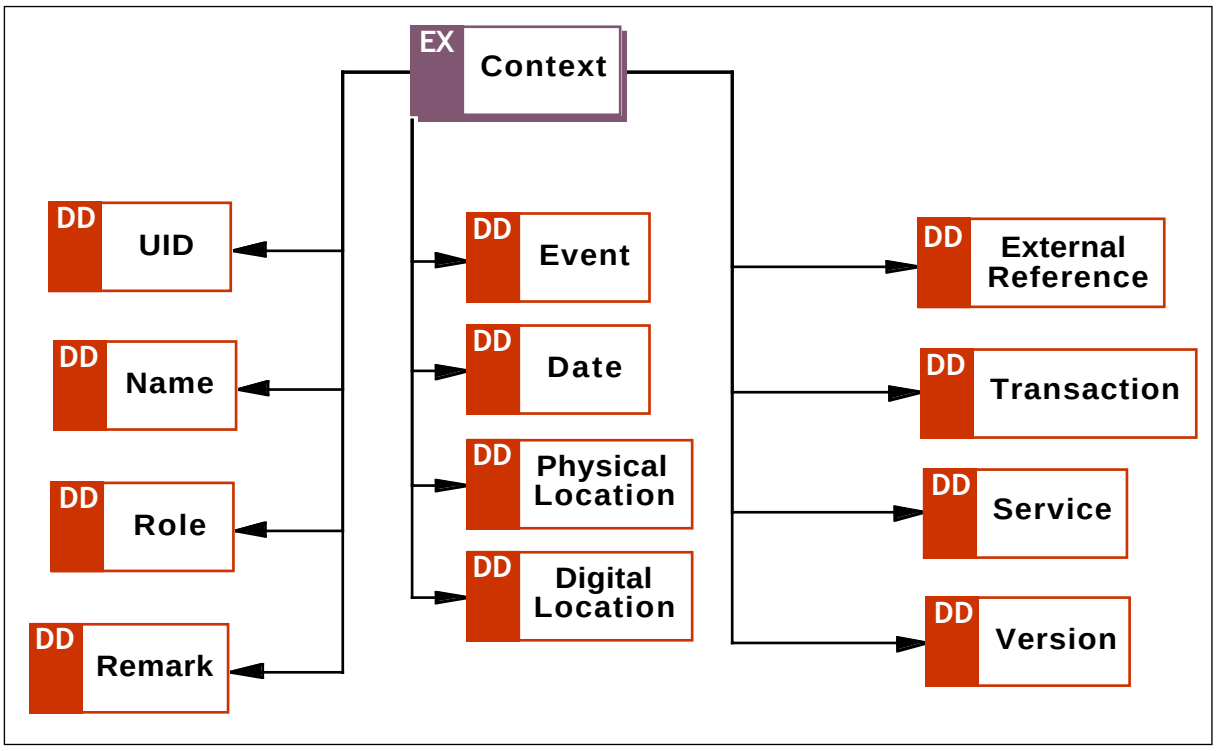

Figure 7. ODRL Context Model

The Context entity is an aggregation of ten other entities and includes:

- UID - a unique identification code for the entity.
- Name - a name used to describe the entity.
- Role - the role played by the entity.
- Remark - comments related to the entity.
- Version - indicates the version of the entity.
- Date - indicates the date the entity occurred or is valid for.
- Event - indicates the type of event
- Physical Location - indicates the physical location of the event/entity.
- Digital Location - indicates the digital location of the event/entity.
- External Reference - a link (URI) to an additional information about the entity.
- Transaction - information about the purchased transaction related to the entity.
- Service - a link (URI) to the service providing the entity.

Note: The listed elements above (eg UID etc) are defined in full in Section 3 "ODRL Data Dictionary Semantics" and form the basis of the ***ODRL*** Data Dictionary.

The Context is used for a number of different purposes and can be associated with any entity. When declaring Assets, it is used to indicate the unique identifier for the asset. When declaring Parties, it is used to indicate the unique identifier for the party, what role they may have played, and their name. The whole rights expression (eg Offer) can also have a Context to indicate the unique identifier for the expression as well as when it was created. The Agreement entity can also have a Context to provide information about the transaction.

The text-based Content entities (ie Name and Remark) may also indicate the human language of the text value (using the xml “lang” attribute).

Important Note:

The reference to identifiers (in the uid element) and vocabulary values (in the role element) should be conformant URIs. For identifiers, this requires the value to be one of the URI family (eg URL, URNs etc). For vocabulary values, this requires the scheme to be registered with an appropriate namespace. This mechanisms provides the most flexibility for additional URIs and vocabulary schemes to be utilised in ***ODRL***. However, for backward compatibility, the <uid> element may also contain string datatypes.

Multiple uid elements can also be used in the context. For example, an Asset may contain many parts and each can be referenced via a uid element enabling the collection to be viewed as a whole Asset.

2.7.1 Context XML Example

The ***ODRL*** Context Model can be expressed using an XML binding. An example is shown in Example 8 in which the context for a party is described.

```
<party>
    <context>
        <uid>x500:c=EX;o=FederalLibrary;ou=Registry;cn=MariaKBrown</uid>
        <name>Maria Brown</name>
        <role>onix:AO1</role>
        <reference>http://www.maria-k-brown.com/vcard.xml</reference>
    </context>
</party>
```

Example 8. Context Model XML Syntax

Complete and formal syntactical examples are given in Section 4 "ODRL XML Syntax".

2.8 ODRL Offer Model

ODRL supports expressing offers made from Rights Holders for specific rights over their assets. The ***ODRL*** Offer Model is shown in Figure 9.

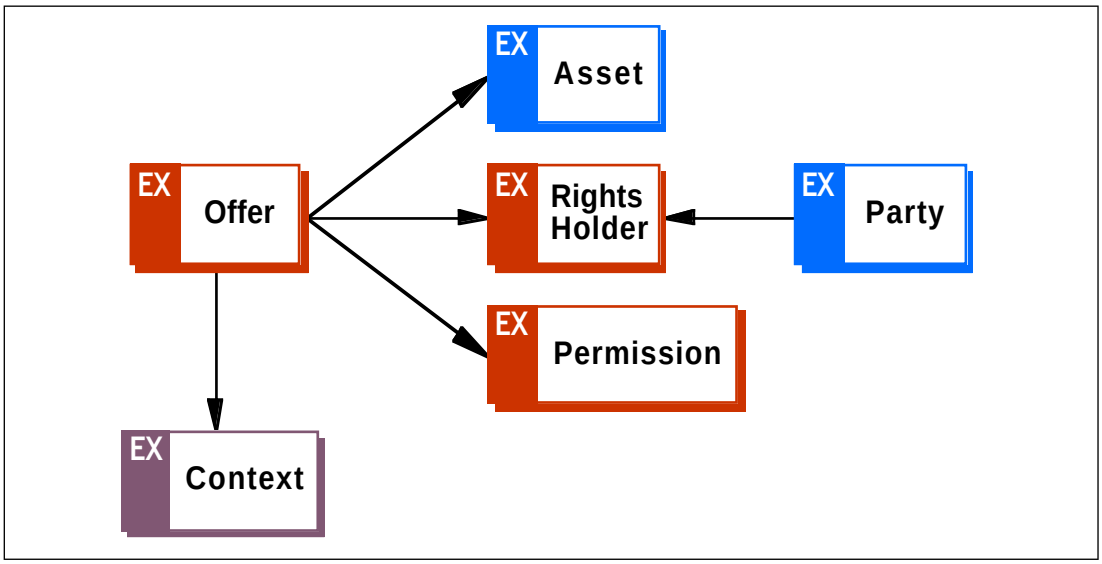

Figure 8. ODRL Offer Model

The Offer entity is an aggregation of four other entities and includes:

- Asset - information about the asset
- Rights Holder - information about the parties making the offer.

- Permission - information (or a link) to the usage permissions being offered.
- Context - details about the offer such as the date, time, location, identifier etc.

The Offer entity allows for detailed expressions of particular Rights Holders who have negotiated and agreed to offer particular permissions over their assets. Although not mandatory, the use of Context to assign unique identifiers to Offers is highly recommended. An Offer must include at least one Asset and Permission. If Rights Holder is not specified, then the system must support obtaining this information from other sources.

2.8.1 Offer XML Example

The ***ODRL*** Offer Model can be expressed using an XML binding. An example is shown in Example 10 in which an offer, with context, is described for a set of permissions over an asset with two rights holders.

```
<offer>
    <context>
        <uid>http://www.example.com/offer/3893823823472384888373</uid>
        <date><fixed>2001-10-10T09:00:00</fixed></date>
        <service>http://www.example.com/e-book-store</service>
    </context>
    <asset> ... </asset>
    <permission> ... </permission>
    <party>
        <rightsholder> ... </rightsholder>
    </party>
</offer>
```

Example 9. Offer Model XML Syntax

Complete and formal syntactical examples are given in Section 4 "ODRL XML Syntax".

2.9 ODRL Agreement Model

ODRL supports expressing agreements made between parties for specific rights over assets. The ***ODRL*** Agreement Model is shown in Figure 9.

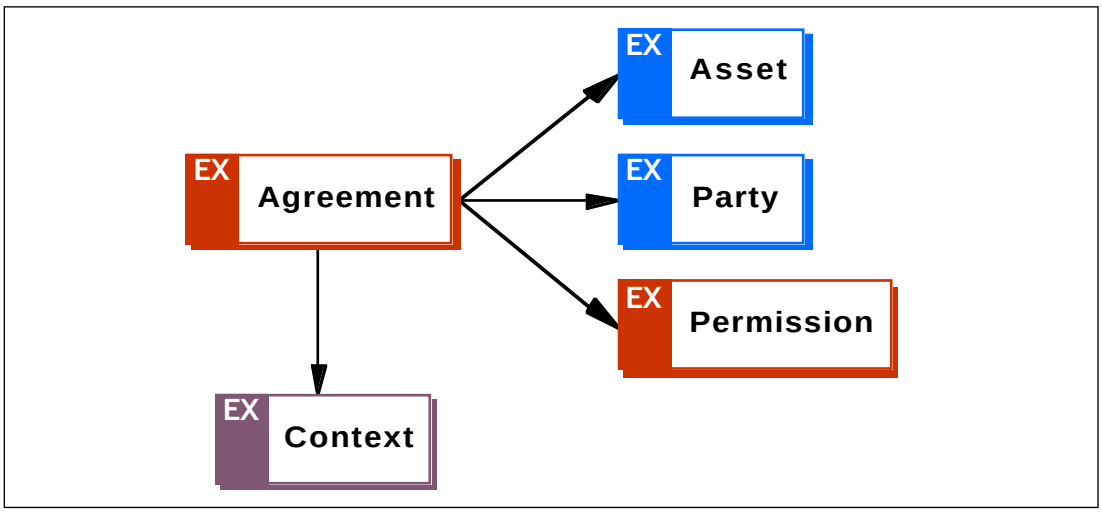

Figure 9. ODRL Agreement Model

The Agreement entity is an aggregation of four other entities and includes:

- Asset - information about the asset
- Party - information about the parties to the agreement.

- Permission - information (or a link) to the usage permissions being agreed to.
- Context - details about the agreement such as the date, time, location, identifier etc.

The Agreement entity allows for detailed expressions of particular parties who have negotiated and agreed to a set of particular permissions over some assets. Although not mandatory, the use of Context to assign unique identifiers to Agreements is highly recommended. An Agreement must include at least one Asset and Permission. If Party is not specified, then the system must support obtaining this information from other sources.

2.9.1 Agreement XML Example

The ***ODRL*** Agreement Model can be expressed using an XML binding. An example is shown in Example 10 in which an agreement, with context, is described between a party and a set of permissions over an asset.

```
<agreement>
    <context>
        <uid>doi:10.999/license/20010701/8736282828AAS</uid>
        <date><fixed>2001-07-01T10:31:30</fixed></date>
        <pLocation>Sydney, Australia</pLocation>
        <remark>Transacted by Example.Com</remark>
    </context>
    <party>
        <context> ... </context>
    </party>
    <asset> ... </asset>
    <permission>
        ...
    </permission>
</agreement>
```

Example 10. Agreement Model XML Syntax

Complete and formal syntactical examples are given in Section 4 "ODRL XML Syntax".

2.10 ODRL Revoke Model

ODRL supports revoking offers, agreements, and other rights expressions. The ***ODRL*** Revoke Model is shown in Figure 10.

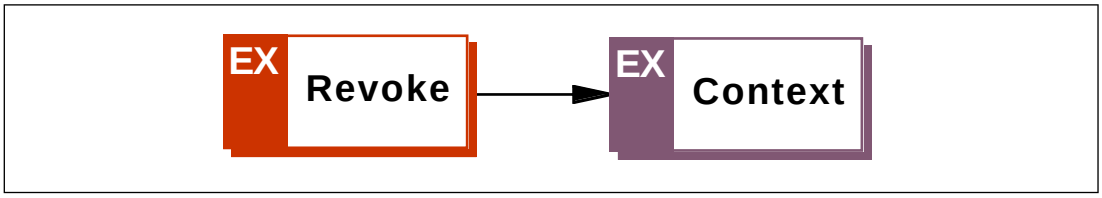

Figure 10. ODRL Revoke Model

The Revoke entity is an aggregation of one other entity:

- Context - identifier of the expression

The Revoke entity allows for the specification, via the unique identifier (uid) in Context, of the rights expression that is being revoked. The identifier (and identifier scheme) must be known to the consuming system.

Since the unique identifier (uid) in Context can be applied to any of the following:

- the entire rights expression
- Offers

- Agreements
- Permissions

then any (or all) of the above can be revoked.

Multiple expressions can be revoked at one time with multiple contexts in a Revoke statement.

2.10.1 Revoke XML Example

The ***ODRL*** Revoke Model can be expressed using an XML binding. An example is shown in Example 11 in which the prior agreement expressed in Example 10 (above) is being revoked. The Agreement identifier is used by the uid element.

```
<rights>
    <revoke>
        <context>
            <uid>doi:10.999/license/20010701/8736282828AAS</uid>
            <date><fixed>2001-10-30T12:30:30</fixed></date>
            <remark>Error in Original Agreement</remark>
        </context>
    </revoke>
</rights>
```

Example 11. Revoke Model XML Syntax

Complete and formal syntactical examples are given in Section 4 "ODRL XML Syntax".

2.11 ODRL Security Model

ODRL supports both secure rights expressions with digital signatures and specifying the encryption of assets. The security model uses a profile of the W3C XML Signature [XML-SIG] and W3C XML Encryption [XML-ENC] specifications to support enveloped signing of the entire rights expression (when using an XML binding) and including encryption information about assets. The ***ODRL*** security profile uses a subset of the elements used in these two specifications to ensure interoperability.

2.11.1 Encryption

The ***ODRL*** Encryption Model is shown in Figure 11 and consists of additional ***ODRL*** entities (entities marked with "EX") and entities taken from the Digital Signature (entities marked with "DS") and Encryption (entities marked with "EC") specifications.

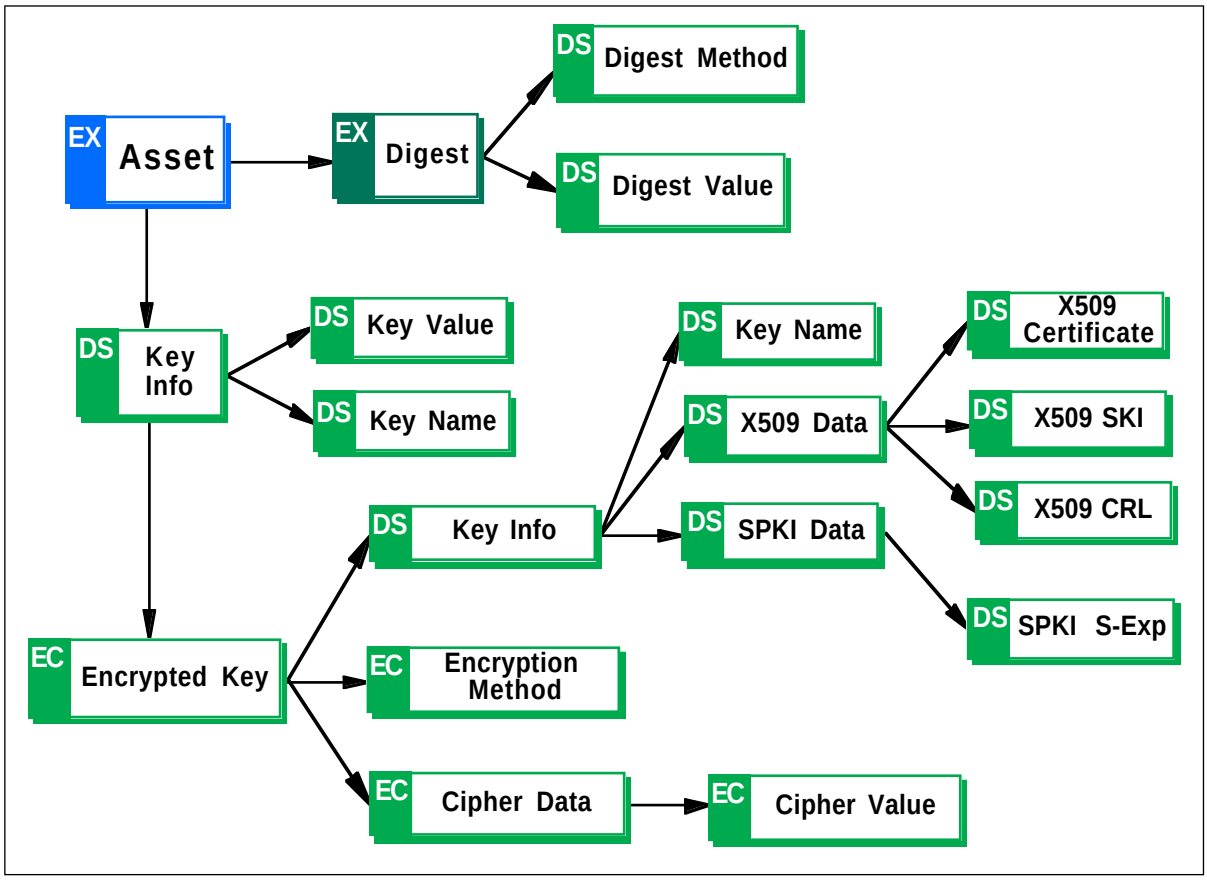

Figure 11. ODRL Encryption Model

To support information about the encryption of assets, ***ODRL*** defines a new entity called "Digest" that is a child of the ***ODRL*** Asset entity.

The Digest entity is designed to protect the integrity of the binding with the associated content and contains the following child entities:

- DigestMethod - indicates the algorithm used to calculate the digest value. The "SHA-1" algorithm must be supported.
- DigestValue - the value of the calculated digest.

The Asset entity contains a single "Key Info" entity that can either contain:

- "Key Value" and "Key Name", or
- multiple "Encrypted Key" child entities.

The Encrypted Key entity contains the following child entities:

- EncryptionMethod - indicates the encryption algorithm used. The “RSA” algorithm must be supported.
- Cipher Data - contains the raw encrypted data in the CipherValue child entity.
- Key Info - See section Section 2.11.3 "Digital Signature" for details on this entity.

2.11.2 XML Encryption Profile

A ***ODRL*** Encryption Profile is limited to the following required constraints:

- The permissible EncryptionMethod is RSA:
  http://www.w3.org/2001/04/xmlenc#rsa-1_5
- The permissible KeyInfo is X509Data:
  http://www.w3.org/2000/09/xmldsig#X509Data

2.11.3 Digital Signature

The ***ODRL*** Digital Signature Model is shown in Figure 12 and consists of entities taken from the Digital Signature (entities marked with “DS”) and Encryption (entities marked with “EC”) specification.

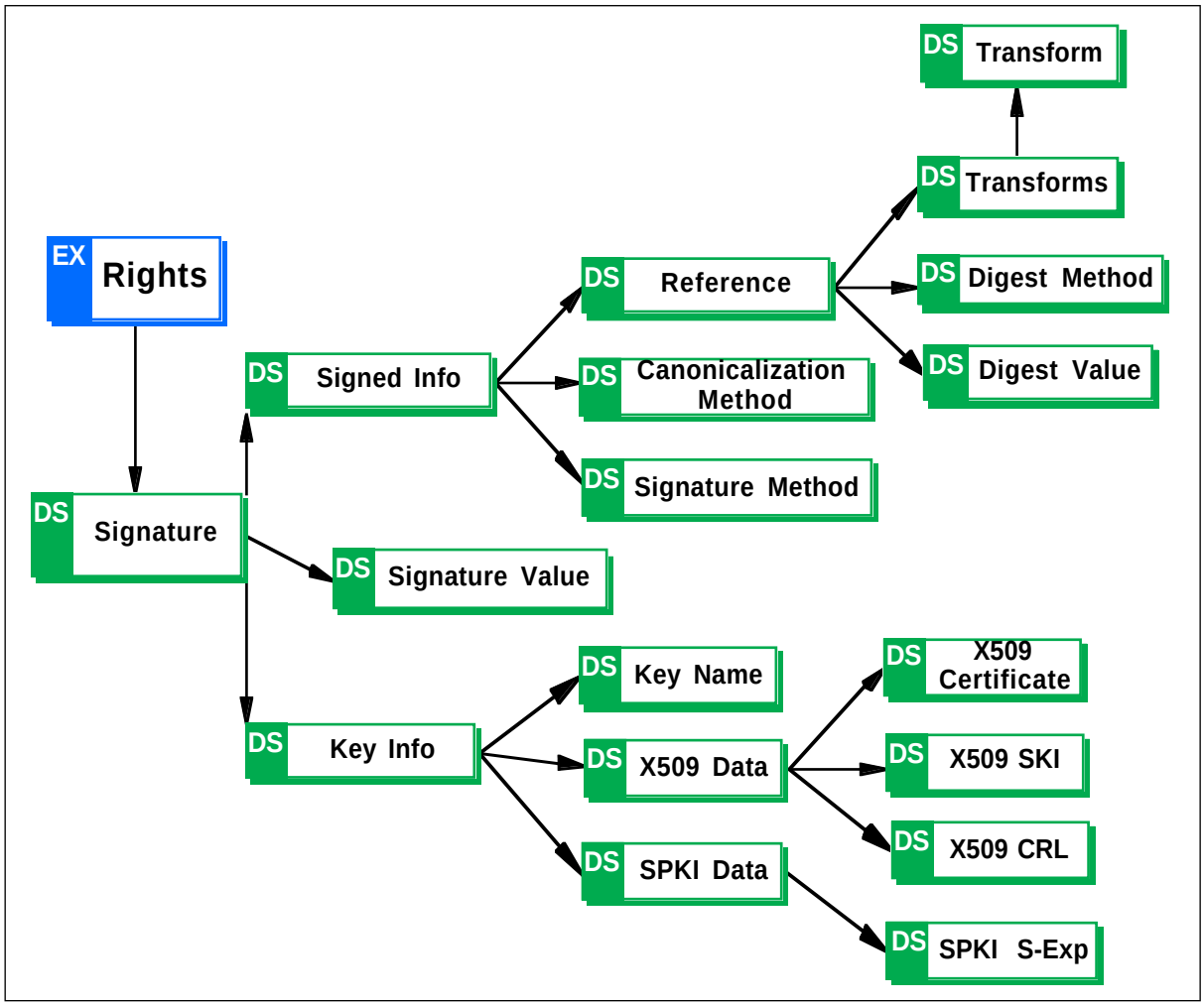

Figure 12. ODRL Digital Signature Model

To support the digital signing of the rights expression, ***ODRL*** utilises entities from the XML Signature specification. The ***ODRL*** expression and Signature are “enveloped” within the “rights” entity. The Signature entity is used to refer to the rights expression via its “id” attribute.

The Signature entity includes the "SignedInfo" entity containing the following three entities:

- CanonicalizationMethod - specifies the algorithm for canonicalization to be applied to the SignedInfo element prior to performing signature calculations. The "C14N" algorithm must be supported
- SignatureMethod - indicates the method used to generate the signature. The "RSA" method must be supported
- Reference - a link to the rights expression (via reference to its "id"). The Reference entity also contains a Transform entity to indicate any transformations required ("Enveloped Signature" and "C14N" are required to be supported in that order). The Reference entity also contains the DigestMethod and DigestValue entities (as described above in Section 2.11.1 "Encryption").

The Signature entity also includes:

- SignatureValue - the base64 encoded signature value.
- KeyInfo - this entity contains three other child entities; "X509Data", "SPKI Data" and "Key Name".

The "Key Name" entity contains a string value which may be used by the signer to communicate a key identifier to the recipient.

The "SPKI Data" entity contains information related to Simple Public Key Infrastructure (SPKI) public key pairs, certificates and other SPKI data and includes the "SPKI S-Exp" entity which is the base64 encoding of an SPKI canonical S-expression.

The "X509 Data" entity includes the following:

- X509 Certificate - contains a base64-encoded binary Distinguished Encoding Rules (DER) X509 V.3 certificate
- X509 SKI - contains the base64-encoded plain (non-DER-encoded) value of a X509 V.3 Subject Key Identifier (SKI) extension
- X509 CRL - contains a base64-encoded Certificate Revocation List (CRL)

2.11.4 XML Digital Signature Profile

A ***ODRL*** Digital Signature Profile signature is a *valid signature* (as defined in [XML-SIG]) limited to the following required constraints:

- The permissible CanonicalizationMethod is Canonical XML:
  http://www.w3.org/TR/2001/REC-xml-c14n-20010315
- The permissible SignatureMethod is RSA.
  http://www.w3.org/2000/09/xmldsig#rsa-sha1
- The permissible DigestMethod is SHA-1
  http://www.w3.org/2000/09/xmldsig#sha1
- The permissible Transform is Enveloped Signature and Canonical XML:
  http://www.w3.org/TR/2001/REC-xml-c14n-20010315
  http://www.w3.org/2000/09/xmldsig#enveloped-signature
- The permissible KeyInfo is X509Data and SPKIData
  http://www.w3.org/2000/09/xmldsig#X509Data
  http://www.w3.org/2000/09/xmldsig#SPKIData

2.11.5 Security XML Example

The ***ODRL*** Security Model can be expressed using XML. Example 12 shows the asset element now including the encryption element with appropriate digest and content encryption key (cek) values.

Example 13 shows a rights expression that has been digitally signed. The rights expression element has been allocated the “id” of “MyRightsData”. (Note: this “id” attribute is not the same as the “uid” element used in the “context” element, although they both could have the same value.) The id is used by the “Reference” element inside the Signature element. The “transforms” element also indicates the algorithms required to process the signed XML expression.

Both Example 12 and Example 13 need to include a mixture of XML namespaces (to support ***ODRL***, XML Signature, and XML Encryption) and these are clearly indicated with the namespaces prefixes.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<o-ex:rights xmlns:o-ex=”http://odrl.net/1.1/ODRL-EX”
                    xmlns:o-dd=”http://odrl.net/1.1/ODRL-DD”
                    xmlns:ds=”http://www.w3.org/2000/09/xmldsig#”
                    xmlns:enc=”http://www.w3.org/2001/04/xmlenc#”>
    <o-ex:context>
        <o-dd:uid>http://example.com/offers/3838383838.odrl</o-dd:uid>
        <o-dd:version>MRV Profile 2.8.99</o-dd:version>
    </o-ex:context>
    <o-ex:offer>
        <o-ex:asset>
            <o-ex:context>
                <o-dd:uid>http://example.com/1793874932.mov</o-dd:uid>
            </o-ex:context>
            <o-ex:digest>
                <ds:DigestMethod Algorithm=”http://www.w3.org/2000/09/xmldsig#sha1”/>
                <ds:DigestValue>--Base64-Encoded-Hash-Value--</ds:DigestValue>
            </o-ex:digest>
            <ds:KeyInfo>
                <enc:EncryptedKey>
                    <enc:CarriedKeyName>CEK-33247299234</enc:CarriedKeyName>
                    <enc:EncryptionMethod
                                        Algorithm=”http://www.w3.org/2001/04/xmlenc#rsa-1_5”/>
                    <ds:KeyInfo>
                        <ds:X509Data>
                            <ds:X509SKI>--Base64-Encoded-Subject-Key-ID--</ds:X509SKI>
                        </ds:X509Data>
                    </ds:KeyInfo>
                    <enc:CipherData>
                        <enc:CipherValue>--Base64-Enc-Encrypt-CEK--</enc:CipherValue>
                    </enc:CipherData>
                </enc:EncryptedKey>
            </ds:KeyInfo>
        </o-ex:asset>
        <o-ex:permission>
            <o-dd:play/>
        </o-ex:permission>
    </o-ex:offer>
</o-ex:rights>
```

Example 12. Security Model XML Syntax - Encryption

```
<o-ex:rights o-ex:id="MyRightsData"
                    xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
                    xmlns:o-dd="http://odrl.net/1.1/ODRL-DD"
                    xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <o-ex:context>
        <o-dd:uid>http://example.com/license/1191918237827.odrl</o-dd:uid>
    </o-ex:context>
    <o-ex:agreement>
        <o-ex:asset/>
        <o-ex:permission/>
        <o-ex:party/>
    </o-ex:agreement>
    <ds:Signature>
        <ds:SignedInfo>
            <ds:CanonicalizationMethod
                    ds:Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
            <ds:SignatureMethod ds:Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <ds:Reference ds:URI="#MyRightsData">
                <ds:Transforms>
                    <ds:Transform
                    ds:Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                    <ds:Transform
                    ds:Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
                </ds:Transforms>
                <ds:DigestMethod ds:Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <ds:DigestValue>---base64-encoded-hash-value---</ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>---base64-encoded-signature-value---</ds:SignatureValue>
        <ds:KeyInfo>
            <ds:X509Data>
                <ds:X509Certificate>---base64-encoded-signer-certificate---</ds:X509Certificate>
            </ds:X509Data>
        </ds:KeyInfo>
    </ds:Signature>
</o-ex:rights>
```

Example 13. Security Model XML Syntax - Digital Signature

Complete and formal syntactical examples are given in Section 4 "ODRL XML Syntax".

2.11.6 Encrypting *ODRL* Elements

In some circumstances, it may be necessary to only encrypt parts of ***ODRL*** expressions. For example, to make the rights holders royalty payments information confidential. The Security models described above can also support this requirements.

The encryption of specific ***ODRL*** elements can take place exactly as specified in Section 4 "Processing Rules" from [XML-ENC]. (The introductory examples in Section 2 of [XML-ENC] also provide a good overview.) The plain text ***ODRL*** elements to be protected will be replaced with the resulting encrypted XML structures. Sharing of the key used to protect these elements is out of the scope of ***ODRL*** and must be handled by the parties involved.

2.12 Expression Containers

The ***ODRL*** expression language also supports a number of mechanisms for grouping entities into containers. The groupings in the containers imply explicit semantics and must be supported.

There are three types of containers defined:

- And - the entities all must be supported / recognised
- Exclusive Or - only one of the entities must be supported / recognised
- Inclusive Or - one or more of the entities must be supported / recognised

A new structure, called “container” is utilised to aggregate other entities to enforce the above semantics. The container structure has an attribute called “type” (with fixed values of “and”, “ex-or”, “in-or”) to indicate the container type. The default container type is “and”. Additionally, when no container structure is used, the default of “and” is implied.

The container semantics only applies to the immediate children of the container.

Typically the container structure is used to aggregate permissions, constraints, conditions, and requirements but can be used on other entities.

2.12.1 Container XML Example

The ***ODRL*** Container structure can be expressed using XML. An example is shown in Example 14 in which the play permission is constrained to either (or both) the CPU or Storage device and the requirement is to either pay up front ($AUD200) or per use ($AUD1.50).

```
<permission>
    <play>
        <constraint>
            <container type="in-or">
                <cpu/>
                <storage/>
            </container>
        </constraint>
    </play>
    <requirement>
        <container type="ex-or">
            <prepay>
                <payment>
                    <amount currency=”AUD”>200.00</amount>
                </payment>
            </prepay>
            <peruse>
                <payment>
                    <amount currency=”AUD”>1.50</amount>
                </payment>
            </peruse>
        </container>
    </requirement>
</permission>
```

Example 14. Container XML Syntax

Complete and formal syntactical examples are given in Section 4 "ODRL XML Syntax".

2.13 Expression Sequence

The ***ODRL*** expression language also supports a number of mechanisms for specifying the sequence of entities for both total and partial ordering. A sequence of the entities implies the order in which the entities must be addressed/supported.

There are two types of sequence orders defined:

- total - the order is absolute
- partial - the order is optional

A new structure, called “sequence” is utilised to aggregate other entities to enforce the above semantics. The sequence structure has an attribute called “order” (with fixed values of “total”, “partial”) to indicate the sequence order. The default sequence order is “total”. Additionally, when no sequence structure is used, then no assumptions on the order should be made.

Within the sequence structure, the order is specified with “seq-item” elements, each with an attribute indicating the order number. The attribute (called “number”) is an integer equal to or greater than one. Note: the order is dictated by this number attribute, not the physical order of the expressions.

The sequence semantics only applies to the immediate children of the sequence.

Typically the sequence structure is used to order permissions, constraints, conditions, and requirements but can be used on other entities.

2.13.1 Sequence XML Example

The ***ODRL*** Sequence structure can be expressed using XML. An example is shown in Example 15 in which the requirement for the play permission is to undertake the absolute order of (1) register you details with the service provider, and then (2) pay the fee specified.

```
<permission>
    <play/>
    <requirement>
        <sequence order="total">
            <seq-item number=”1”>
                <register>
                    <context>
                        <service> http://example.com/registerhere </service>
                    </context>
                </register>
            </seq-item>
            <seq-item number=”2”>
                <prepay>
                    <payment>
                        <amount currency=”AUD”>100.00</amount>
                    </payment>
                </prepay>
            </seq-item>
        </sequence>
    </requirement>
</permission>
```

Example 15. Sequence XML Syntax

Complete and formal syntactical examples are given in Section 4 "ODRL XML Syntax".

2.14 Expression Linking

The ***ODRL*** expression language also supports the ability to explicitly link expression fragments. The links between the expression fragments imply explicit semantics and must be supported. The linking allows for ***ODRL*** expressions to be constructed by reusing existing expressions and allowing explicit links between fragments. Each ***ODRL*** fragment can be uniquely identified with the “id” attribute and referred to with the “idref” attribute.

In the usual case, entities appearing in ***ODRL*** rights expressions (offers or agreements) are assumed to be related. That is, a Permission entity that appears with an Asset entity are assumed to be related (ie this expresses the Asset’s permissions). When explicit linking is used, then the link relationships take precedence. The examples in the next section show how these cases can be expressed.

2.14.1 Link XML Example

The ***ODRL*** Link structure can be expressed using XML. An example is shown in Example 16 in which the sell permission only applies to ASSET-01 and the lend permission applies to both ASSET-01 and ASSET-02.

```
<rights>
    <offer>
        <asset id=”ASSET-01”>
            <context> ... <context>
        </asset>
        <asset id=”ASSET-02”>
            <context> ... <context>
        </asset>
        <permission>
            <asset idref=”ASSET-01”>
            <sell/>
        </permission>
        <permission>
            <asset idref=”ASSET-01”>
            <asset idref=”ASSET-02”>
            <lend/>
        </permission>
    </offer>
</rights>
```

Example 16. Link XML Syntax

Complete and formal syntactical examples are given in Section 4 "ODRL XML Syntax".

2.15 Expression Inheritance

The ***ODRL*** expression language also supports the ability to specify the inheritance of expressions between assets. That is, to allow parent/child relationships to be defined.

The child asset element can include the “inherit” element which will indicate from which other parent asset(s) to inherit the rights from. All the parent asset rights will be merged with the child asset rights. During this merge process, care must be taken that no conflicts arise inthe resultant rights expression.

The inherit element has an “override” attribute for child assets, that, if true, will not inherit the parent asset rights, but will define its own rights. The default is false.

The inherit element also has a “default” attribute for parent assets, that, if true, will not allow child assets to override its rights. The default is false.

Both “override” and “default” cannot be true at the same time.

2.15.1 Inheritance XML Example

The ***ODRL*** Inheritance structure can be expressed using XML. An example is shown in Example 17 in which the first rights expression specifies the play Permission for the identified asset. The second rights expression, for an asset related to the first, specifies that the rights should be inherited from the identified asset.

The expression also indicates not to override the inherited rights (the default), hence, the complete rights for the second asset includes the play and give Permissions. If the second expression did indicate to override the inherited rights, then the only Permissons for the second asset would be give.

```
<rights>
    <offer>
        <asset>
            <context>
                <uid>urn:example:asset:007</uid>
            </context>
        </asset>
        <permission>
            <play/>
        </permission>
    </offer>
</rights>

<rights>
    <offer>
        <asset>
            <context>
                <uid>urn:example:asset:007-Part1</uid>
            </context>
            <inherit override=”false” default=”false”>
                <context>
                    <uid>urn:example:asset:007</uid>
                </context>
            </inherit>
        </asset>
        <permission>
            <give/>
        </permission>
    </offer>
</rights>
```

Example 17. Inheritance XML Syntax

Complete and formal syntactical examples are given in Section 4 "ODRL XML Syntax".

3 ODRL Data Dictionary Semantics

This section details the semantics of all the ***ODRL*** data dictionary elements that is used within the ***ODRL*** expression language. The ***ODRL*** data dictionary elements form the basis of the language and can be extended (see Section 5 "ODRL Extensibility") by additional elements. Each element in the data dictionary is defined using the following attributes defined by ISO-11179:

- Name - The label assigned to the data element
- Identifier - The unique identifier assigned to the data element (that is used for the syntactical encoding of the element)
- Definition - A statement that clearly represents the concept and essential nature of the data element
- Comment - A remark concerning the application of the data element

The data dictionary elements are defined below in the following sections:

- Section 3.1 "Permissions Elements"
- Section 3.2 "Constraint Elements"
- Section 3.3 "Requirement Elements"
- Section 3.4 "Rights Holder Elements"
- Section 3.5 "Context Elements"

3.1 Permissions Elements

The Elements are defined in Table 1: Permission Elements.

Table 1. Permission Elements

Name	Identifier	Definition	Comment
Usage Permissons (pertaining to the end use of an asset)			
Display	display	The act of rendering the asset onto a visual device.	
Print	print	The act of rendering the asset onto paper or hard copy form.	
Play	play	The act of rendering the asset into audio/video form.	
Execute	execute	The act of executing the asset.	For example, machine executable code or Java
Transfer Permissions (pertaining to the downstream transfer rights of an asset)			
Sell	sell	The act of allowing the asset to be sold (ownership transfer) in exchange of value.	
Lend	lend	The act of allowing the asset to be made available for temporary use then returned (without exchange of value). During this period, the asset is only available to the lendee.	Temporal constraints are required for downstream use.
Give	give	The act of allowing the asset to be given away (ownership transfer) in perpetuity without exchange of value.	

Table 1. Permission Elements

Name	Identifier	Definition	Comment
Lease	lease	The act of allowing the asset to be made available for a fixed period of time then returned (for exchange of value). During this period, the asset is only available to the lessee.	Temporal constraints are required for downstream use.
Asset Management Permissions (pertaining to the digital management of an asset)			
Move	move	The act of allowing a digital asset to move between data storage devices.	Specification of constraints on the data storage devices may be allowed.
Duplicate	duplicate	The act of making an exact copy of a digital asset between data storage devices.	Specification of constraints on the data storage devices may be allowed.
Delete	delete	The act of deleting a copy of an asset.	
Verify	verify	The act of allowing authorization to check the authenticity of an asset.	
Backup	backup	The act of making copies of an asset for the purpose of guarding against the loss of the original due to accident or catastrophic media or equipment failure.	
Restore	restore	The act of allowing the conversion of a backup copy into a usable copy in a controlled manner.	
Save	save	The act of saving a copy (including any changes) of an asset to permanent storage.	
Install	install	The act of allowing for the operation of loading, verification and certification of an asset into a data storage device.	
Uninstall	uninstall	The act of allowing for the removal from or disabling of an asset in a data storage device.	
Reuse Permissions (pertaining to the re-utilisation of an asset creating a new asset)			
Modify	modify	The act of changing parts of the asset creating a new asset.	
Excerpt	excerpt	The act of extracting (replicating) unchanged parts (or all) of the asset for reuse into another asset.	
Annotate	annotate	The act of adding notations/ commentaries to the asset creating a new asset.	
Aggregate	aggregate	The act of using an asset (or parts of it) as part of a composite work or collection.	

3.2 Constraint Elements

The Elements are defined in Table 2: Constraint Elements.

Table 2. Constraint Elements

Name	Identifier	Definition	Comment
User Constraints (Any human or organisation)			
Individual	individual	An identifiable party acting as an individual.	Use Context to identify the individual.
Group	group	A number of identifiable parties acting as a collection of individuals.	Use Context to identify the group
Device Constraints (Any electronic or digital equipment or system)			
CPU	cpu	An identifiable computing system with a central processing unit (CPU).	Use Context to identify the device.
Network	network	An identifiable data network.	Use Context to identify the device. Use Range to indicate the IP Address restriction.
Screen	screen	An identifiable display output screen device.	For example, a screen reader or braille device. Use Context to identify the device.
Storage	storage	An identifiable storage media device.	For example, a hard disk or removable cartridge. Use Context to identify the device.
Memory	memory	An identifiable memory device.	For example, the clipboard. Use Context to identify the device.
Printer	printer	An identifiable hard copy printer.	Use Context to identify the device.
Software	software	An identifiable software application that must be present.	Use Context to identify the device.
Hardware	hardware	An identifiable generic hardware device.	Use Context to identify the device.
Bound Constraints (The limits/extent within which any entity can function)			
Count	count	A numeric count indicating the number of times the corresponding entity may be exercised.	A positive Integer. For example, the Print usage may be constraint with a count of 10 meaning that the asset can be printed zero to 10 times.

Table 2. Constraint Elements

Name	Identifier	Definition	Comment
Range	range	A numeric range indicating the min/max values of the corresponding entity that the constraint applies to. Contains the following sub entities: • min - the beginning of the range (inclusive) • max - the end of the range (inclusive) The numeric values must use the ordinal position when referring to external objects.	Positive and negative decimals must be supported. If there is no "min" or "max" value, then the range is open-ended. For example, a min of "1" (and no max) means that the range has an unlimited maximum. Note: "min" must always be less than or equal to "max" and one must always be present. For example, this is used to specify that only pages numbered 1 to 10 may be printed (using the "pages" subunit entity).
Spatial	spatial	Specification of a geographic area	Use Context to identify the spatial area with codes specified from a controlled vocabulary (eg [ISO3166] for country codes).
Temporal Constraints (The time limits within which any entity can function)			
Date Time	datetime	A date and/or time-based range. Contains the following sub entities: • start - the beginning of the range (inclusive) • end - the end of the range (inclusive) • fixed - an exact point in date/time	Date and Time value must conform to [ISO8601]. If there is no "start" and/or "end" value, then the range is open-ended. Note: "start" must always be less than or equal to "end" and one must always be present if there is no "fixed" value. "fixed" can only appear by itself.
Accumulated	accumulated	The maximum period of metered usage time.	Period value must conform to [ISO8601]. For example "P30H" indicates a 30 hour period.
Interval	interval	Recurring period of time in which the rights can be exercised.	Date and Time value must conform to [ISO8601]. For example "P7D" indicates a 7 day period.
Aspect Constraints (Any distinct feature of the Asset)			
Quality	quality	Specification of constraints on the quality aspects of the asset. Contains the following attribute: • type - the classification of the quality type	The values for the type attribute must be from a well known vocabulary and represented as a URI. For example, the resolution of an image or number of colours.

Table 2. Constraint Elements

Name	Identifier	Definition	Comment
Format	format	Specification of constraints on the format(s) of the asset. Contains the following attribute: • type - the classification of the format type	The values for the type attribute must be from a well known vocabulary and represented as a URI. For example, values can taken from the Internet Media Type [IMT] list.
Unit	unit	Specification of constraints on the whole asset or sub-parts of the asset. Contains the following attribute: • type - the classification of the sub-unit part type	The values for the type attribute must be from a well known vocabulary and represented as a URI.
Water Mark	watermark	Specification of watermarking requirements for the asset.	Use Context to identify the watermark information.
Target Constraints (How and where limits over the Asset)			
Purpose	purpose	Specification of a specific purpose to which the usage is constrained.	Use Context to identify the purpose from a known vocabulary
Industry	industry	Specification of a specific industry group to which the usage is constrained	Use Context to identify the industry from a known vocabulary
ReContext	recontext	Specification if the asset may or may not be re-contextualised.	Boolean value.
Rights Constraints (Constraints over Rights Permissions)			
Transfer Permission	transferPerm	Specification of constraints over Transfer Permissions for the downstream transfer of assets. Contains the following attribute: • downstream - the allowable narrowing of the specified Permissions (equal, less, notgreater)	See example in Section 2.3.2 "Transfer Permission Constraint Example".

3.3 Requirement Elements

The Elements are defined in Table 3: Requirement Elements.

Table 3. Requirement Elements

Name	Identifier	Definition	Comment
Common Entities			
Payment	payment	The amount of the payment. Contains the following sub-entities and attributes: • amount • currency (attribute) • taxpercent • code (attribute)	The amount must be a positive decimal to two decimal places. The mandatory currency must use [ISO4217] codes. The taxpercent must be a positive decimal between 0 and 100 (inclusive). The optional tax code can be any standard tax identifier.
Payment Requirements (Obligations in the form of financial payments)			

Table 3. Requirement Elements

Name	Identifier	Definition	Comment
Pre Pay	prepay	The amount due prior to the granting/use of the rights.	Use Payment entity. Temporal constraints may also be used.
Post Pay	postpay	The amount due after the use of the rights.	Use Payment entity. Temporal constraints may also be used.
Per Use	peruse	The amount due for each use of the granted rights.	Use Payment entity.
Interactions Requirements (Obligations in the form of user actions)			
Accept	accept	User must view and agree to textual information.	
Register	register	User must register their details with a service provider.	
Usage Requirements (Obligations in the form of usage conditions)			
Attribution	attribution	The use of the asset must always include attribution of the asset owners.	
Tracked	tracked	The User will be tracked for their use of the asset.	User must be aware of Privacy policy of the service provider.

3.4 Rights Holder Elements

The Elements are defined in Table 4: Rights Holder Elements.

Table 4. Rights Holder Elements

Name	Identifier	Definition	Comment
Royalty (Financial payments to Rights Holders)			
Percentage	percentage	The percentage of the net transaction amount due to the rights holder.	The value must be a positive decimal between 0 and 100 (inclusive). The total of all rights holders must not exceed 100.
Fixed Amount	fixedamount	The fixed amount due to the rights holder for each net transaction amount.	Use Payment entity.

3.5 Context Elements

The Elements are defined in Table 5: Context Elements.

Table 5. Context Elements

Name	Identifier	Definition	Comment
UID	uid	The unique identifier for the entity.	Values should be conformat to [URI] and/or string values.
Name	name	The general name given to the entity.	
Role	role	The role played by the identified entity.	The role values must be selected from existing vocabulary schemes and represented as a URI.

Table 5. Context Elements

Name	Identifier	Definition	Comment
Remark	remark	General comments or description about the entity	
Date	date	The date/time related to the entity	See DateTime entity for complete semantics.
Physical Location	pLocation	The physical place where the entity occurred or is located.	
Digital Location	dLocation	The digital location where the entity is located.	The link must be a URI.
External Reference	reference	A link to additional information about the entity.	The link must be a URI and resolve to (at least) well formed XML.
Service	service	A link to the service providing the entity	The link must be a URI
Version	version	The version of the entity	
Transaction	transaction	Information about the purchased transaction related to the entity	
Event	event	The type of event related to the entity	

4 ODRL XML Syntax

ODRL is expressed in schema-valid XML syntax. This syntax is formally specified by the XML Schemas defined in Appendix A and Appendix B. These are the normative references for the ***ODRL*** expression language and data dictionary (respectively).

Note: If the human language needs to be specified for any elements containing string values, then the use of the standard "xml:lang" attribute is recommended including the XML namespace.

4.1 ODRL XML Schemas

ODRL utilises two XML Schemas. One schema defines the Expression Language elements and constructs (see Appendix A), the other defines the Data Dictionary elements (see Appendix B). Both must be used to support valid ***ODRL*** expressions. Further, the Data Dictionary schema is dependent on the Expression Language schema as the former defines elements that are constrained by the expression language model.

4.2 ODRL XML Namespaces

ODRL supports XML Namespaces to indicate the scope and identity of its elements and other content description elements.

The XML Namespace URI for the ***ODRL*** Expression Language version 1.1 is:

http://odrl.net/1.1/ODRL-EX

The XML Namespace URI for the ***ODRL*** Data Dictionary version 1.1 is:

http://odrl.net/1.1/ODRL-DD

NOTE: These URIs should be considered *experimental* until the ***ODRL*** specification is formalised by an appropriate body and new XML Namespaces and URIs are assigned.

4.3 ODRL Linking

ODRL uses XML Schema ID and IDREF to refer from XML fragments to other fragments as described in Section 2.14 "Expression Linking". This is used to express the relationship between the core ***ODRL*** entities such as Asset, Permission, Offer and Agreement. Such elements can be identified with the "id" attribute then referred to via "idref" attribute.

It is important to recognise that as the ***ODRL*** expressions become more complicated, the need to partition and express linkages becomes paramount in order to have manageable and reusable rights expressions. The linking mechanism allows for quite complex expressions to be generated whilst preserving the interpretability of the overall rights language.

4.4 ODRL XML Examples

The XML syntax will be explained via a serious of scenarios covering different content sectors (eg ebooks, video, education, etc). Note that some of the XML Namespaces used in the examples (for vocabulary values) are fictional.

4.4.1 Ebook Scenario #1

Corky Rossi (an author) and Addison Rossi (an illustrator) publish their ebook via "EBooksRUS Publishers". They wish to offer consumers to purchase (a requirement to pay $AUD20.00 plus 10% tax) the ebook which is restricted to a single CPU only and they are allowed to print a maximum of 2 copies. They will also allow the first 5 pages (Units) of the

ebook to be viewed online for free (no requirement). Note the use of the NumberOfPages entity from the ONIX namespace to indicate the unit type.

The revenue split is 60% to the Author, 10% to the Illustrator and 30% to the Publisher. Their identities are shown from an X.500 repository and their roles indicated using the ONIX and MARC terms.

The XML encoding of this scenario in ***ODRL*** is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<o-ex:rights xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
        xmlns:o-dd="http://odrl.net/1.1/ODRL-DD"
        xmlns:onix="http://www.editeur.org/onix/ReferenceNames"
        xmlns:marc="http://www.loc.gov/marc/">
    <o-ex:offer>
        <o-ex:asset>
            <o-ex:context>
                <o-dd:uid>urn:ebook.world/999999/ebook/rossi-000001</o-dd:uid>
                <o-dd:name>Why Cats Sleep and We Don't</o-dd:name>
            </o-ex:context>
        </o-ex:asset>
        <o-ex:permission>
            <o-dd:display>
                <o-ex:constraint>
                    <o-dd:cpu/>
                </o-ex:constraint>
            </o-dd:display>
            <o-dd:print>
                <o-ex:constraint>
                    <o-dd:count>2</o-dd:count>
                </o-ex:constraint>
            </o-dd:print>
            <o-ex:requirement>
                <o-dd:prepay>
                    <o-dd:payment>
                        <o-dd:amount o-dd:currency="AUD">20.00</o-dd:amount>
                        <o-dd:taxpercent o-dd:code="GST">10.00</o-dd:taxpercent>
                    </o-dd:payment>
                </o-dd:prepay>
            </o-ex:requirement>
        </o-ex:permission>
        <o-ex:permission>
            <o-dd:display>
                <o-ex:constraint>
                    <o-dd:unit o-ex:type="onix:NumberOfPages">
                        <o-ex:constraint>
                            <o-dd:range>
                                <o-dd:min>1</o-dd:min>
                                <o-dd:max>5</o-dd:max>
                            </o-dd:range>
                        </o-ex:constraint>
                    </o-dd:unit>
                </o-ex:constraint>
            </o-dd:display>
        </o-ex:permission>
```

```
        <o-ex:party>
            <o-ex:context>
                <o-dd:uid>x500:c=AU;o=RightsDir;cn=CorkyRossi</o-dd:uid>
                <o-dd:role>onix:roles/A01</o-dd:role>
            </o-ex:context>
            <o-ex:rightsholder>
                <o-dd:percentage>60</o-dd:percentage>
            </o-ex:rightsholder>
        </o-ex:party>
        <o-ex:party>
            <o-ex:context>
                <o-dd:uid>x500:c=AU;o=RightsDir;cn=AddisonRossi</o-dd:uid>
                <o-dd:role>onix:roles/A12</o-dd:role>
            </o-ex:context>
            <o-ex:rightsholder>
                <o-dd:percentage>10</o-dd:percentage>
            </o-ex:rightsholder>
        </o-ex:party>
        <o-ex:party>
            <o-ex:context>
                <o-dd:uid>x500:c=AU;o=RightsDir;cn=EBooksRUS</o-dd:uid>
                <o-dd:role>marc:roles/pbl</o-dd:role>
            </o-ex:context>
            <o-ex:rightsholder>
                <o-dd:percentage>30</o-dd:percentage>
            </o-ex:rightsholder>
        </o-ex:party>
    </o-ex:offer>
</o-ex:rights>
```

4.4.2 Ebook Scenario #2

Following from Ebook Scenario #1 (above), a consumer (Mary Smith) decides to purchase the ebook under the conditions outlined. The ***ODRL*** expression below shows the Agreement generated. The Agreement has a context (with identifier and date). The Permission now shows the details of the constraint that is specific to the consumer. (In this case, the CPU identifier is saved with the license.)

The XML encoding of this scenario in ***ODRL*** is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<o-ex:rights xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
        xmlns:o-dd="http://odrl.net/1.1/ODRL-DD">
    <o-ex:agreement>
        <o-ex:context>
            <o-dd:uid>urn:ebook.world/999999/license/1234567890-ABCDEF</o-dd:uid>
            <o-dd:pLocation>Sydney, Australia</o-dd:pLocation>
            <o-dd:remark>Transacted by Example.Com</o-dd:remark>
        </o-ex:context>
        <o-ex:asset>
            <o-ex:context>
                <o-dd:uid>urn:ebook.world/999999/ebook/rossi-000001</o-dd:uid>
            </o-ex:context>
        </o-ex:asset>
        <o-ex:permission>
            <o-dd:display>
```

```
            <o-ex:constraint>
                <o-dd:cpu>
                    <o-ex:context>
                        <o-dd:uid>Adobe-WebBuy:CPD-ID:ER-393939-DSS-787878</o-dd:uid>
                    </o-ex:context>
                </o-dd:cpu>
            </o-ex:constraint>
        </o-dd:display>
        <o-dd:print>
            <o-ex:constraint>
                <o-dd:count>2</o-dd:count>
            </o-ex:constraint>
        </o-dd:print>
        <o-ex:requirement>
            <o-dd:prepay>
                <o-dd:payment>
                    <o-dd:amount o-dd:currency="AUD">20.00</o-dd:amount>
                    <o-dd:taxpercent o-dd:code="GST">10.00</o-dd:taxpercent>
                </o-dd:payment>
            </o-dd:prepay>
        </o-ex:requirement>
    </o-ex:permission>
    <o-ex:party>
        <o-ex:context>
            <o-dd:uid>urn:ebook.world/999999/users/msmth-000111</o-dd:uid>
            <o-dd:name>Mary Smith</o-dd:name>
        </o-ex:context>
    </o-ex:party>
  </o-ex:agreement>
</o-ex:rights>
```

4.4.3 Ebook Scenario #3

The ebook for an "Electronic Book Exchange" voucher is entitled "XML: A Manager's Guide". The rights owner is Addison-Wesley.

There is an agreement with the Distributor of this book (a company called "XYZ"). They have rights to Sell up to 5000 copies of the book.

Another licensed end user for this book is "John Doe". All his permissions start from the beginning of 2001 and finish at the end of 2004. He has rights to view the book for a total of 30 days. He can print up to 5 copies on a "trusted printer". He can print up to 5 pages between page 1 and 100 every week - up to a total of 100 pages - on any printer. He can also extract 5000 bytes every week up to a total of 1,000,000 bytes onto the Clipboard (memory). He has the right to Give the book away after one year of the permissions starting.

Note the use of the NumberOfPages and NumbeOfBytes entities from the ONIX and EBX namespaces. Note the use of id and idref for indicating the relationships between the agreements/rightsholder and the asset.

The XML encoding of this scenario in ***ODRL*** is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<o-ex:rights xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
        xmlns:o-dd="http://odrl.net/1.1/ODRL-DD"
        xmlns:onix="http://www.editeur.org/onix/ReferenceNames"
        xmlns:ebx="http://www.ebxwg.org/ebook/vocab/"
```

```
        xmlns:marc="http://www.loc.gov/marc/">
    <o-ex:context>
        <o-dd:uid>urn:ebook.world/999999/voucher/2001/1234567890</o-dd:uid>
        <o-dd:date>
            <o-dd:fixed>2001-05-01T08:30:00</o-dd:fixed>
        </o-dd:date>
        <o-dd:event>issued</o-dd:event>
    </o-ex:context>
    <o-ex:asset o-ex:id="a001">
        <o-ex:context>
            <o-dd:uid>isbn:872-2345-981</o-dd:uid>
            <o-dd:name>XML: A Manager's Guide</o-dd:name>
        </o-ex:context>
    </o-ex:asset>
    <o-ex:party>
        <o-ex:context>
            <o-dd:uid>http://publishers.net/registry/AWL </o-dd:uid>
            <o-dd:name>Addison-Wesley </o-dd:name>
            <o-dd:reference>http://www.addison-wesley.com</o-dd:reference>
        </o-ex:context>
        <o-ex:rightsholder/>
    </o-ex:party>
    <o-ex:agreement>
        <o-ex:asset o-ex:idref="a001"/>
        <o-ex:party>
            <o-ex:context>
                <o-dd:uid>http://distributors.net/registry/xyz</o-dd:uid>
                <o-dd:name>XYZ Company </o-dd:name>
                <o-dd:role>marc:dst</o-dd:role>
            </o-ex:context>
        </o-ex:party>
        <o-ex:permission>
            <o-dd:sell>
                <o-ex:constraint>
                    <o-dd:count>5000</o-dd:count>
                </o-ex:constraint>
            </o-dd:sell>
        </o-ex:permission>
    </o-ex:agreement>
    <o-ex:agreement>
        <o-ex:asset o-ex:idref="a001"/>
        <o-ex:party>
            <o-ex:context>
                <o-dd:uid>http://people.net/registry/john-doe-9999</o-dd:uid>
                <o-dd:name>John Doe</o-dd:name>
            </o-ex:context>
        </o-ex:party>
        <o-ex:permission>
            <o-ex:constraint>
                <o-dd:datetime>
                    <o-dd:start>2001-01-01T00:00:00</o-dd:start>
                    <o-dd:end>2004-12-31T23:59:59</o-dd:end>
                </o-dd:datetime>
            </o-ex:constraint>
```

```
<o-dd:display>
    <o-ex:constraint>
        <o-dd:accumulated>P30D</o-dd:accumulated>
    </o-ex:constraint>
</o-dd:display>
<o-dd:print>
    <o-ex:container o-ex:type=”and”>
        <o-ex:constraint>
            <o-dd:count>5</o-dd:count>
            <o-dd:printer>
                <o-ex:context>
                    <o-dd:uid>guid:TrustPrint/4747474742222</o-dd:uid>
                </o-ex:context>
            </o-dd:printer>
        </o-ex:constraint>
        <o-ex:constraint>
            <o-dd:unit o-ex:type=”onix:NumberOfPages”>
                <o-ex:constraint>
                    <o-dd:count>100</o-dd:count>
                </o-ex:constraint>
            </o-dd:unit>
            <o-dd:printer/>
        </o-ex:constraint>
        <o-ex:constraint>
            <o-dd:unit o-ex:type=”onix:NumberOfPages”>
                <o-ex:constraint>
                    <o-dd:range>
                        <o-dd:min>1</o-dd:min>
                        <o-dd:max>100</o-dd:max>
                    </o-dd:range>
                    <o-dd:count>5</o-dd:count>
                </o-ex:constraint>
            </o-dd:unit>
            <o-dd:interval>P7D</o-dd:interval>
            <o-dd:printer/>
        </o-ex:constraint>
    </o-ex:container>
</o-dd:print>
<o-dd:excerpt>
    <o-ex:constraint>
        <o-dd:memory>
            <o-ex:container o-ex:type=”and”>
                <o-ex:constraint>
                    <o-dd:unit o-ex:type=”ebx:NumberOfBytes”>
                        <o-ex:constraint>
                            <o-dd:count>1000000</o-dd:count>
                        </o-ex:constraint>
                    </o-dd:unit>
                </o-ex:constraint>
                <o-ex:constraint>
                    <o-dd:unit o-ex:type=”ebx:NumberOfBytes”>
                        <o-ex:constraint>
                            <o-dd:count>5000</o-dd:count>
                            <o-dd:interval>P7D</o-dd:interval>
```

```
                                </o-ex:constraint>
                            </o-dd:unit>
                        </o-ex:constraint>
                    </o-ex:container>
                </o-dd:memory>
            </o-ex:constraint>
        </o-dd:excerpt>
        <o-dd:give>
            <o-ex:constraint>
                <o-dd:datetime>
                    <o-dd:start>2002-01-01T00:00:00</o-dd:start>
                </o-dd:datetime>
            </o-ex:constraint>
        </o-dd:give>
    </o-ex:permission>
</o-ex:agreement>
</o-ex:rights>
```

4.4.4 Video Scenario #1

A video has three rights holders. Massimo Canale receives 75% of the transactions and Simona Canale receives the other 25%. Additionally, Maria Canale receives 10% of Simona's share. (Note the nesting of the last two parties.)

The video is offered at two different levels of quality (30 and 90dpi) and each has a different per use fee.

Note the use of the "resolution" entity from the MPEG7 namespace to indicate the Quality aspect of the asset. Note also that the whole expression has a context with a unique identifier (this is used in the next example).

The XML encoding of this scenario in ***ODRL*** is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<o-ex:rights xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
        xmlns:o-dd="http://odrl.net/1.1/ODRL-DD"
        xmlns:mpeg7="http://www.mpeg7.org/2001/MPEG-7_Schema">
    <o-ex:context>
        <o-dd:uid>doi:/voucher/383838383</o-dd:uid>
        <o-dd:name>The Voucher for XML: The Movie</o-dd:name>
        <o-dd:dLocation>http://example.com/odrl/383838383.xml</o-dd:dLocation>
    </o-ex:context>
    <o-ex:offer>
        <o-ex:asset>
            <o-ex:context>
                <o-dd:uid>doi:0.9999999/video/383838383</o-dd:uid>
                <o-dd:name>XML: The Movie</o-dd:name>
            </o-ex:context>
        </o-ex:asset>
        <o-ex:party>
            <o-ex:context>
                <o-dd:uid>x500:c=IT;o=Registry;cn=MassimoCanale</o-dd:uid>
            </o-ex:context>
            <o-ex:rightsholder>
                <o-dd:percentage>75</o-dd:percentage>
            </o-ex:rightsholder>
        </o-ex:party>
```

```
<o-ex:party>
    <o-ex:context>
        <o-dd:uid>x500:c=IT;o=Registry;cn=SimonaCanale</o-dd:uid>
    </o-ex:context>
    <o-ex:rightsholder>
        <o-dd:percentage>25</o-dd:percentage>
    </o-ex:rightsholder>
    <o-ex:party>
        <o-ex:context>
            <o-dd:uid>x500:c=IT;o=Registry;cn=MariaCanale</o-dd:uid>
        </o-ex:context>
        <o-ex:rightsholder>
            <o-dd:percentage>10</o-dd:percentage>
        </o-ex:rightsholder>
    </o-ex:party>
</o-ex:party>
<o-ex:permission>
    <o-dd:play>
        <o-ex:constraint>
            <o-dd:quality o-ex:type=”mpeg7:resolution”>
                <o-ex:constraint>
                    <o-dd:range>
                        <o-dd:max>30</o-dd:max>
                    </o-dd:range>
                </o-ex:constraint>
            </o-dd:quality>
        </o-ex:constraint>
        <o-ex:requirement>
            <o-dd:peruse>
                <o-dd:payment>
                    <o-dd:amount o-dd:currency=”ITL”>1000.00</o-dd:amount>
                </o-dd:payment>
            </o-dd:peruse>
        </o-ex:requirement>
    </o-dd:play>
    <o-dd:play>
        <o-ex:constraint>
            <o-dd:quality o-ex:type=”mpeg7:resolution”>
                <o-ex:constraint>
                    <o-dd:range>
                        <o-dd:max>90.0</o-dd:max>
                    </o-dd:range>
                </o-ex:constraint>
            </o-dd:quality>
        </o-ex:constraint>
        <o-ex:requirement>
            <o-dd:peruse>
                <o-dd:payment>
                    <o-dd:amount o-dd:currency=”ITL”>5000.00</o-dd:amount>
                </o-dd:payment>
            </o-dd:peruse>
        </o-ex:requirement>
    </o-dd:play>
</o-ex:permission>
```

```
    </o-ex:offer>
</o-ex:rights>
```

4.4.5 Super Distribution Example #1

A company offers the free distribution (ie the give permission) for the video asset (from 4.4.4 Video Scenario #1) to a particular individual (J J Jones) for a period ending at 31 December 2001. They do this by allowing the rights expression (itself an asset) to be given away.

Note that the video is not the asset in this case, but the rights expression is treated as an asset.

The XML encoding of this scenario in ***ODRL*** is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<o-ex:rights xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
             xmlns:o-dd="http://odrl.net/1.1/ODRL-DD">
    <o-ex:agreement>
        <o-ex:asset>
            <o-ex:context>
                <o-dd:uid>doi:10.9999999/voucher/383838383</o-dd:uid>
                <o-dd:name>The Voucher for XML: The Movie</o-dd:name>
            </o-ex:context>
        </o-ex:asset>
        <o-ex:party>
            <o-ex:context>
                <o-dd:uid>x500:c=US;o=Example;cn=JJJones</o-dd:uid>
            </o-ex:context>
        </o-ex:party>
        <o-ex:permission>
            <o-dd:give>
                <o-ex:constraint>
                    <o-dd:datetime><o-dd:end>2001-12-31T23:59:59</o-dd:end> </o-dd:datetime>
                </o-ex:constraint>
            </o-dd:give>
        </o-ex:permission>
    </o-ex:agreement>
</o-ex:rights>
```

4.4.6 Education Scenario #1

An online lecture (software application) is offered (execute) for a per-use fee of $AUD2.00 which must be constrained to a certain network. Additionally, for a fee of $AUD1000.00 the lecture can be lent to (up to) 100 individuals.

The rights holder receives 50% of the execute fee and 25% of the lend fee. (Note the linking between the party entity and the permission entities.)

The XML encoding of this scenario in ***ODRL*** is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<o-ex:rights xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
             xmlns:o-dd="http://odrl.net/1.1/ODRL-DD">
    <o-ex:offer>
        <o-ex:asset>
            <o-ex:context>
                <o-dd:uid>urn:example.com/learnobject/uni/aa/7373722</o-dd:uid>
                <o-dd:name>XML: The Lecture</o-dd:name>
            </o-ex:context>
        </o-ex:asset>
```

```
        <o-ex:party>
            <o-ex:context>
                <o-dd:uid>x500:c=AU;o=UniA;cn=ProfBean</o-dd:uid>
            </o-ex:context>
            <o-ex:rightsholder>
                <o-ex:container o-ex:type=”and”>
                    <o-dd:percentage>50</o-dd:percentage>
                    <o-ex:permission o-ex:idref=”p001”/>
                </o-ex:container>
                <o-ex:container o-ex:type=”and”>
                    <o-dd:percentage>25</o-dd:percentage>
                    <o-ex:permission o-ex:idref=”p002”/>
                </o-ex:container>
            </o-ex:rightsholder>
        </o-ex:party>
        <o-ex:permission o-ex:id=”p001”>
            <o-dd:execute>
                <o-ex:constraint>
                    <o-dd:network/>
                </o-ex:constraint>
                <o-ex:requirement>
                    <o-dd:peruse>
                        <o-dd:payment>
                            <o-dd:amount o-dd:currency=”AUD”>2.00</o-dd:amount>
                        </o-dd:payment>
                    </o-dd:peruse>
                </o-ex:requirement>
            </o-dd:execute>
        </o-ex:permission>
        <o-ex:permission o-ex:id=”p002”>
            <o-dd:lend>
                <o-ex:constraint>
                    <o-dd:individual>
                        <o-ex:constraint>
                            <o-dd:count>100</o-dd:count>
                        </o-ex:constraint>
                    </o-dd:individual>
                </o-ex:constraint>
                <o-ex:requirement>
                    <o-dd:prepay>
                        <o-dd:payment>
                            <o-dd:amount o-dd:currency=”AUD”>1000.00</o-dd:amount>
                        </o-dd:payment>
                    </o-dd:prepay>
                </o-ex:requirement>
            </o-dd:lend>
        </o-ex:permission>
    </o-ex:offer>
</o-ex:rights>
```

4.4.7 PRISM Scenario #1

A content provider of images provides an image to a company for use in their online magazine publication. Both parties use the [PRISM] metadata standard describing the content and include the ***ODRL*** inside the PRISM metadata to describe the agreement. The agreement

uses vocabularies defined by PRISM and limits the use to the Real Estate industry only, and in the USA after 2001-01-01 and in Greece before 2003-12-31.

The XML encoding of this scenario in ***ODRL*** is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:prism="http://prismstandard.org/namespaces/basic/1.0/"
        xmlns:prl="http://prismstandard.org/namespaces/prl/1.0/"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
        xmlns:o-dd="http://odrl.net/1.1/ODRL-DD">
 <rdf:Description rdf:about="http://wanderlust.com/2000/08/Corfu.jpg">
  <dc:title>Walking on the Beach in Corfu</dc:title>
  <dc:creator>John Peterson</dc:creator>
  <dc:contributor>Sally Smith, lighting</dc:contributor>
  <dc:format>image/jpeg</dc:format>
  <dc:identifier rdf:resource="http://wanderlust.com/content/2357845"/>

  <o-ex:rights>
   <o-ex:agreement>
    <o-ex:permission>
     <o-dd:display>
      <o-ex:constraint>
       <o-dd:industry o-ex:type="prism:vocabs/SIC/6532"/>
       <o-dd:spatial o-ex:type="prism:vocabs/ISO-3166/US">
        <o-ex:constraint>
         <o-dd:datetime>
          <o-dd:start>2001-01-01T00:00:00</o-dd:start>
         </o-dd:datetime>
        </o-ex:constraint>
       </o-dd:spatial>
       <o-dd:spatial o-ex:type="prism:vocabs/ISO-3166/GR">
        <o-ex:constraint>
         <o-dd:datetime>
          <o-dd:end>2003-12-31T23:59:59</o-dd:end>
         </o-dd:datetime>
        </o-ex:constraint>
       </o-dd:spatial>
      </o-ex:constraint>
     </o-dd:display>
    </o-ex:permission>
    <o-ex:party>
     <o-ex:context>
      <o-dd:uid>urn:prism.orgs:cool-cat-magazine</o-dd:uid>
     </o-ex:context>
    </o-ex:party>
   </o-ex:agreement>
  </o-ex:rights>
 </rdf:Description>
</rdf:RDF>
```

4.4.8 MPEG-21 Scenario #1

A digital audio CD is being offered for sale, including the restriction on downstream sales to be for "educational" purposes only. The systems involved utlise the [MPEG-21] Digital Item

Identification and Description metadata for identifying the content. The ***ODRL*** expression is part of one of the Descriptors describing the content of the Digital Item.

The XML encoding of this scenario in ***ODRL*** is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<di:DIDL xmlns:di="urn:mpeg:mpeg21:2002/01-DIDL-NS"
         xmlns:diid="urn:mpeg:mpeg21:2002/01-DIID-NS"
         xmlns:sector="urn:sectors:2002:vocab"
         xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
         xmlns:o-dd="http://odrl.net/1.1/ODRL-DD">
   <di:Item>
      <di:Descriptor>
         <di:Statement type="text/text">Cool Cats CD Single</di:Statement>
      </di:Descriptor>
      <di:Descriptor>
         <di:Statement type="text/xml">
            <diid:identifier>A1-888999-0029733-22-F</diid:identifier>
         </di:Statement>
      </di:Descriptor>
      <di:Descriptor>
         <di:Statement type="text/xml">
            <o-ex:rights>
               <o-ex:offer>
                  <o-ex:permission>
                     <o-dd:sell>
                        <o-dd:transferPerm o-dd:downstream="equal">
                           <o-dd:play>
                              <o-ex:constraint>
                                 <o-dd:purpose o-ex:type="sectors:educational"/>
                              </o-ex:constraint>
                           </o-dd:play>
                        </o-dd:transferPerm>
                     </o-dd:sell>
                  </o-ex:permission>
               </o-ex:offer>
            </o-ex:rights>
         </di:Statement>
      </di:Descriptor>
   </di:Item>
</di:DIDL>
```

5 ODRL Extensibility

ODRL defines both the core structure of the expression language and a set of core elements as part of the ***ODRL*** data dictionary. Additional data dictionaries can be defined and used within the ***ODRL*** framework To define a new data dictionary, a new XML Schema must be created that imports the ***ODRL*** expression language schema.

The expression language schema defines elements that are utilised as the "place holder" for data dictionary elements. These elements are defined as "abstract" ensuring that they cannot appear in an instance of an ***ODRL*** XML expressions and (by default) are of "anyType" data type to allow maximum extensibility. For example, the ***ODRL*** Expression Language schema defines:

```
<xsd:element name="permissionElement" abstract="true"/>
```

which allows Permission elements to be defined in the ***ODRL*** Data Dictionary schema, such as:

```
<xsd:element name="display" type="o-ex:permissionType"
                    substitutionGroup="o-ex:permissionElement"/>
```

The data type of the Data Dictionary element should either be the corresponding data type (eg "permissionType") or any specific data type that is relevant to that elements definition. For example, it could be a string or boolean. In the example above, the data type of "permissionType" allows for maximum flexibility such as nesting other permissions, constraints, etc. When creating new Data Dictionary elements ensure that you are aware of how the element will be used to in the expression instances, and make sure the assigned data type can cater for this.

The Data Dictionary schema utilises the "substitutionGroup" mechanism from XML Schema to ensure that only the appropriate elements can be used in the correct position in the ***ODRL*** Expression Language.

There are six substitution elements that can be used for additional data dictionaries:

- permissionElement (permissionType)
- requirementElement (requirementType)
- constraintElement (constraintType)
- conditionElement (conditionType)
- contextElement (contextType)
- rightsholderElement (rightsHolderType)

When defining additional data dictionary elements, ensure that they are defined using one (and only one) of these substitution groups. The data dictionary elements can either have the datatypes (listed above) or any derived datatype. The additional data dictionary elements may also define extensions, such as attributes or other complex data structures.

5.1 Example

Lets assume that the Mobile sector wishes to use ***ODRL*** but needs to extend the data dictionary to provide two new Permissions (ie "ring", "send") that are specific to mobile content. There two permissions can be declared in the schema as substitution group elements for the permissionElement.

Additionally, one new Requirement (ie "sms-accept") is defined to support the users need to accept SMS messages as part of any license offer. This requirement also indicates the language of the text messages. In this case, the new requirement is declared in the schema as

substitution group elements for the requirementElement and defines an extension (of the base datatype) to include a new attribute for the language.

The first step is to define the new XML Schema for this data dictionary as shown in Example 18 and assign an appropriate XML Namespace.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://example.net/MOBILE-DD"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
        xmlns:mm="http://example.net/MOBILE-DD"
        elementFormDefault="qualified" attributeFormDefault="qualified">
    <xsd:import namespace="http://odrl.net/1.1/ODRL-EX"
            schemaLocation="http://odrl.net/1.1/ODRL-EX-11.xsd"/>
    <xsd:element name="ring" type="o-ex:permissionType"
                substitutionGroup="o-ex:permissionElement"/>
    <xsd:element name="send" type="o-ex:permissionType"
                substitutionGroup="o-ex:permissionElement"/>
    <xsd:element name="sms-accept" substitutionGroup="o-ex:requirementElement">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="o-ex:requirementType">
                    <xsd:attribute name="lang" type="xsd:language"/>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Example 18. Mobile Data Dictionary XML Schema

The next step is to use the Mobile Data Dictionary as you would normally for ***ODRL*** expressions. Consider the Offer shown in Example 19. In this case the offer over the specified asset is to "give" it away and to allow the "ring" permission with a constraint of 200 times. There is also a Requirement to pay an amount and to accept SMS messages (sms-accept) in Australian English.

```
<?xml version="1.0" encoding="UTF-8"?>
<o-ex:rights xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
             xmlns:o-dd="http://odrl.net/1.1/ODRL-DD"
             xmlns:mm="http://example.net/MOBILE-DD">
    <o-ex:offer>
        <o-ex:asset>
            <o-ex:context>
                <o-dd:uid>10.999999/ringtone/1234567WWW</o-dd:uid>
            </o-ex:context>
        </o-ex:asset>
        <o-ex:permission>
            <o-dd:give/>
            <mm:ring>
                <o-ex:constraint>
                    <o-dd:count>200</o-dd:count>
                </o-ex:constraint>
            </mm:ring>
            <o-ex:requirement>
                <o-dd:prepay>
                    <o-dd:payment>
                        <o-dd:amount o-dd:currency="USD">5.00</o-dd:amount>
                    </o-dd:payment>
                </o-dd:prepay>
                <mm:sms-accept mm:lang="en-au"/>
            </o-ex:requirement>
        </o-ex:permission>
    </o-ex:offer>
</o-ex:rights>
```

Example 19. Mobile Example

6 References

Technical Standards:

[AAP] Digital Rights Management for Ebooks: Publisher Requirements, Version 1.0, Association of American Publishers, 2000.
<http://www.publishers.org/home/drm.pdf>

[DCMI] Dublin Core Metadata Initiative
<http://dublincore.org>

[DIG35] DIG35 Specification – Metadata for Digital Images
<http://www.i3a.org/i_dig35.html>

[DOI] Digital Object Identifier
<http://www.doi.org/>

[EBX] Electronic Book Exchange
<http://www.ebxwg.org/>

[IFLA] Functional Requirements for Bibliographic Records
<http://www.ifla.org/VII/s13/frbr/frbr.htm>

[IMS] IMS Global Learning Consortium
<http://www.imsproject.org/>

[IMT] Internet Media Types
<http://www.isi.edu/in-notes/iana/assignments/media-types/media-types>

[INDECS] Interoperability of Data in Ecommerce Systems
<http://www.indecs.org/>

[ISO3166] Country Names and Code Elements
<http://www.din.de/gremien/nas/nabd/iso3166ma/codlstp1/>

[ISO4217] Currency Names
<http://www.xe.net/gen/iso4217.htm>

[ISO8601] ISO (International Organization for Standardization). Representations of dates and times

[MPEG] Moving Picture Experts Group
<http://mpeg.telecomitalialab.com/>

[MPEG7] MPEG7 Roles List
<http://mpeg.telecomitalialab.com/standards/mpeg-7/mpeg-7.htm>

[MPEG-21] Information Technology — Multimedia Framework — Part 3: Digital Item Identification and Description. ISO/IEC CD 21000-3, December 2001.

[OEBF] OpenEBook Forum
<http://www.openebook.org/>

[ONIX] ONIX International
<http://www.editeur.org/onix.html>

[PRISM] Publishing Requirements for Industry Standard Metadata, 2001
<http://prismstandard.org>

[PROPAGATE] Propagate Project, Access Cooperative Multimedia Centre and Impart Corporation, 1996

[RFC2119] Key words for use in RFCs to Indicate Requirement Levels
<http://www.ietf.org/rfc/rfc2119.txt>

[URI] Uniform Resource Identifiers (URI): Generic Syntax
<http://www.ietf.org/rfc/rfc2396.txt>

[VCARD] vCard MIME Directory Profile
<http://www.ietf.org/rfc/rfc2426.txt>

[WBXML] WAP Binary XML Content Format, W3C NOTE 24 June 1999
<http://www.w3.org/TR/wbxml/>

[XML] Extensible Markup Language 1.0. W3C Recommendation
<http://www.w3.org/TR/REC-xml>

[XML-ENC] XML Encryption Syntax and Processing, W3C Candidate Recommendation, 4 March 2002
<http://www.w3.org/TR/xmlenc-core/>

[XML NAMESPACE] Namespaces in XML, W3C Recommendation
<http://www.w3.org/TR/REC-xml-names/>

[XML SCHEMA] XML Schema Part 1: Structures, Part 2: Datatypes. W3C Recommendation
<http://www.w3.org/TR/xmlschema-1/>
<http://www.w3.org/TR/xmlschema-2/>

[XML-SIG] XML-Signature Syntax and Processing, W3C Recommendation, 12 February 2002
<http://www.w3.org/TR/xmldsig-core/>

Digital Rights Management Papers:

[ERICKSON] A Digital Object Approach to Interoperable Rights Management, John S Erickson, D-Lib Magazine, June 2001, Volume 7 Number 6
<http://www.dlib.org/dlib/june01/erickson/06erickson.html>
<DOI:10.1045/june2001-erickson>

[HIGGS] Rights Management: Managing the Layers of Rights and Roles in the Knowledge Based Economy, Peter Higgs, 2000, IPR Systems Report.
<http://www.iprsystems.com/assets/0.2_Rights_Management.pdf>

[IANNELLA] Digital Rights Management (DRM) Architectures, Renato Iannella, D-Lib Magazine, June 2001, Volume 7 Number 6
<http://www.dlib.org/dlib/june01/iannella/06iannella.html>
<DOI:10.1045/june2001-iannella>

[W3C-DRM] W3C Workshop on Digital Rights Management for the Web, 22-23 January 2001, INRIA - Sophia-Antipolis, France.
<http://www.w3.org/2000/12/drm-ws/>

7 Acknowledgements

The editors wish to acknowledge the following organisations as formal supporters of the ODRL Initiative:

- IPR Systems
- Nokia
- PurpleCast
- Octalis
- Simpsons Solicitors
- OzAuthors
- Pipers
- ARPA
- Vienna University of Economics and Business Administration
- Information Management Australia

The editors wish to specially acknowledge the contributions to this document from Nokia:

- Julian Durand
- Honglang Zhang
- Leon Hurst
- Markku Kontio
- Oliver Bremer

The editors wish to acknowledge the support from the following organisations:

- Real Networks
- IBM
- Adobe
- Panasonic
- MarkAny

The editors gratefully acknowledge feedback to this document from:

- Peter Higgs, IPR Systems
- David Parrott, Reuters
- Robert Bolick, McGraw-Hill
- Judie Mulholland, Florida State University
- Susanne Guth, Vienna University of Economics and Business Administration
- Sheng Mei Shen, Panasonic
- Andy Powell, UKOLN
- Craig A Schultz, Accessticket
- Joseph Reagle Jr, W3C
- Bob Mathews, Adobe
- Rick Jelliffe, Topologi
- Jason Boyer, Adobe
- Robert Lönn, Hi3G Access AB
- Jukka Lewandowski, University of Jyväskylä
- Luo Shihui
- Takeshi Imamura, IBM
- Eiji Takahashi, Panasonic
- Yoichi Takayama and Ahrum Song, WebMCQ

Appendix A: ODRL Expression Language XML Schema (Normative)

The XML Namespace URI for the ***ODRL*** Expression Language version 1.1 is:

http://odrl.net/1.1/ODRL-EX

The actual location of the XML Schema is:

http://odrl.net/1.1/ODRL-EX-11.xsd

The XML Schema is documented at:

http://odrl.net/1.1/ODRL-EX-11-DOC/index.html

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://odrl.net/1.1/ODRL-EX"
        xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
        elementFormDefault="qualified" attributeFormDefault="qualified" version="1.1">
    <xsd:import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-
schema.xsd"/>
    <!-- NOTE: This URI will be updated as the Encryption specification is advanced -->
    <xsd:import namespace="http://www.w3.org/2001/04/xmlenc#"
        schemaLocation="http://www.w3.org/Encryption/2001/Drafts/xmlenc-core/xenc-schema.xsd"/
>
    <xsd:element name="rights" type="o-ex:rightsType"/>
    <xsd:element name="offer" type="o-ex:offerAgreeType"/>
    <xsd:element name="agreement" type="o-ex:offerAgreeType"/>
    <xsd:complexType name="offerAgreeType">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="o-ex:context" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:party" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:asset" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:permission" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:constraint" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:requirement" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:condition" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:choice>
    </xsd:complexType>
    <xsd:complexType name="rightsType">
        <xsd:complexContent>
            <xsd:extension base="o-ex:offerAgreeType">
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                    <xsd:element ref="o-ex:revoke" minOccurs="0" maxOccurs="unbounded"/>
                    <xsd:element ref="o-ex:offer" minOccurs="0" maxOccurs="unbounded"/>
                    <xsd:element ref="o-ex:agreement" minOccurs="0" maxOccurs="unbounded"/>
                    <xsd:element ref="ds:Signature" minOccurs="0"/>
                </xsd:choice>
                <xsd:attributeGroup ref="o-ex:IDGroup"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:element name="context" type="o-ex:contextType"/>
```

```
<xsd:element name="contextElement" abstract="true"/>
<xsd:complexType name="contextType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="o-ex:context" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:contextElement" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:choice>
    <xsd:attributeGroup ref="o-ex:IDGroup"/>
</xsd:complexType>
<xsd:complexType name="partyType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="o-ex:context" minOccurs="0"/>
        <xsd:element ref="o-ex:rightsholder" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:party" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:container" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:asset" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:choice>
    <xsd:attributeGroup ref="o-ex:IDGroup"/>
</xsd:complexType>
<xsd:element name="party" type="o-ex:partyType"/>
<xsd:element name="rightsholder" type="o-ex:rightsHolderType"/>
<xsd:element name="rightsHolderElement" abstract="true"/>
<xsd:complexType name="rightsHolderType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="o-ex:context" minOccurs="0"/>
        <xsd:element ref="o-ex:rightsHolderElement" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:container" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:choice>
    <xsd:attributeGroup ref="o-ex:IDGroup"/>
</xsd:complexType>
<xsd:complexType name="assetType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="o-ex:context"/>
        <xsd:element ref="o-ex:inherit"/>
        <xsd:element name="digest">
            <xsd:complexType>
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                    <xsd:element ref="ds:DigestMethod"/>
                    <xsd:element ref="ds:DigestValue"/>
                </xsd:choice>
            </xsd:complexType>
        </xsd:element>
        <xsd:element ref="ds:KeyInfo"/>
    </xsd:choice>
    <xsd:attributeGroup ref="o-ex:IDGroup"/>
    <xsd:attribute name="type">
        <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
                <xsd:enumeration value="work"/>
                <xsd:enumeration value="expression"/>
                <xsd:enumeration value="manifestation"/>
                <xsd:enumeration value="item"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>
```

```
</xsd:complexType>
<xsd:element name="asset" type="o-ex:assetType"/>
<xsd:complexType name="inheritType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="o-ex:context" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:choice>
    <xsd:attribute name="override" type="xsd:boolean" default="false"/>
    <xsd:attribute name="default" type="xsd:boolean" default="false"/>
</xsd:complexType>
<xsd:element name="inherit" type="o-ex:inheritType"/>
<xsd:element name="permission" type="o-ex:permissionType"/>
<xsd:element name="permissionElement" abstract="true"/>
<xsd:complexType name="permissionType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="o-ex:context" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:permissionElement" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:container" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:constraint" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:sequence" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:requirement" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:condition" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:asset" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:choice>
    <xsd:attribute name="exclusive" type="xsd:boolean" use="optional"/>
    <xsd:attributeGroup ref="o-ex:IDGroup"/>
</xsd:complexType>
<xsd:element name="constraint" type="o-ex:constraintType"/>
<xsd:element name="constraintElement" abstract="true"/>
<xsd:complexType name="constraintType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="o-ex:constraint" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:constraintElement" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:container" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:sequence" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:context" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:choice>
    <xsd:attributeGroup ref="o-ex:IDGroup"/>
    <xsd:attribute name="type" type="xsd:anyURI"/>
</xsd:complexType>
<xsd:element name="requirement" type="o-ex:requirementType"/>
<xsd:element name="requirementElement" abstract="true"/>
<xsd:complexType name="requirementType">
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="o-ex:context" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:requirementElement" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:container" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="o-ex:sequence" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="o-ex:IDGroup"/>
</xsd:complexType>
<xsd:element name="condition" type="o-ex:conditionType"/>
<xsd:element name="conditionElement" abstract="true"/>
<xsd:complexType name="conditionType">
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
```

```
            <xsd:element ref="o-ex:context" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:conditionElement" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:permission" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:constraint" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:container" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:sequence" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attributeGroup ref="o-ex:IDGroup"/>
    </xsd:complexType>
    <xsd:complexType name="revokeType">
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="o-ex:context" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attributeGroup ref="o-ex:IDGroup"/>
    </xsd:complexType>
    <xsd:element name="revoke" type="o-ex:revokeType"/>
    <xsd:complexType name="sequenceType">
        <xsd:sequence>
            <xsd:element ref="o-ex:seq-item" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="order" default="total">
            <xsd:simpleType>
                <xsd:restriction base="xsd:NMTOKEN">
                    <xsd:enumeration value="total"/>
                    <xsd:enumeration value="partial"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:element name="sequence" type="o-ex:sequenceType"/>
    <xsd:complexType name="containerType">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="o-ex:container" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:permission" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:permissionElement" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:constraintElement" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:conditionElement" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:requirementElement" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:rightsHolderElement" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:constraint" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:condition" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:sequence" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:party" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:choice>
        <xsd:attribute name="type" default="and">
            <xsd:simpleType>
                <xsd:restriction base="xsd:NMTOKEN">
                    <xsd:enumeration value="and"/>
                    <xsd:enumeration value="in-or"/>
                    <xsd:enumeration value="ex-or"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attributeGroup ref="o-ex:IDGroup"/>
```

```
    </xsd:complexType>
    <xsd:element name="container" type="o-ex:containerType"/>
    <xsd:complexType name="seqItemType">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="o-ex:container" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:permission" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:permissionElement" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:constraintElement" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:conditionElement" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:requirementElement" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:rightsHolderElement" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:constraint" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:condition" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="o-ex:sequence" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:choice>
        <xsd:attribute name="number" type="xsd:integer" use="required"/>
    </xsd:complexType>
    <xsd:element name="seq-item" type="o-ex:seqItemType"/>
    <xsd:attributeGroup name="IDGroup">
        <xsd:attribute name="id" type="xsd:ID"/>
        <xsd:attribute name="idref" type="xsd:IDREF"/>
    </xsd:attributeGroup>
</xsd:schema>
```

Appendix B: ODRL Data Dictionary XML Schema (Normative)

The XML Namespace URI for the ***ODRL*** Data Dictionary version 1.1 is:

http://odrl.net/1.1/ODRL-DD

The actual location of the XML Schema is:

http://odrl.net/1.1/ODRL-DD-11.xsd

The XML Schema is documented at:

http://odrl.net/1.1/ODRL-DD-11-DOC/index.html

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://odrl.net/1.1/ODRL-DD"
        xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
        xmlns:o-dd="http://odrl.net/1.1/ODRL-DD"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="qualified" version="1.1">
    <xsd:import namespace="http://odrl.net/1.1/ODRL-EX"
        schemaLocation="http://odrl.net/1.1/ODRL-EX-11.xsd"/>
    <!-- Declare all the Permission Elements -->
    <xsd:element name="display" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="print" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="play" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="execute" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="sell" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="lend" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="give" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="lease" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="modify" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="excerpt" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="aggregate" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="annotate" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="move" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="duplicate" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="delete" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="verify" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="backup" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
```

```
    <xsd:element name="restore" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="install" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="uninstall" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <xsd:element name="save" type="o-ex:permissionType" substitutionGroup="o-
ex:permissionElement"/>
    <!-- Declare the Payment Element (used in Requirements) -->
    <xsd:element name="payment">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="amount">
                    <xsd:complexType>
                        <xsd:simpleContent>
                            <xsd:extension base="xsd:decimal">
                                <xsd:attribute name="currency" type="xsd:NMTOKEN" use="required"/>
                            </xsd:extension>
                        </xsd:simpleContent>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="taxpercent" minOccurs="0">
                    <xsd:complexType>
                        <xsd:simpleContent>
                            <xsd:extension base="xsd:decimal">
                                <xsd:attribute name="code" type="xsd:NMTOKEN" use="required"/>
                            </xsd:extension>
                        </xsd:simpleContent>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <!-- Define the dataTypes used for Requirements that use Payment element -->
    <xsd:complexType name="feeType">
        <xsd:complexContent>
            <xsd:extension base="o-ex:requirementType">
                <xsd:sequence>
                    <xsd:element ref="o-dd:payment"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <!-- Declare all the Requirements Elements -->
    <xsd:element name="prepay" type="o-dd:feeType" substitutionGroup="o-
ex:requirementElement"/>
    <xsd:element name="postpay" type="o-dd:feeType" substitutionGroup="o-
ex:requirementElement"/>
    <xsd:element name="peruse" type="o-dd:feeType" substitutionGroup="o-
ex:requirementElement"/>
    <xsd:element name="accept" type="o-ex:requirementType" substitutionGroup="o-
ex:requirementElement"/>
    <xsd:element name="register" type="o-ex:requirementType" substitutionGroup="o-
ex:requirementElement"/>
```

```
    <xsd:element name="attribution" type="o-ex:requirementType" substitutionGroup="o-
ex:requirementElement"/>
    <xsd:element name="tracked" type="o-ex:requirementType" substitutionGroup="o-
ex:requirementElement"/>
    <!-- Declare all the RightsHolder Elements -->
    <xsd:element name="fixedamount" substitutionGroup="o-ex:rightsHolderElement">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="o-ex:rightsHolderType">
                    <xsd:sequence>
                        <xsd:element ref="o-dd:payment"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="percentage" substitutionGroup="o-ex:rightsHolderElement">
        <xsd:simpleType>
            <xsd:restriction base="xsd:decimal">
                <xsd:minInclusive value="0"/>
                <xsd:maxInclusive value="100"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:element>
    <!-- Declare all the Context Elements -->
    <xsd:simpleType name="uriAndOrString">
        <xsd:union memberTypes="xsd:anyURI xsd:string"/>
    </xsd:simpleType>
    <xsd:element name="uid" type="o-dd:uriAndOrString" substitutionGroup="o-ex:contextElement"/
>
    <xsd:element name="role" type="xsd:anyURI" substitutionGroup="o-ex:contextElement"/>
    <xsd:element name="name" type="xsd:string" substitutionGroup="o-ex:contextElement"/>
    <xsd:element name="remark" type="xsd:string" substitutionGroup="o-ex:contextElement"/>
    <xsd:element name="event" type="xsd:string" substitutionGroup="o-ex:contextElement"/>
    <xsd:element name="pLocation" type="xsd:string" substitutionGroup="o-ex:contextElement"/>
    <xsd:element name="dLocation" type="xsd:anyURI" substitutionGroup="o-ex:contextElement"/>
    <xsd:element name="reference" type="xsd:anyURI" substitutionGroup="o-ex:contextElement"/>
    <xsd:element name="version" type="xsd:string" substitutionGroup="o-ex:contextElement"/>
    <xsd:element name="transaction" type="xsd:string" substitutionGroup="o-ex:contextElement"/>
    <xsd:element name="service" type="xsd:anyURI" substitutionGroup="o-ex:contextElement"/>
    <xsd:element name="date" type="o-dd:dateType" substitutionGroup="o-ex:contextElement"/>
    <!-- Declare all the Constraint Elements -->
    <xsd:element name="individual" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="group" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="cpu" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="network" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="screen" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="storage" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
```

```
    <xsd:element name="memory" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="printer" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="software" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="hardware" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="spatial" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="quality" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="format" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="unit" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="watermark" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="purpose" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="industry" type="o-ex:constraintType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="count" type="xsd:positiveInteger" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="range" substitutionGroup="o-ex:constraintElement">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="o-ex:constraintType">
                    <xsd:sequence>
                        <xsd:element name="min" type="xsd:decimal" minOccurs="0"/>
                        <xsd:element name="max" type="xsd:decimal" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="datetime" type="o-dd:dateType" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:simpleType name="dateAndOrTime">
        <xsd:union memberTypes="xsd:date xsd:dateTime"/>
    </xsd:simpleType>
    <xsd:complexType name="dateType">
        <xsd:complexContent>
            <xsd:extension base="o-ex:constraintType">
                <xsd:choice>
                    <xsd:sequence>
                        <xsd:element name="start" type="o-dd:dateAndOrTime" minOccurs="0"/>
                        <xsd:element name="end" type="o-dd:dateAndOrTime" minOccurs="0"/>
                    </xsd:sequence>
                    <xsd:element name="fixed" type="o-dd:dateAndOrTime" minOccurs="0"/>
                </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
```

```
    <xsd:element name="accumulated" type="xsd:duration" substitutionGroup="o-
ex:constraintElement"/>
    <xsd:element name="interval" type="xsd:duration" substitutionGroup="o-ex:constraintElement"/>
    <xsd:element name="recontext" type="xsd:boolean" substitutionGroup="o-
ex:constraintElement"/>
    <!-- Transfer Permission is defined as a ContainerType to enable complete expression of
         rights in the Constraint  -->
    <xsd:element name="transferPerm" substitutionGroup="o-ex:container">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="o-ex:containerType">
                    <xsd:attribute name="downstream" default="equal">
                        <xsd:simpleType>
                            <xsd:restriction base="xsd:NMTOKEN">
                                <xsd:enumeration value="equal"/>
                                <xsd:enumeration value="less"/>
                                <xsd:enumeration value="notgreater"/>
                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:attribute>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Appendix C: Document Change History (Informative)

Major changes from Version 1.0 to 1.1 of the ***ODRL*** Expression Language include:

- The type attribute for Constraints can now only contain URIs.
- Fixed the substitution head elements to be all “abstract” and “anyType” (by default).
- Fixed XML Schema for Substitution group elements having the correct derived data types.
- Updated links to new XML Schemas for W3C XML Digital Signatures and W3C XML Encryption.
- Added <context> to <requirement> element.
- Added <context> to <condition> element.
- Modified the Security Model by replacing the CEK and PlainTextKey with elements from the Digital Signatures specification.
- Added <asset> to PermissionType.
- Added forEachMember type to Constraint (see Section 2.3.3).

Major changes from Version 1.0 to 1.1 of the ***ODRL*** Data Dictionary include:

- The <uid> element can now contain URIs and/or strings. All previous identification schemes have been removed.
- The <dLocation> element can now only contain URIs.
- The <role> element can now only contain URI values. All previous role schemes have been removed.
- The idscheme attribute (<uid> and <role>) have been deprecated.
- Fixed <date> and <datetime> elements to be of the same “dateTime” datatype.
- Modified <count> element to only contain Integer values. The use of <start>, <end>, and <fixed> has been deprecated.
- Modified date elements to contain either date and/or dateTime datatypes.

Other changes include:

- Added new example scenarios for PRISM and MPEG-21.
- Removed section on “XML Minimisation”.