INTERNATIONAL STANDARD

ISO/IEC 21000-19

First edition
2010-06-15

Information technology — Multimedia framework (MPEG-21) —

Part 19: Media Value Chain Ontology

Technologies de l'information — Cadre multimédia (MPEG-21) —

Partie 19: Ontologie de chaîne de valeur de média

Reference number
ISO/IEC 21000-19:2010(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 21000-19 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 21000 consists of the following parts, under the general title *Information technology — Multimedia framework (MPEG-21)*:

— *Part 1: Vision, Technologies and Strategy* [Technical Report]

— *Part 2: Digital Item Declaration*

— *Part 3: Digital Item Identification*

— *Part 4: Intellectual Property Management and Protection Components*

— *Part 5: Rights Expression Language*

— *Part 6: Rights Data Dictionary*

— *Part 7: Digital Item Adaptation*

— *Part 8: Reference Software*

— *Part 9: File Format*

— *Part 10: Digital Item Processing*

— *Part 11: Evaluation Tools for Persistent Association Technologies* [Technical Report]

— *Part 12: Test Bed for MPEG-21 Resource Delivery* [Technical Report]

— *Part 14: Conformance Testing*

— *Part 15: Event Reporting*

— *Part 16: Binary Format*

— *Part 17: Fragment Identification of MPEG Resources*

— *Part 18: Digital Item Streaming*

— *Part 19: Media Value Chain Ontology*

Introduction

Today, many elements exist to build an infrastructure for the delivery and consumption of multimedia content. There was, however, no “big picture” to describe how these elements, either in existence or under development, relate to each other. The aim for the set of standards ISO/IEC 21000 has been to describe how these various elements fit together. New standards as appropriate will be developed while other relevant standards may be developed by other bodies.

The result is an open framework for multimedia delivery and consumption, with both the content creator and content consumer as focal points. This open framework provides content creators and service providers with equal opportunities in the ISO/IEC 21000-enabled open market. This will also be to the benefit of the content consumer providing them access to a large variety of content in an interoperable manner. The vision for ISO/IEC 21000 is to define a multimedia framework *to enable transparent and augmented use of multimedia resources across a wide range of networks and devices* used by different communities.

This part of ISO/IEC 21000 specifies a machine readable ontology of the media value chain defining a minimal set of kinds of intellectual property, the roles of the users interacting with them, and the relevant actions regarding intellectual property among other features.

Information technology — Multimedia framework (MPEG-21) —

Part 19:
Media Value Chain Ontology

1 Scope

This Part of ISO/IEC 21000 describes MPEG-21 Media Value Chain Ontology (MVCO). The MVCO may be used to capture knowledge about media value chains and to represent, in a computer readable way, concepts in the domain and the relationships between those concepts.

This Part of ISO/IEC 21000 consists of seven Clauses and two Annexes. This technology is described in the following sections of this Part of ISO/IEC 21000.

— Model:
the model is described in Clause 6, by way of a narrative description of the Value Chain, its main elements and relations.

— Representation:
the MVCO has been formalised as a normative OWL Ontology, and the description of which is given in this Clause. The description consists of listing the classes, the object properties, the datatype properties, and the class individuals. Classes are described by giving the name, an English definition, the class hierarchy, and the restrictions imposed on the class. The representation is given in Clause 7. Annex B contains the normative OWL (XML/RDF) comprising the entire semantics of the elements in the model.

— Ontology use:
an Informative section is provided with non normative descriptions of use, extensions and an API (Annex A).

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC TR 21000-1, *Information technology — Multimedia framework (MPEG-21) — Part 1: Vision, Technologies and Strategy*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC TR 21000-1 and the following apply.

NOTE Some of these definitions are taken from the Digital Media Project terminology, see Reference [16].

3.1.1
action
process of performing functions

3.1.2
adaptation
IP entity that is a **work** derived or adapted from another work

3.1.3
AdaptationInstance
IP entity that is an example of an identified **AdaptationManifestation**, for example a file

3.1.4
AdaptationInstanceCopy
IP entity that is a copy of an **AdaptationInstance**

3.1.5
AdaptationManifestation
IP entity (object or event) which is an expression of an **Adaptation**

3.1.6
AdaptationManifestationCopy
IP entity that is a **copy** of an **AdaptationManifestation**

3.1.7
adaptor
user who produces an **adaptation** and its **AdaptationManifestations**

3.1.8
anonymous
user whose identity is unknown

3.1.9
broadcast
action that delivers **content** to a **device** in a point-to-multipoint modality

3.1.10
collective
set of two or more **users**

3.1.11
content
one or more **content elements**

EXAMPLE A type of content is a digital item.

3.1.12
content elements
any of the following types of data: **resource**, metadata, nested content, license, IPMP data, IPMP tools, and **use data**

3.1.13
ContentHandler
user who is appointed to act on **content** on behalf of another **user** and within the scope and responsibility of that second user's rights

3.1.14
copy
mechanical reproduction of analogue or digital representation of a given **IP entity**

NOTE In the case of a digital copy the result is of virtually identical quality whilst in the case of an analogue copy the result can vary considerably in quality.

3.1.15
CopyrightException
permission to invoke a right under exceptional circumstances, for example when a particular **fact** is true

3.1.16
CreateWork
action of creating a **work** without any previous **IP entity**

3.1.17
creator
author
user who generates a **work** and makes its **manifestations**

3.1.18
device
combination of hardware and software that allows a **user** to execute functions over **content** and/or **IP entities**

3.1.19
distribute
action of selling, renting or lending

3.1.20
distributor
user who **distributes** a **product**

3.1.21
download
action of transferring a file or program from a central computer to a smaller computer or to a computer at a remote location

3.1.22
EndUser
user in a **valuechain** who ultimately consumes **content**

3.1.23
EndUser action
action performed by an **EndUser**

3.1.24
fact
positive proposition

3.1.25
identify
function of assigning a unique signifier that establishes the identity of entities, devices, content and content elements

3.1.26
instance
IP entity (object or event) which is an example of an **identified manifestation** such as a file

3.1.27
instantiator
user who produces an **instance**

3.1.28
IP
intellectual property
any identifiable product of the mind attributable to any person(s) or one or more legal entities that can be represented or communicated physically and protectable by copyright or similar laws

3.1.29
IP entity
type of **IP represented** by **content**: **work**, **adaptation**, **manifestation**, **instance**, **product**

3.1.30
MakeAdaptation
action of making an **adaptation**

3.1.31
MakeAdaptationManifestationCopy
action of making an **AdaptationManifestationCopy**

3.1.32
MakeAdaptationInstanceCopy
action of making an **AdaptationInstanceCopy**

3.1.33
MakeAdaptationInstance
action of making an **instance** from an **AdaptationManifestation**

3.1.34
MakeAdaptationManifestation
action of making an **AdaptationManifestation**

3.1.35
MakeCopy
action of making a **copy**

3.1.36
MakeInstance
action of making an **instance** from a **manifestation**

3.1.37
MakeManifestation
action of making a **manifestation**

3.1.38
MakeWorkInstanceCopy
action of making a **WorkInstanceCopy**

3.1.39
MakeWorkManifestationCopy
action of making a **WorkManifestationCopy**

3.1.40
MakeWorkInstance
action of making an **instance** from a **WorkManifestation**

3.1.41
MakeWorkManifestation
action of making a **manifestation** of a work

3.1.42
manifestation
IP entity (object or event) which is an expression of a **work**

3.1.43
permission
authorisation from one **rights** owner to one or more **users** to realise one or more **actions** on a given **IP entity**

3.1.44
private copy
action of storing content and holding it privately for non-commercial purposes

3.1.45
produce
action of making **products**

3.1.46
producer
user that makes **products**

3.1.47
product
IP entity that adds value to **IP entities** by including them with an appropriate licence for the purpose of publishing

3.1.48
public communication
action of publicly displaying or performing

EXAMPLE Public communication can include live performance, radio, television, internet streaming, multicast of **instances** and **manifestations**, and download.

3.1.49
render
action of converting a **resource** to a human-perceivable form

3.1.50
represent
expressing information in a form that can be processed by either a digital or analogue **device**

3.1.51
resource
data, whose **representation** is not specified by ISO/IEC 21000 (e.g. an MP3 file or an executable code), that can be processed by a **device**

3.1.52
right
ability of performing one or more functions over **IP entities** as a consequence of ownership or **permissions**

3.1.53
role
defined set of **actions** and corresponding conditions attributed to, and required of, a **user**

3.1.54
synchronise
action of concurrently performing or displaying two or more distinct **IP entities** each for a different human sense, for example text and audio, or video and song

3.1.55
use data
data documenting the functions performed by a **device** on a **content** item and the associated context

3.1.56
valuechain
group of interacting **users**, connecting (and including) **creators** to **EndUsers**

3.1.57
work
IP entity that is an original or derived creation that retains intellectual or artistic attributes independently of its Manifestations

3.1.58
WorkInstance
IP entity (object or event) which is an example of an **identified manifestation** of a **work**

EXAMPLE A file is a WorkInstance.

3.1.59
WorkInstanceCopy
IP entity copy of a **WorkInstance**

3.1.60
WorkManifestation
IP entity (object or event) which is an expression of a **work**

3.1.61
WorkManifestationCopy
IP entity copy of a **WorkManifestation**

3.2 Abbreviated terms

B2B	Business to Business
B2C	Business to Consumer
IP	Intellectual Property
MVCO	Media Value Chain Ontology
OWL	Web Ontology Language
OWL-DL	OWL Description Logic
RDF	Resource Description Framework
REL	Rights Expression Language
URI	Uniform Resource Identifier

URL	Uniform Resource Locator
URN	Uniform Resource Name
W3C	World Wide Web Consortium
XML	Extensible Markup Language

4 Conventions

4.1 Documentation conventions

The reader is informed that terms beginning with a capital letter (like "Work") are used according to definitions given in Clause 3. Terms referred to in the ontology elements are given with its namespace (e.g. `mvco:Work`).

4.1.1 Class description

7.2.4 provides a systematic enumeration of the MVCO classes. Some conventions are explained in this subclause. For each OWL class, the following features may be given:

- **Class name**. The OWL class name, given as an `owl:Class` in the Ontology. Class names follow the convention of using an uppercase for the first letter, and leaving no blankspaces or dashes between words if more words are included. Qualified names start with the ontology URI, like `http://purl.oclc.org/NET/MVCO.owl#User`
- **Comment**. The English definition, given as a `rdfs:comment` in the Ontology.
- **Restrictions**. A set of expressions with an accompanying human explanation that must hold for an individual to belong to the class in question. Each of these propositions is either *necessary* (introduced by the symbol ⊆) or *necessary and sufficient* (introduced by the symbol ≡). The former is a condition that each class individual must satisfy, and the later is a condition which, if held, is enough to state that the individual belongs to this class. Each restriction is given as a parent class of the class in question, and can be one of the following:
  - **Existential restriction ("some").** Expressed by means of the `owl:someValuesFrom` OWL element and represented here by the symbol ∃, it expresses the fact that an individual of this class **must be related to at least one** individual of the given Range class. This is a *quantifier restriction*.
  - **Universal restriction ("only").** Expressed by means of the `owl:allValuesFrom` OWL element and represented here by the symbol ∀, it expresses the fact that an individual of this class **can only** be related to individuals of the given Range class. This is a *quantifier restriction*.
  - **Cardinality restrictions ("min", "max,", "exactly").** Determine the number of relationships that an individual of this class must participate in (by giving a minimum, a maximum or an exact number).
  - **Disjoint classes restriction.** Classes can be required to be disjoint (i.e., a class individual may not be allowed to belong to two given classes at the same time.), but disjoint classes have not been represented in the enumeration.
- **In-domain-of.** Object Properties for which the given class is the *Domain*, if any.
- **Known subclasses.** Graphical representation of the known subclasses, if any.

Classes can be joint with the union operator, so that (`mvco:Creator` ∪ `mvco:Adaptor`) means the set of individuals which are either Creator or Adaptor individuals (or both). The intersection operators ∩ defines individuals belonging to both classes. Propositions can use the operators AND, OR and NOT (¬).

4.1.2 Property description

7.4 lists all the defined relations in the ontology. For each relation the following features may be given:

- **Name.** The name of the relation. Properties follow the convention of including as a first word an English verb starting with a lowercase letter, followed by subsequent words starting with an uppercase letter and no blankspaces or dashes in between.
- **Description.** Description of the relation, given as an English text in the `rdfs:comment` element.
- **Domain of the property.** Limits the individuals to which the property can be applied.
- **Range of the property.** Limits the individuals that the property may have as its value. In datatype properties, only literal values are accepted.
- **Characteristics of the property**. These give a more precise description of the relation, stating whether the property is symmetric, transitive, functional or inverse of the other property.

Object properties are represented in Figure 1 and Figure 3 of this document following this convention:

- Arcs represent object properties, the name of which is next to its head. The cardinality is given next to its tail, indicating how many class individuals can be related to one individual. For example, `1.n` indicates that one individual can be linked with this property with one or more individuals.
- Source nodes are the domain of the object property.
- Sink nodes are the range of the object property.

4.2 Namespace prefix conventions

MVCO includes an initial `rdf:RDF` component with some XML namespace declarations. Namespaces provide a means to unambiguously interpret identifiers and make the rest of the ontology presentation much more readable. It also contains the base URI for this document and the default namespace, which has been chosen as the URI of the document containing the ontology.

The ontology URI shall be `http://purl.oclc.org/NET/mvco.owl`. This is a permanent URI deemed to provide a permanent reference regardless of the actual location of the MVCO ontology. The following namespace conventions have been adopted:

Table 1 — Namespace prefixes

Namespace prefix	Namespace
None (default namespace)	http://purl.oclc.org/NET/mvco.owl#
xsd	http://www.w3.org/2001/XMLSchema#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
owl	http://www.w3.org/2002/07/owl#
dc	http://purl.org/dc/elements/1.1/
dii	urn:mpeg:mpeg21:2002:01-DII-NS#
mx	urn:mpeg:mpeg21:2003:01-REL-MX-NS#
daml	http://www.daml.org/2001/03/daml+oil#

Dublin Core annotations (ISO 15836-2003) have been made to the ontology:

Annotation	Value
dc:title	Media Value Chain Ontology
dc:language	en

5 Relationship to other ISO/IEC 21000 Parts

The Digital Item is the fundamental unit of distribution and transaction in the Multimedia framework. While the different parts of ISO/IEC 21000 deal with the components and different aspects of Digital Items, together they form a complete integrated interoperable framework. This Clause describes the relationship of this part of ISO/IEC 21000 with other parts of ISO/IEC 21000 in addressing the Intellectual Property (IP) represented by Digital Items and their elements as well as how such IP is subsequently handled by Users.

The MVCO represents a model integrating different elements of the ISO/IEC 21000 interoperable framework.

MVCO defines different types of objects subject to intellectual property; these IP Entities are represented through Digital Items or parts thereof defined in ISO/IEC 21000-2. IP Entities represented as MVCO IP Entity class instances are identified with the Digital Item Identifiers as described in ISO/IEC 21000-3. MVCO User class instances refer to Users as conceived in ISO/IEC 21000-1.

Furthermore, MVCO based applications makes use of ISO/IEC 21000-15 (Event Reporting), e.g. events in MVCO trigger Event Reports (ISO/IEC 21000-15), and ISO/IEC 21000-5 (Rights Expression Language), e.g. to express permissions.

Also, ISO/IEC 21000-6 Rights Data Dictionary (RDD) and MVCO are complementary within the MPEG-21 Framework and their vocabularies may be jointly used within MVCO based applications according to users needs. Many relationships between the elements in these two standards may be established without logical contradictions or interpretative ambiguities.

6 Media Value Chain Model

6.1 Introduction

This Clause is intended to facilitate a high level understanding of the MVCO core model. Although all aspects referenced are normative, the complete formal description is given in Clause 7 of this document.

The Media Value Chain Ontology is designed to represent common aspects of content creation, distribution, use and handling. Central to this is the need to adhere to basic common notions related to Intellectual Property.

6.2 Fundamentals of Intellectual Property

Any notion of Intellectual Property implies the existence of a minimum and necessary set of entities, roles, rights and actions, each a corollary of the other. The objective and scope of the MVCO is to represent a minimum set in a machine readable fashion either directly in a core model or through appropriate extensions.

While there are clear differences in the legal treatment of IP between different jurisdictions, this does not mean that a common core between them does not exist. The existence of such a core is reflected by the numerous International World Intellectual Property Organization (WIPO [1)]) treaties such as the Berne Convention where if it were not for a clear understanding of common terms such as *work*, *adaptation*,

1) The World Intellectual Property Organization (WIPO) defines Intellectual Property as *the creations of the mind: inventions, literary and artistic works, and symbols, names, images, and designs used in commerce*.

performance etc., it would not be possible for such a treaty to be assumed by 163 different national jurisdictions. Presently, this common IP model lacks a standard computer representation that the MVCO provides.

The kind of IP Entity represented by a Digital Item will be specified in the context of the MVCO, so that it is possible that the same bits of a Digital Item may represent different kinds of IP (e.g., a Manifestation or Product).

6.3 Description of the IP Entity relations within the Value Chain

As far as IP Entities are concerned, and for each IP Entity, Users have Roles associated to them (Creator, Adaptor, Instantiator, Producer...) that represent the set of Actions that can be exercised on specific corresponding IP Entities.

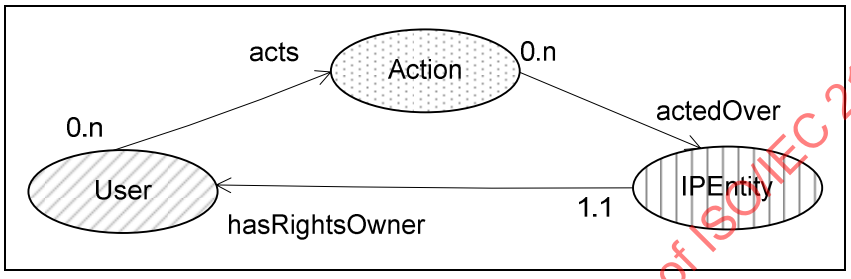


Figure 1 — Relations between Actions, Users and IP Entities

Figure 1 illustrates the fundamental relations. A User (subject) "acts" Action (verb), and Action is actedOver an IP Entity (object). Finally, the IP Entity has a User as its rights owner. Each arc is labeled with the corresponding relation with cardinality expressed in the direction indicated by the arrow heads as 0.1 (minimum number of arrow is 0 and maximum is 1), 0.n (0 to any number) and 1.1 (one to one).

The origin of any IP Value Chain necessarily consists of the creation and subsequent manifestation of the original IP Entity referred to as a Work, the rights over which are exclusively of the Work's author(s) unless as otherwise determined by the author(s). This original IP Entity is subject to be used to create new dependent IP Entities leading to what is referred to as a *Value Chain*. Rights for the exploitation of IP Entities can be transferred along this value chain.

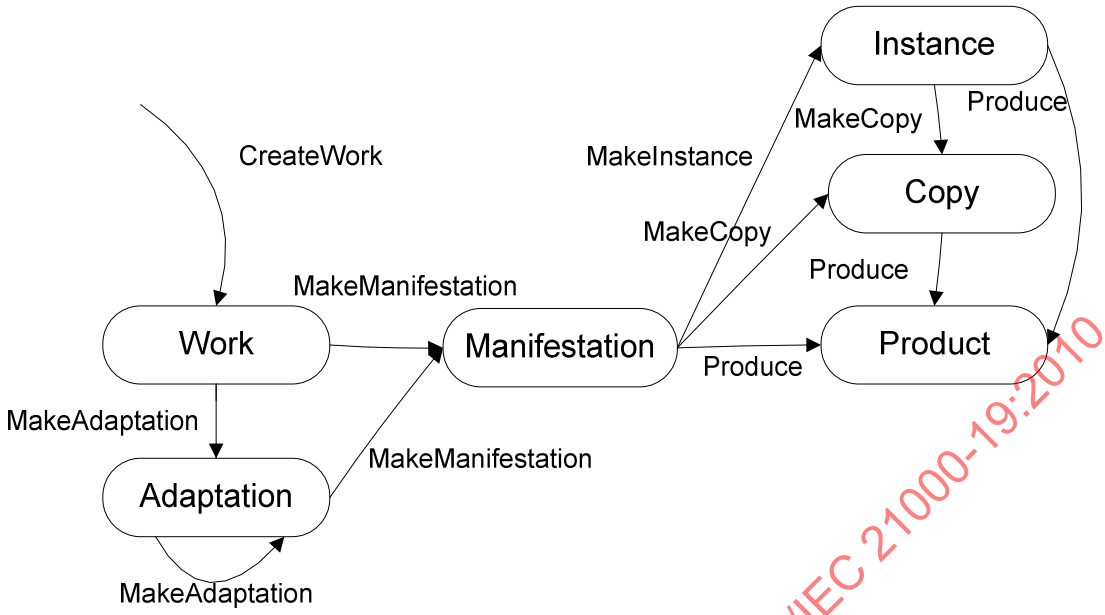


Figure 2 — Main IP Entities and Relating Actions in the IP Value Chain

The Figure above illustrates different IP Entities and the Actions that give rise to them, the formal representation of which is given in Clause 7. Here, “Create Work” gives rise to “Work”, then by virtue of the Action “MakeAdaptation” an “Adaptation” is made (from which other “Adaptations” can be generated). “MakeManifestation” brings about “Manifestations” of both “Works” and “Adaptation”. “Manifestations” gives rise to the possibility of the Actions “Produce”, “MakeCopy”, “MakeInstance” that in turn may give rise to “Products”, “Copies” and “Instances” respectively.

Although not shown in the diagram, some IP Entities can be further specialiced, e.g. the Manifestation of a Work is defined as WorkManifestation, while Manifestation of an Adaptation is defined as AdaptationManifestation. Both are specializaitons of the more general concept Manifestations. The same can be said about the Actions in the figure, where “MakeWorkManifestation” and “MakeAdaptationManifestation” are specialisations of MakeManifestation.

Figure 3 represents the main classes in the Value Chain linked by the relationships “acts”, “actedOver” and “resultsIn”. Each node represents a class (Actions in green, IPEntities in blue and Users in orange), whilst each arc represents one of the object properties mentioned in 4.1.2. Note that for clarity, MakeManifestation is common to Creator and Adaptor but each case refers to the appropriate sub classes MakeWorkManifestation and MakeAdaptationManifestation.

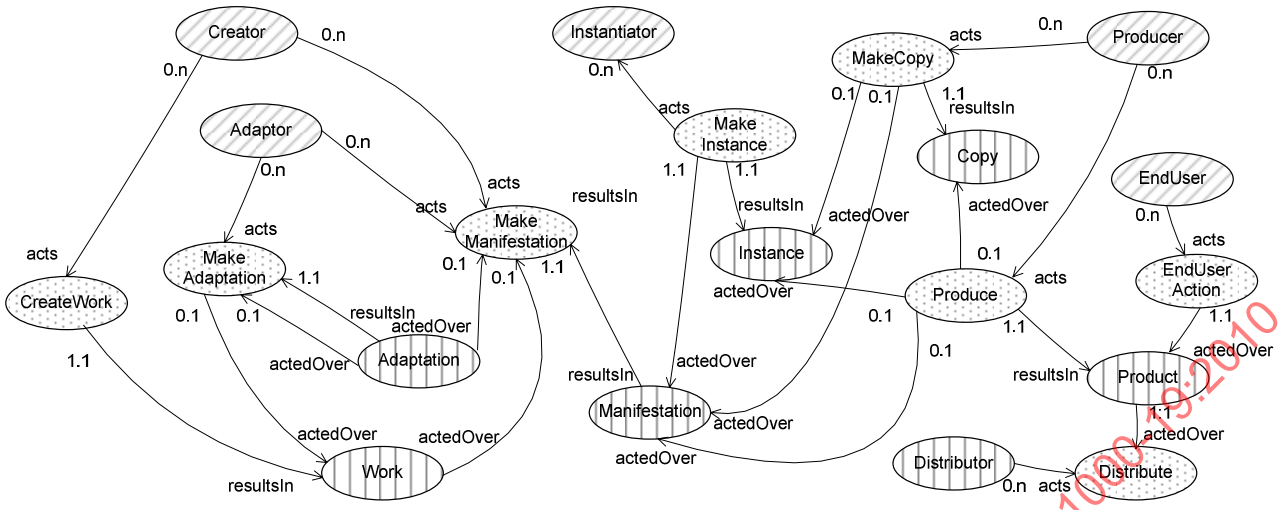


Figure 3 — Diagram of classes connected by the relationships "acts", "actedOver" and "resultsIn"

In Figure 3, dotted pattern represent Actions, vertical lines represent IP Entities and diagonal lines represent User roles.

6.4 IP Entities

All entities subject to Intellectual Property are said to be IP Entities. The natural sequence of dependence between IP Entities begins with the IP Entity Work and ends with Product from which copies can be made and distributed. Intermediate IP Entities are Adaptation, Manifestation, Instance and corresponding Copies, as can be seen in Figure 2. Each of these can be further specialized in extensions of the MVCO [audio, video, image, text etc.]. Their definition is given in 3.1 in particular:

- Work. (See 3.1.57)
- Adaptation. (See 3.1.2)
- Manifestation. (See 3.1.42)
- Instance. (See 3.1.26)
- Copy. (See 3.1.14)
- Product. (See 3.1.47)

6.5 Singular vs Composite IP Entities

IP Entities may be singular objects while others maybe be made up of a combination of existing IP Entities (e.g. a film has a script, a score, soundtrack etc.). We speak of these as "Composite IP Entities". Usually a Composite IP Entity will have several rights owners (Collective, (3.1.10) made up of the individual Creators of each component.).

6.6 Use Data

Monitoring and exploiting Use Data may require the explicit authorisation from the User whose use data has been collected. Since Use Data is not an IP Entity, a `mvco:UseData` class has been defined at the root-level. Use Data can be further specialised as required in extensions of the MVCO.

6.7 Users (Single vs Collective)

In ISO/IEC TR 21000-1, a user is defined as an *entity that interacts in the MPEG-21 environment or makes use of a Digital Item. Such Users include individuals, consumers, communities, organisations, corporations, consortia, governments and other standards bodies and initiatives around the world.*

From the above it is clear that the class Users includes both single individuals as well as groups of individuals. Moreover, it happens that a given User sometimes acts as part of a Collective of Users, in the sense that several decision-enabled parties share jointly the responsibility for a given IP Entity or collectively hold certain rights. The MVCO model explicitly acknowledges this by defining the specialisaition of User called Collective.

6.8 Roles

6.8.1 General

In order to operate within the MVCO and with respect to particular IP Entities, the relationship between a User and a particular IP Entity type is specified formally in the ontology as specialisations of the Class User i.e. Creator, Adaptor, Instantiator, Producer, Distributor and End-User and refer to the concept of Role. Thus and depending on the IP Entiety in question any given User may take on any number of Roles within the MVCO.

Figure 4 illustrates the core Roles and the IP Entity by which they are linked. Definitions are given in Clause 3.1.

- Creator. (See 3.1.17)
- Adaptor. (See 3.1.7)
- Instantiator. (See 3.1.27)
- Producer. (See 3.1.46)
- Distributor. (See 3.1.20)
- EndUser. (See 3.1.22)

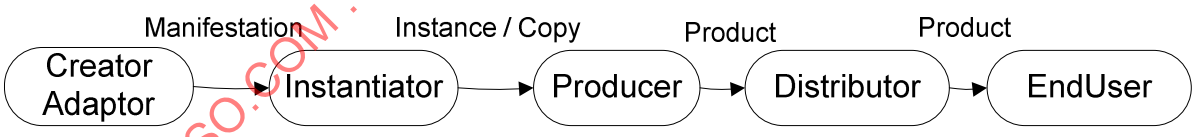


Figure 4 — Roles and corresponding IP Entities along the Value Chain

The figure above, illustrates how Roles are linked through the transmission of IP Entities, “Creators” and “Adaptors” produce “Manifestations” that are communicated and become known such that “Instantiators” can make “Instances” and “Copies” etc. This set of core Roles can be extended to include further specialisations. For example, from Distributor, two specialized Roles could be distinguished: ContentAggregator who may operate in a B2B oriented model, and ContentProviders operating in a B2C oriented model. Both roles would be specialisations of the generic Distributor Role and their definition would be wholly consistent with the MVCO Core model.

6.8.2 Content Handlers

On occasions, Users delegate tasks for which they are ultimately responsible in other Users called Content Handlers. This Role has no Rights associated with it other than those conferred solely by virtue of its “surrogate” function of the User whose auspices it operates on Content. Thus, the MVCO model explicitly acknowledges the class ContentHandler.

6.9 Actions

Actions are the process of doing something over IP Entities. Actions can be over the IP Entities themselves or their representations be they either analogue or digital.

The Media Value Chain Ontology recognizes all actions affecting Intellectual Property. Besides creation of an original Work, Actions that concern Intellectual Property are those that lead to the transformation of IP Entities, their distribution, public communication and consumption.

The result of some Actions may imply the creation of new IP Entities (for example, a MakeAdaptation Action generates a new IP Entity of the kind Adaptation) while others do not as in the case of Render.

Each of the Actions are executed by Users

- CreateWork. (See 3.1.16)
- MakeAdaptation. (See 3.1.30)
- MakeManifestation. (See 3.1.37)
- MakeInstance. (See 3.1.36)
- MakeCopy. (See 3.1.35) Similar to mechanical reproduction.
- Produce. (See 3.1.45)
- Distribute. (See 3.1.19)
- Public communication. (See 3.1.48)
- Synchronise. (See 3.1.54) Synchronisation implies the simultaneous rendition of two IP Entities requiring the additional Permission of the Creator.
- EndUserAction. (See 3.1.23)

6.10 Permissions

6.10.1 General

Transfer of Rights are represented in Permissions. A Permission relates an IP Entity with the transmitted Right, the original Rights owner and the new Rights owner. A Permission may require the prior satisfaction of conditions. Requirements in Permissions are expressed as Facts, which are simply defined as positive propositions with a binary truth value. A prohibition is thus expressed as the negation of a particular Fact.

Permissions are generally transferred from User to User consecutively along the Value Chain in the normal course of business. However, in some cases Permissions may be required directly (i.e. not passed down through relay) from the Creator to a User in order to authorise certain Actions, Examples of such cases are Actions that require specific authorisation by the Creator each and everytime they are excercised by virtue of the Creators´s moral rights such as in the case of PublicCommunication.

ISO/IEC 21000-5 licenses represent a set of Permissions that can be expressed in many ways in the context of extensions of MVCO.

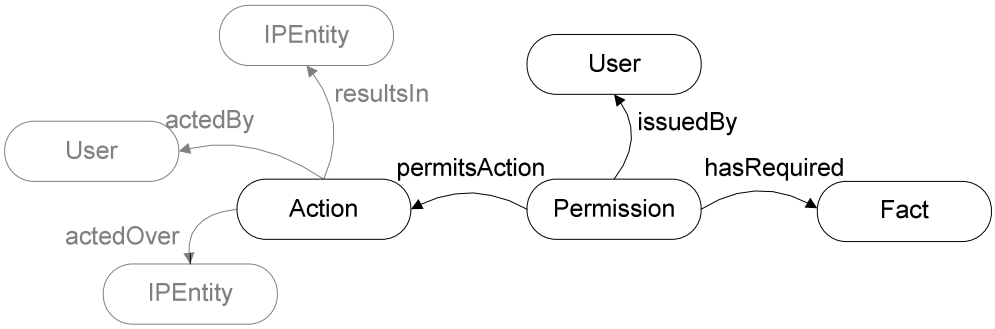


Figure 5 — Relations of Permissions with other MVCO elements

Figure 5 expresses how Permissions are related to other MVCO elements. Required Facts and the Action that is permitted is directly associated with the User that issues the Permission. The Action in turn and by its corresponding associations defines over which IP Entity the permission is applied, by which User it can be Acted and the type of IP Entity it may generate etc.

6.10.2 Copyright Exceptions

Some rights can be invoked by virtue of certain conditions (Facts) being met. For example, complete quotes are allowed for scientific purposes. The MVCO Ontology provides mechanisms for specifying such copyright exceptions, although the exceptions themselves are not specified. Permissions from one User to another are not needed to invoke a CopyrightException, CopyrightExceptions are given based on the existence of the corresponding CopyrightExceptionFact attributable to particular IP Entities and/or Users.

7 Media Value Chain Representation

7.1 Introduction

The core model presented so far has been represented as a Web Ontology Language (OWL) ontology. Its normative representation is the XML/RDF codification.

The Web Ontology Language is a specification from the World Wide Web Consortium (W3C) based on the Resource Description Framework (RDF), keystone of the Semantic Web. Their codification can be as simple as a single XML file which can be universally accessed. By using this model supported by existing technology standards, companies can create end-to-end applications based on the MVCO Core Model described in Clause 6 of this document.

The MVCO Core Model is fully described by giving a set of classes and properties of those classes. Clause 7 of this document provides an exhaustive review of all these classes and properties.

7.2 Ontology General Features

7.2.1 Ontology Sublanguage

OWL ontologies can be categorised into three species or sub-languages: OWL-Lite, OWL-DL and OWL-Full, with increasing level of expressivity. OWL-DL is based on *Description Logic*, a decidable fragment of First Order Logic which grants automated reasoning in a finite time.

The MVCO is consistent and uses the DL subset of the OWL language, granting thus its decidability and its usability in practical applications.

7.2.2 Ontology Metrics

The main features of the MVCO ontology are summarized in Table 2.

Table 2 — Main features of MVCO

Property	Value	Property	Value
owl:versionInfo	1.0	No. of functional properties	6
Total No. of classes	55	No. of inverse functional properties	0
Total No. of datatype properties	5	No. of transitive properties	2
Total No. of object properties	13	No. of symmetric properties	0
Total No. of annotation properties	4	No. of inverse properties	6
Total No. of individuals	1	Max depth of property tree	1
No. of disjoint axioms	68	Max dept of class tree	3
Expressivity	SIF(D)[2)]	Avg. branching factor of class tree	6.75

7.2.3 Ontology Versioning

Classes and properties have been annotated with the `owl:versionInfo` tag. This tag version is 1.0 in the first edition of the International Standard. Possible corrigenda to the standard should upgrade the decimal (e.g. 1.1, 1.2 etc.), while possible ammendments would increase the version number (e.g. 2.0 etc.). Further ammendments of this standard may introduce new classes and relationships with higher versions, but they should not deprecate the existing axioms (e.g. no `owl:class` will be turned out into `owl:deprecatedclass` etc.).

7.2.4 Extension mechanisms

The Semantic Web allows any entity to post, reuse and extend OWL ontologies to suit their needs. MVCO foresees extensions of the ontology, for which the following extension mechanisms are listed:

- *Addition of new classes*. Subclasses can be created and inherit all the features of the parent class while allowing refining concepts for more specialised purposes.
- *Adding new relations*. New object properties can be defined, either at root level or derived from other object properties.
- *Adding new properties*. New datatype properties can be defined at will.
- *Creating individuals*. The extended ontology may define individuals.

Any ontology extending MVCO that is consistent can be said to be MVCO compatible. The built-in MVCO properties and classes should not be redefined. In general, this means that elements from the *mvco* namespace should not appear as *subjects* of triples (only as *objects*).

7.2.5 Ontology alignment

Other ontologies may want to map some of their concepts to those in MVCO. Individuals of one ontology may be mapped to individuals of another, through the use of the construct `owl:sameAs`, but classes should not be matched (if classes were matched with `owl:sameAs`, the ontology would result in a OWL Full).

2) S represents a transitive Base Description Logic (Attributive Language with Complements, (transitive ALC). It indicates that inverse operators are used in the ontology. F represents the functional attribute of relations (that is to say, when a relation can at most be present only once for an individual). (D) represents the use of datatypes. SIN logics satisfy the finite model property, finite table can be built and logic is thus granted to be decidable.

Ontology alignment is useful in scenarios sharing some concepts with the MVCO and applications may use both ontologies having matched certain terms. For example, the Music Ontology focuses in a particular media – Music – and defines a workflow whose main concepts can be assimilated by some concepts in the MVCO.

7.3 Description of MVCO Classes

7.3.1 User-related classes

class	**User**
rdfs:comment	Any person or legal entity in a Value-Chain connecting (and including) Creator and EndUser.
rdfs:subClassOf	owl:Thing
Restrictions	**Expression** / **Meaning** ⊆ ∀ acts . Action — A User may only act Actions ⊆ ∀ belongsTo . Collective — A User may only belong to Collectives
in-domain-of	isRightsOwnerOf (IPEntity) belongsTo (Collective) acts (Action) socialTag (String)
in-range-of	actedBy hasRightsOwner issuedBy actOnBehalfOf rightGivenBy
Known subclasses	
Individuals	Anonymous

class	**Collective**
rdfs:comment	Set of two or more Users.
rdfs:subClassOf	mvco:User
Restrictions	None

class	**ContentHandler**
rdfs:comment	A User who is appointed to act on Content on behalf of another User and within the scope and responsibility of that second User´s rights
rdfs:subClassOf	mvco:User
Domain	actOnBehalfOf (User)
Restrictions	**Expression** / **Meaning** ≡ ∃ actOnBehalfOf . User — A ContentHandler is any User who acts on behalf a User.

class	**Creator**
rdfs:comment	A User who generates a Work and makes its first Manifestation, also referred to as author
rdfs:subClassOf	mvco:User
Restrictions	**Expression** / **Meaning** ≡ ∃ acts . CreateWork — A Creator is a User who has created a Work

class	**Adaptor**	
rdfs:comment	A User who produces an Adaptation	
rdfs:subClassOf	mvco:User	
Restrictions	**Expression**	**Meaning**
	≡ ∃ acts . MakeAdaptation	An Adaptor is a User who performs a MakeAdaptation

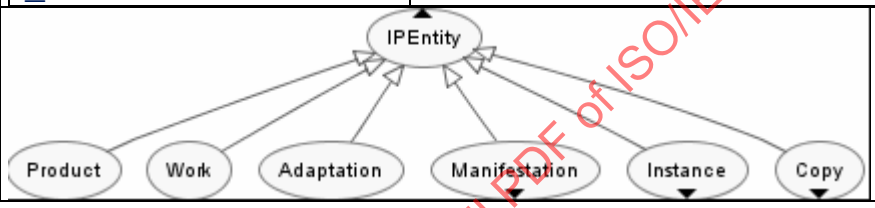
class	**Instantiator**	
rdfs:comment	A User who produces an Instance	
rdfs:subClassOf	mvco:User	
Restrictions	**Expression**	**Meaning**
	≡ ∃ acts . MakeInstance	An Instantiator is a User who has executed a MakeInstance

class	**Producer**	
rdfs:comment	A User that makes Products	
rdfs:subClassOf	mvco:User	
Restrictions	**Expression**	**Meaning**
	≡ ∃ acts . Produce	A Producer is a User who has executed a Produce Action.

class	**Distributor**	
rdfs:comment	A User who distributes a Product	
rdfs:subClassOf	mvco:User	
Restrictions	**Expression**	**Meaning**
	≡ ∃ acts . Distribute	A Distributor is a User who has executed a Distribute Action

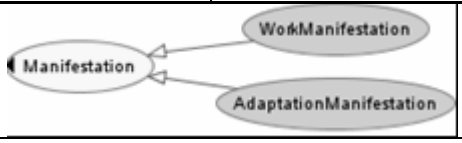
class	**EndUser**	
rdfs:comment	A User in a Value Chain who ultimately consumes Content	
rdfs:subClassOf	mvco:User	
Restrictions	**Expression**	**Meaning**
	≡ ∃ acts . EndUserAction	An EndUser is a User who has executed any of the EndUserActions.

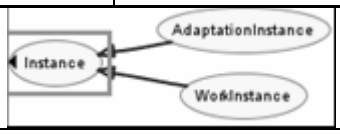
7.3.2 IP Entities and UseData

class	**IPEntity**
rdfs:comment	Types of IP Represented as Content: Work, Adaptation, Manifestation, Instance, Product
rdfs:subClassOf	owl:Thing
in-domain-of	hasRightsOwner (User) isMadeUpOf (IPEntity) resultedFrom (Action) dii:RelatedIdentifier (URI) socialTag (String)
Restrictions	**Expression** / **Meaning** ⊆ ∀ hasRightsOwner User ⊆ hasRightsOwner exactly 1 — An IP Entity has one and only one User as rights owner. Note: anonymous Users are allowed by using the defined mvco:Anonymous individual. ⊆ ∀ isMadeUpOf IPEntity — A Composite IP Entity may be made of several IPEntities. ⊆ ∀ resultedFrom Action — IPEntities are created as results of Actions
Known subclasses	

class	**Work**
rdfs:comment	An IP Entity that is an original or derived creation that retains intellectual or artistic attributes independently of its Manifestations
rdfs:subClassOf	mvco:IPEntity
Restrictions	**Expression** / **Meaning** ⊆ ∀ resultedFrom CreateWork — A Work is only the result of a CreateWork Action

class	**Adaptation**
rdfs:comment	An IP Entity that is a Work derived from another Work or Adaptation
rdfs:subClassOf	mvco:IPEntity
Restrictions	**Expression** / **Meaning** ⊆ ∀ resultedFrom MakeAdaptation U Synchronise) — An Adaptation derives only from the execution of a MakeAdaptation or a Synchronise Action

class	**Manifestation**
rdfs:comment	An IP Entity (object or event) which is an expression of a Work
rdfs:subClassOf	mvco:IPEntity
Restrictions	**Expression** / **Meaning** ⊆ ∀ resultedFrom MakeManifestation — A Manifestation derives from the execution of a MakeManifestation Action
Known subclasses	

class	**Instance**	
rdfs:comment	An IP Entity (object or event) which is an example of an Identified Manifestation (e.g. a file)	
rdfs:subClassOf	mvco:IPEntity	
Restrictions	**Expression**	**Meaning**
	⊆ ∀ resultedFrom MakeInstance	An Instance derives from the execution of a MakeInstance Action
Known subclasses		

class	**Copy**	
rdfs:comment	A mechanical reproduction of analogue or digital representations of a given IP Entity. In the case of digital Copies the result is virtually identical quality while in the case of analogue Copies the results can vary considerably in quality.	
rdfs:subClassOf	mvco:IPEntity	
Restrictions	**Expression**	**Meaning**
	⊆ ∀ resultedFrom MakeCopy	A Copy derives from the execution of a MakeCopy Action
Known subclasses		

class	**Product**	
rdfs:comment	An IP Entity that adds value to IP Entities by including them with an appropriate licence for the purpose of publishing	
rdfs:subClassOf	mvco:IPEntity	
Restrictions	**Expression**	**Meaning**
	⊆ ∃ resultedFrom. Produce	A Product derives from the execution of a Produce Action

class	**UseData**
rdfs:comment	Data documenting the Actions performed by a User on a content item and the associated context
rdfs:subClassOf	owl:Thing

class	**AdaptationManifestation**	
rdfs:comment	An IP Entity (object or event) which is an expression of an Adaptation	
rdfs:subClassOf	mvco:Manifestation	
Restrictions	**Expression**	**Meaning**
	≡ (∃ resultedFrom MakeAdaptationManifestation ∩ Manifestation)	An AdaptationManifestation is every Manifestation resulting from executing a MakeAdaptationManifestation Action.

class	**WorkManifestation**	
rdfs:comment	An IP Entity (object or event) which is an example of an Identified Manifestation of a Work (e.g. a file)	
rdfs:subClassOf	mvco:Manifestation	
Restrictions	**Expression**	**Meaning**
	≡ (∃resultedFrom MakeWorkManifestation ∩ Manifestation)	A WorkManifestation is the Manifestation resulting of executing a MakeWorkManifestation action.

class	**WorkInstance**	
rdfs:comment	An IP Entity (object or event) which is an example of an Identified Manifestation of a Work (e.g. a file))	
rdfs:subClassOf	mvco:Instance	
Restrictions	**Expression**	**Meaning**
	≡ (∃ resultedFrom MakeWorkInstance ∩ Instance)	A WorkInstance is every Instance coming from a WorkManifestation.

class	**AdaptationInstance**	
rdfs:comment	An IP Entity which is an example of an Identified Adaptation Manifestation (e.g. a file)	
rdfs:subClassOf	mvco:Instance	
Restrictions	**Expression**	**Meaning**
	≡ (∃ resultedFrom MakeAdaptationInstance ∩ Instance)	An AdaptationInstance is every Instance coming from a AdaptationManifestation.

class	**AdaptationInstanceCopy**	
rdfs:comment	An IP Entity that is a copy of an AdaptationInstance	
rdfs:subClassOf	mvco:Copy	
Restrictions	**Expression**	**Meaning**
	⊆ ∃ resultedFrom MakeAdaptationInstanceCopy	An AdaptationInstanceCopy is the result of executing a MakeAdaptationInstanceCopy Action.

class	**AdaptationManifestationCopy**	
rdfs:comment	An IP Entity that is copy of an AdaptationManifestation	
rdfs:subClassOf	mvco:Copy	
Restrictions	**Expression**	**Meaning**
	≡ (∃ resultedFrom MakeAdaptationManifestationCopy ∩ Copy)	An AdaptationInstanceCopy is every copy resulting of executing a MakeAdaptationManifestationCopy.

class	**WorkInstanceCopy**	
rdfs:comment	A copy of WorkInstance	
rdfs:subClassOf	mvco:Copy	
Restrictions	**Expression**	**Meaning**
	≡ (∃ resultedFrom MakeWorkInstanceCopy ∩ Copy)	A WorkInstanceCopy is every copy resulting of executing a MakeWorkInstanceCopy.

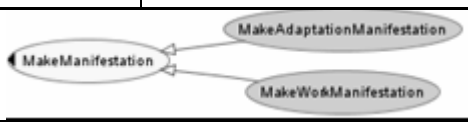
class	**WorkManifestationCopy**	
rdfs:comment	An IP Entity copy of a WorkInstance	
rdfs:subClassOf	mvco:Copy	
Restrictions	**Expression**	**Meaning**
	≡ (∃ resultedFrom MakeWorkManifestationCopy ∩ Copy)	A WorkManifestationCopy is every copy resulting of executing a MakeWorkManifestationCopy.

7.3.3 Actions and Rights

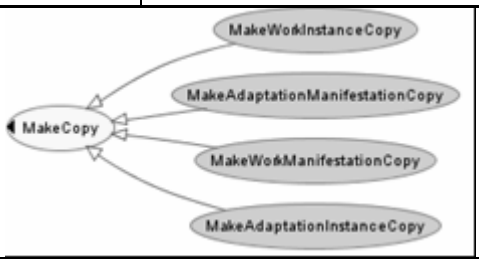
class	**Action**	
rdfs:comment	Process of performing functions	
rdfs:subClassOf	owl:Thing	
Restrictions	**Expression**	**Meaning**
	⊆ ∃ actedBy User ⊆ actedBy exactly 1	Actions are acted by one and only one User.
	⊆ ∃ impliesAlso Action	Actions may imply other Actions
	⊆ ∃ rightsGivenBy User	Only Users can give rights to execute Actions
in-domain-of	actedOver (IPEntity) actedBy (User) rightGivenBy (User) resultsIn (IPEntity) impliesAlso (Action)	
Known subclasses		

class	**CreateWork**	
rdfs:comment	The Action of creating a Work without any previous IP Entity	
rdfs:subClassOf	mvco:Action	
Restrictions	**Expression**	**Meaning**
	⊆ (¬ (∃actedOver IPEntity))	Cannot be executed over any IPEntity
	⊆ (∃resultsIn Work)	The result of creating a work is at least one Work (and at most one, because *resultsIn* is a functional relation)
	⊆ rightGivenBy exactly 0	No one can give the right to create a Work

Class	**MakeAdaptation**	
rdfs:comment	The Action of making an Adaptation	
rdfs:subClassOf	mvco:Action	
Restrictions	**Expression**	**Meaning**
	⊆ (∃ actedOver Work)	An Adaptation is made over some Work
	⊆ (∃ rightGivenBy Creator)	Only Creators give the right to make an Adaptation
	⊆ (∃resultsIn Adaptation)	The act of making an adaptation results in one Adaptation.

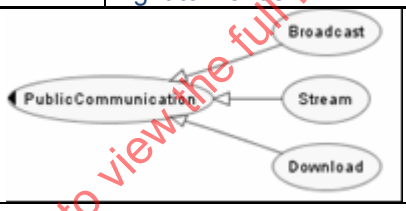
Class	**MakeManifestation**	
rdfs:comment	The Action of making a Manifestation.	
rdfs:subClassOf	mvco:Action	
Restrictions	**Expression**	**Meaning**
	⊆ (∃actedOver (Work U Adaptation))	A Manifestation has to be done over some Work or Adaptation
	⊆ (∃resultsIn . Manifestation)	The act of making a manifestation results in one Manifestation.
Known subclasses		

class	**MakeInstance**	
rdfs:comment	The Action of making an Instance from a Manifestation.	
rdfs:subClassOf	mvco:Action	
Restrictions	**Expression**	**Meaning**
	⊆ (∃actedOver . Manifestation)	Instances are created from Manifestations
	⊆ (∃resultsIn . Instance)	The act of making an Instance results in one Instance.
Known subclasses		

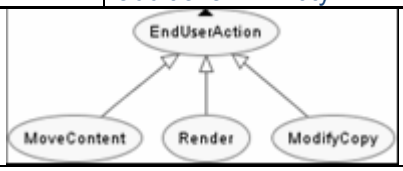
class	**MakeCopy**	
rdfs:comment	The Action of making a Copy	
rdfs:subClassOf	mvco:Action	
Restrictions	**Expression**	**Meaning**
	⊆ (∃actedOver . Instance)	Copies are created from Instances
	⊆ (∃resultsIn . Copy)	The act of making a Copy results in Copy.
Known subclasses		

class	**Produce**	
rdfs:comment	The Action of making Products	
rdfs:subClassOf	mvco:Action	
Restrictions	**Expression**	**Meaning**
	⊆ (∃ actedOver (Copy U Instance U Manifestation))	Products are created from Copies or Instances
	⊆ (∃resultsIn . Product) ⊆ (resultsIn exactly 1)	The act of making a Product results in exactly 1 Product.
	⊆ (∃rightsGivenBy Creator)	Producing requires the additional permission of the Creator.

class	**Synchronise**	
rdfs:comment	The Action of concurrently perform/display two distinct IP Entities each for a different human sense e.g. text and audio or video and song	
rdfs:subClassOf	mvco:Action	
Restrictions	**Expression**	**Meaning**
	⊆ (∃actedOver . Work)	Synchronisation is made over Works
	⊆ (∃rightGivenBy . Creator)	Making a Synchronisation requires the additional permission given by the Creator.
	⊆ (∀impliesAlso . (Render U ModifyCopy))	Any User who has the right to Synchronise, can also invoke the right to Render or Modify a Copy.

class	**PublicCommunication**	
rdfs:comment	The Action of publicly displaying/performing, e.g. live performance, radio, television, internet streaming, multicast of Instances and Manifestations, and download	
rdfs:subClassOf	mvco:Action	
Restrictions	**Expression**	**Meaning**
	⊆ (∃actedOver . Copy U Product)	Public Communication can be performed on Copies or Products
	⊆ (∃rightGivenBy .Creator)	Making PublicCommunication requires the additional permission given by the Creator.
	⊆ (∀impliesAlso . Render)	Any User who has the right to make a PublicCommunication, can indeed invoke the right to Render
Known subclasses		

Class	**Distribute**	
rdfs:comment	The Action of selling, renting and lending	
rdfs:subClassOf	mvco:Action	
Restrictions	**Expression**	**Meaning**
	⊆ (∃actedOver Product)	There must be a Product to distribute
	⊆ (resultsIn exactly 0)	The act of Distribute cannot generate any additional IPEntity
	⊆ (∃rightsGivenBy Creator Producer)	Distributing requires the additional permission of the Creator.

class	**EndUserAction**
rdfs:comment	Action performed by an EndUser
rdfs:subClassOf	mvco:Action
Restrictions	**Expression** / **Meaning** ⊆ (∃actedOver . Product) — End Users only deal with Products ⊆ (resultsIn exactly 0) — EndUserActions cannot generate any additional IPEntity
Known subclasses	

class	**Broadcast**
rdfs:comment	The Action that Delivers Content to a Device in a point-to-multipoint modality
rdfs:subClassOf	mvco:PublicCommunication
Restrictions	none

class	**Download**
rdfs:comment	The Action of transferring a file or program from a central computer to a smaller computer or to a computer at a remote location
rdfs:subClassOf	mvco:PublicCommunication
Restrictions	none

class	**Stream**
rdfs:comment	The function of delivering Content to a Device where the transferred Content is processed for rendering only and not stored
rdfs:subClassOf	mvco:PublicCommunication
Restrictions	none

class	**MakeAdaptationInstanceCopy**
rdfs:comment	The Action of making an AdaptationInstanceCopy
rdfs:subClassOf	mvco:MakeCopy
Restrictions	**Expression** / **Meaning** ≡ (∃ actedOver AdaptationInstance ∩ MakeCopy) — MakeCopy executed over an AdaptationInstance. ⊆ (∃ resultsIn AdaptationInstanceCopy) — The result must be an AdaptationInstanceCopy

class	**MakeAdaptationManifestationCopy**
rdfs:comment	The Action of making an AdaptationManifestationCopy
rdfs:subClassOf	mvco:MakeCopy
Restrictions	**Expression** / **Meaning** ≡ (∃ actedOver AdaptationManifestation ∩ MakeCopy) — MakeCopy executed over an AdaptationManifestation. ⊆ (∃ resultsIn AdaptationManifestationCopy) — The result must be an AdaptationManifestationCopy

class	**MakeWorkInstanceCopy**	
rdfs:comment	The Action of making an WorkInstanceCopy	
rdfs:subClassOf	mvco:MakeCopy	
Restrictions	**Expression**	**Meaning**
	≡ (∃ actedOver WorkInstance ∩ MakeCopy)	MakeCopy executed over a WorkInstance.
	⊆ (∃ resultsIn WorkInstanceCopy)	The result must be an WorkInstanceCopy

class	**MakeWorkManifestationCopy**	
rdfs:comment	The Action of making a WorkManifestationCopy	
rdfs:subClassOf	mvco:MakeCopy	
Restrictions	**Expression**	**Meaning**
	≡ (∃ actedOver WorkManifestation ∩ MakeCopy)	MakeCopy executed over a WorkManifestation.
	⊆ (∃ resultsIn WorkManifestationCopy)	The result must be an WorkManifestationCopy

class	**MakeAdaptationInstance**	
rdfs:comment	The Action of making an Instance from an AdaptationManifestation	
rdfs:subClassOf	mvco:MakeInstance	
Restrictions	**Expression**	**Meaning**
	≡ (∃ actedOver AdaptationManifestation ∩ MakeInstance)	IMakeInstance executed over one AdaptationManifestation.
	⊆ (∃ resultsIn AdaptationInstance)	The result must be an AdaptationInstance

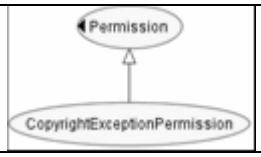
class	**MakeWorkInstance**	
rdfs:comment	The Action of making an Instance from a WorkManifestation	
rdfs:subClassOf	mvco:MakeInstance	
Restrictions	**Expression**	**Meaning**
	≡ (∃ actedOver WorkManifestation ∩ MakeInstance)	It is a MakeInstance executed over one WorkManifestation.
	⊆ (∃ resultsIn WorkInstance)	The result must be a WorkInstance

class	**MakeWorkManifestation**	
rdfs:comment	The Action of making a Manifestation of a Work.	
rdfs:subClassOf	mvco:MakeManifestation	
Restrictions	**Expression**	**Meaning**
	≡ (∃ actedOver Work ∩ MakeManifestation)	MakeManifestation executed over a Work.
	⊆ (∃ resultsIn WorkManifestation)	The result must be a WorkManifestation

7.3.4 Permission and related classes

class	**Fact**	
rdfs:comment	Positive proposition.	
rdfs:subClassOf	owl:Thing	
Restrictions	**Expression**	**Meaning**
	≡ (truth exactly 1)	Any individual belongs to the mvco:Fact class as long as its truth or falsehood can be asserted.
Known subclasses		

class	**CopyrightExceptionFact**
rdfs:comment	Fact related to the invocation of a CopyrightException.
rdfs:subClassOf	mvco:Fact

class	**Permission**	
rdfs:comment	Authorisation from one Rights owner to one or more Users to perform one or more Actions on a given IPEntity. (Note: This includes the defacto application of CopyrightExceptions via existence of corresponding Facts.)	
rdfs:subClassOf	owl:Thing	
Restrictions	**Expression**	**Meaning**
	⊆ ∃permitsAction Action ⊆ ∀permitsAction Action	Permissions permit at least one Action and only Actions.
	⊆ ∃ issuedBy User ⊆ issuedBy exactly 1	Permissions are issued by exactly one User
	⊆ ∃ hasRequired Fact	Permissions have as requirements only Facts
in-domain-of	hasRequired (Fact) issuedBy (User) permitsAction (Action)	
Known subclasses		

class	**CopyrightExceptionPermission**	
rdfs:comment	Permission to invoke one right exceptionally.	
rdfs:subClassOf	mvco:Permission	
Restrictions	**Expression**	**Meaning**
	⊆ (∃hasRequired . CopyrightExceptionFact)	Permissions must have at least one CopyrightExceptionFact as requirement

7.4 Reference of properties

Two kinds of properties can be defined in an OWL Ontology:

- Object Properties, which link two class individuals.
- Datatype Properties, which attributes a literal value to a class individual.

A *property axiom* defines characteristics of a property. In its simplest form, a property axiom just defines the existence of a property but they can also define additional characteristics of properties. OWL supports the following constructs for property axioms:

- *RDF Schema constructs:* `rdfs:subPropertyOf`, `rdfs:domain` *and* `rdfs:range`. In MVCO there is no hierarchy of properties, but *domains* and *ranges* are given for each property. By giving a *domain* we assert that the property only applies to instances of the given class (i.e. `mvco:Action` is the domain of the property `mvco:resultsIn` because it can only be applied on `mvco:Action` individuals). By giving a *range* we assert that the property only assumes values that are instances of the given class (e.g. executing a `mvco:Action`, results in an `mvco:IPEntity`).

- *Relations to other properties:* `owl:equivalentProperty` *and* `owl:inverseOf`. MVCO defines 4 pairs of property - inverse property.

- *Global cardinality constraints:* `owl:FunctionalProperty`, `owl:InverseFunctionalProperty`. MVCO defines some properties as functionals, (properties that can have only one (unique) value for each instance).

- *Logical property characteristics: owl:SymmetricProperty and owl:TransitiveProperty*. Two properties are defined to be transitive in the MVCO ontology, and none to be symmetric.

Table 3 lists the object properties in the MVCO ontology, including their features (T stands for transitive, F for functional).

Table 3 — Object Properties in the MVCO Ontology.

Relation	Definition	Domain	Range	P	Inverse
`actedOver`	Specifies which IP Entity is the object of the Action	Action	IPEntity		
`resultsIn`	Declares which IP Entity arises as a result of the execution of an Action. It is a functional relation.	Action	IPEntity	F	*resultedFrom*
`actedBy`	Declares the User who has executed the Action	Action	User	F	
`acts`	Performance of an Action by a User	User	Action		
`hasRequired`	For a Permission to be valid, the Fact has to hold	Permission	Fact		
`impliesAlso`	Explicit link between one and any other number of Actions	Action	Action		
`rightGivenBy`	Declares from which User rights are given	Action	User		
`permitsAction`	Relation used to express the Actions that are allowed to be performed.	Permission	Action		
`actOnBehalfOf`	Relates a ContentHandler with the User under the auspices of which the ContentHandler operates.	Content Handler	User	T	
`issuedBy`	Declares who has issued a Permission	Permission	User		
`hasRightsOwner`	Defines the owner of the Rights over an IP Entity.	IPEntity	User	F	*isRightsOwnerOf*
`isMadeUpOf`	Relates a composite IP Entity with its constituent IP Entities	IPEntity	IPEntity		
`belongsTo`	Relates a User with a Collective	User	Collective	T	

Some datatype properties also have been defined in the MVCO ontology. They are listed in Table 4, along with its definition, its domain, and its type (the kind of value it can assume, which is one of the XML Schema types).

Table 4 — Datatype properties in the MVCO Ontology

Property	Domain	type	Definition
`dii:RelatedIdentifier`	owl:Thing	xsd:string	It allows the identification information that is related to a Digital Item (or parts thereof).
`isDigital`	IPEntity	xsd:boolean	Specifies whether the IP Entity is digital or not
`hasSocialTag`	IPEntity U User	xsd:String	Specifies an entry point for social tags to be added
`isTrue`	Fact	xsd:boolean	Truth

Annex A
(informative)

Media Value Chain Ontology Use

A.1 MVCO API

A.1.1 Introduction

The Ontology is presented in a publicly open OWL file. OWL language is based on RDFS, ultimately an XML file, making it technologically neutral and open for any application to use it. However, in order to increase and to facilitate its use, an API has been proposed. Regardless of the existence of automatic code generation tools, this API provides the most common operations that a user of the ontology may request, and makes swifter the development of MVCO-based applications.

The library can be programmed in Java –the standard language for Semantic Web programs and may be part of the Reference Software.

The API is structured in two levels:

- **Level 1 methods** define an access to the Intellectual Property Value Chain with the minimum and necessary methods. These methods are defined using simple types, like String and Vectors, and their use should be straightforward for fast development of simple applications or prototyping more complex applications. An inexperienced programmer with little knowledge on the ontology could yet make useful applications based on it.

- **Level 2 methods** declare a richer set of classes and methods, allowing a complete management of the ontology features. However, their use requires some apriori knowledge of the ontology structure and features.

A.1.2 API Level 1

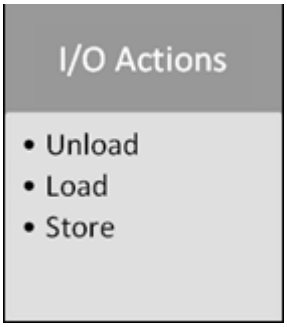
API Level 1 methods belong to a single class, whose interface is detailed in the next Clause. Furthermore, Level 1 API methods allow non object-oriented language to be deployed and a C version of this API can be promptly deduced and programmed. Functions make use of basic types.

The MVCO API class provides the basic functionality to:

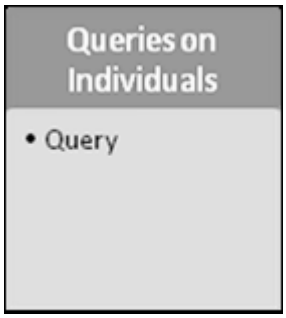
- **To query the ontology for key model properties**. These calls query on the model itself. They allow retrieving which kind of roles have been declared, or which kind of Actions are available for a given IP Entity etc., all of which is fixed in the Core Model but may change in additional extensions of the ontology.

- **To manage a set of class individuals**. Thus, real cases can be dealt, being possible to register Users, make them execute Actions which are validated against particular Permissions, etc.
    - Some of this Actions can only be done by the Super User (or controller of the Ontology) such as registering or deleting users etc.
    - Some of these Actions can be done by the Interested User: execute Actions and issuer Permissions.

- **To store and retrieve an image with the set of declared individuals**. Persistence of class individuals is a key feature to implement practical applicaltions. Storage and retrieval of individuals does not necessarily imply loading them at once in memory, and partial loads of the memory are allowed upon execution of queries.



- **Queries on the ontology individuals.** Queries can be made by any of the users, or even by non-registered users. Information in the response may be not complete depending on the user credentials (some information stored in the ontology may be public but some not). The most basic method accepts SPARQL queries.



A.1.2.1 API Level 2

API Level 2 functions provide a more complete access to the ontology features. An interface class has been derived from MVCO API, and a set of auxiliary classes (representing User, IP Entity, Permission and Action) has been defined too. Programming with API Level 2 classes requires necessarily an Object Oriented approach, and invoking the methods imply handling with classes.

The functionalities not covered by API Level 1 include:

- *Management of Collective Users*: Collective Users can be created and erased, and Users can be subscribed to Collective.
- *Management of Composed IP Entities*: IP Entities can be composited, as a Movie where Music and Screenplay are Works by themselves.
- *Management of ContentHandlers*: Users can be appointed to do a particular Action in representation (on behalf of) an authorised User. API Level 2 allows a User to be a ContentHandler.
- *Requirements on Permissions:* More complex permissions can be issued. Thus, Requirements can be added to the Permissions, actually being no more than boolean Facts but left to the programmer's skill for their representation. A Factory Method (the well known programming pattern) may generate the Facts (conditions) needed to express the requirements. In a C++ environment, *functionals* (pointer to functions) may be used to implement these abstract Facts.
- *Prohibitions on Permissions:* Prohibitions can be stated as the requirement of a Fact not to hold, all of which can lead to a boolean logic that the API Level 2 deals. Thus, Facts can be combined in logical expressions (using the logical operators NOT, AND, OR) to conform Prohibitions and other composed requirements.
- *Copyright Exceptions can be invoked by Users:* Thus, apart from the ExecuteAction in the API Level 1 methods, a new InvokeCopyrightException method is offered. Upon execution, the existence of `mvco:CopyrightExceptionPermission` is searched, and examined against the corresponding `mvco:CopyrightExceptionFacts`.
- *Rights implications:* Having permission to certain Actions, allow the User to execute others additionally. Thus, a User allowed to make PublicCommunication, is allowed to make a Render. This, feature, that is expressed in the ontology with the `mvco:impliesAlso` Object Property, can be checked with an API Level 2 method.
- Implementation of the API may be given as reference software. It can be made with the help of some well-known Semantic Web libraries.

A.1.3 A Java MVCO Interface

This Clause specifies a Java Interface (called MVCOInterface) which implements the Level 1 methods described in A.1.2.

A.1.3.1 Operation to load a data set from a given location.

Interface	MVCOInterface
Method Syntax	
+ Load(String sURI): Boolean	
Java sample code	
`public boolean Load(String sURI) throws IOException;`	
Method description	
This method is employed to load a set of assertions from a URI. This method is additive, i.e., loading a second set of data does not replace the existing one. Data loaded includes Users, IP Entities (Digital Items), Permissions and Actions performed.	
Parameters	
sURI	Any URI making reference to an existing data repository. It can be a file, for example: file:///test.owl, but also database or a remote location (http://abc.com/test.owl).
Return value	
True on success	
Exceptions	
IOException	If the URI is not reachable.

A.1.3.2 Operation to save the current data set in a given location.

Interface	MVCOInterface
Method Syntax	
+ Store(String sURI): Boolean	
Java sample code	
`public boolean Store(String sURI) throws IOException;`	
Method description	
This method is employed to store a set of assertions from a URI	
Parameters	
sURI	Any URI making reference to an existing data repository. It can be a file, for example: file:///test.owl, but also database or a remote location (http://abc.com/test.owl).
Return value	
True on success	
Exceptions	
IOException	If the URI is not reachable.

A.1.3.3 Operation to unload the current data set.

Interface	MVCOInterface
Method Syntax	
+ Unload(): Boolean	
Java sample code	
`public boolean Unload();`	
Method description	
This method is employed to unload the assertions in the ontology. No individuals will remain after calling this operation.	
Parameters	
Return value	
True on success	

A.1.3.4 Operation to create a user in the ontology.

Interface	MVCOInterface
Method Syntax	
+ CreateUser(String sUserId): Boolean	
Java sample code	
`public boolean CreateUser(String sUserId) throws ExistingIdException, BadIdException;`	
Method description	
This method is employed to create a new User, identified by a user id. The new User does not assume any role *a priori*.	
Parameters	
sUserId	Any User identification string, provided the following: • It can only start with alphabetic characters: [a-z][A-Z] • It can be made of alphanumeric characters [a-z][A-Z][0-9], explicitly excluding blank spaces, commas or any other non-alphanumeric character
Return value	
True on success	
Exceptions	
ExistingIdException	If the identifier already exists.
BadIdException	The id string is malformed

A.1.3.5 Operation to delete a user in the ontology.

Interface	MVCOInterface
Method Syntax	
+ DeleteUser(String sUserId): Boolean	
Java sample code	
`public boolean DeleteUser(String sUserId) throws ExistingIdException, BadIdException;`	
Method description	
This method is employed to unregister an existing User, identified by a user id.	
Parameters	
sUserId	Any User identification string, provided the following: • It can only start with alphabetic characters: [a-z][A-Z] • It can be made of alphanumeric characters [a-z][A-Z][0-9], explicitly excluding blank spaces, commas or any other non-alphanumeric character
Return value	
True on success	
Exceptions	
ExistingIdException	If the user identifier does not exist in the ontology.
BadIdException	The id string is malformed

A.1.3.6 Operation for the execution of an action by a User.

Interface	MVCOInterface
Method Syntax	
+ ExecuteAction(String sActedBy, String sAction, String sIPEntity, String sNewIPEntity): Boolean	
Java sample code	
`public boolean ExecuteUser(String sActedBy, String sAction, String sIPEntity, String sNewIPEntity) throws BadIdException, IllegalActionException, ExistingIdException;`	
Method description	
This method is employed for an User to execute an Action over an IPEntity, possible creating a new IPEntity.	
Parameters	
sActedBy	User id acting the action.
sAction	Action to be performed (e.g. “Render”, “CreateWork”)
sIPEntity	Existing IPEntity id over which the Action is executed
sNewIPEntity	New id to for the IPEntity to be created –if needed. This parameter may be an empty string, or a null.
Return value	
True on success	
Exceptions	
IllegalActionException	An operation cannot be performed over a given IPEntity.
ExistingIdException	The given id points to a non-existing User, Action or IPEntity.
BadIdException	An id string is malformed, either in the User, the Action or the IPEntity

A.1.3.7 Operation to create a permission.

Interface	MVCOInterface
Method Syntax	
+ CreatePermission(String sIssuedBy, String sAction, String sActedOver, String sActedBy): boolean	
Java sample code	
`public boolean CreatePermission(String sIssuedBy, String sAction, String sActedOver, String sActedBy) throws BadIdException, IllegalActionException, ExistingIdException;`	
Method description	
This method is employed for an User to generate the permission to execute an Action over an IPEntity, by other User.	
Parameters	
sIssuedBy	User id issuing the permission.
sAction	Action to be performed (e.g. “Render”, “CreateWork”)
sActedOver	Existing IPEntity id over which the Action is authorised to be executed
sActedBy	User id of the authorised User.
Return value	
True on success	
Exceptions	
IllegalActionException	An operation cannot be performed over the given IPEntity.
ExistingIdException	The given id points to a non-existing User, Action or IPEntity.
BadIdException	An id string is malformed, either in the User, the Action or the IPEntity

A.1.3.8 Operation to launch a query.

Interface	MVCOInterface
Method Syntax	
+ Query(String sSPARQL): Vector<String>	
Java sample code	
`public Vector<String> Query(String sSPARQL) throws MalformedQueryException;`	
Method description	
This method is employed to retrieve information on the ontology. The operation always retrieves information. Note that along 2009 “SPARQL Update” is likely to be declared W3C recommendation. If this happens, queries could change the ontology, radically altering the scope of this function.	
Parameters	
sSPARQL	String with the SPARQL query. This string has to be formed according to the W3C recommendation[3].
Return value	
A vector with the strings returned as a response to the query, if this was successful, or empty vector i fit was not.	
Exceptions	
MalformedQueryException	The query execution returned an error.

[3] http://www.w3.org/TR/rdf-sparql-query/

A.1.3.9 Operation to validate an action.

Interface	MVCOInterface
Method Syntax	
+ Validate(String sAction, String sActedOver, String sActedBy): boolean	
Java sample code	
`public boolean Validate(String sAction, String sActedOver, String sActedBy) throws BadIdException, IllegalActionException, ExistingIdException;`	
Method description	
This method is employed to validate an action, giving a boolean answer (yes/no).	
Parameters	
sAction	Action to be authorised (e.g. “Render”, “CreateWork”)
sActedOver	Existing IPEntity id over which the Action is being validated.
sActedBy	User id of the User under validation.
Return value	
True if the operation is validated.	
Exceptions	
IllegalActionException	An operation cannot be performed over the given IPEntity.
ExistingIdException	The given id points to a non-existing User, Action or IPEntity.
BadIdException	An id string is malformed, either in the User, the Action or the IPEntity

A.1.3.10 Operation to get the list of registered users.

Interface	MVCOInterface
Method Syntax	
+ getUsers(): Vector<String>	
Java sample code	
`public Vector<String> getUsers();`	
Method description	
This method is employed to retrieve the list of existing Users	
Return value	
A vector with the String of the User id´s registered in the ontology.	
Exceptions	

A.1.3.11 Operation to get the list of registered IPEntities.

Interface	MVCOInterface
Method Syntax	
+ getIPEntities(): Vector<String>	
Java sample code	
`public Vector<String> getIPEntities();`	
Method description	
This method is employed to retrieve the list of existing IPEntities	
Return value	
A vector with the String of the IPEntities id´s registered in the ontology.	
Exceptions	

A.1.3.12 Operation to get the rights owner of a given IPEntity.

Interface	MVCOInterface
Method Syntax	
+ getRightsOwner(String sIPEntity): String	
Java sample code	
`public boolean getRightsOwner(String sAction, String sActedOver, String sActedBy) throws BadIdException, ExistingIdException;`	
Method description	
This method retrieves the rights Owner of a given IPEntity. It does not retrieve which users have been authorised in valid Parmissions. This method is redundant and given for convenience, as the query could be performed by means of the Query method.	
Parameters	
sIPEntity	IPEntity to be queried
Return value	
User id owner of the IPEntity rights.	
Exceptions	
ExistingIdException	The given id points to a non-existing User, Action or IPEntity.
BadIdException	An id string is malformed, either in the User, the Action or the IPEntity

A.1.3.13 Operation to geg the IPEntity origin of a given IPEntity.

Interface	MVCOInterface
Method Syntax	
+ getOrigin(String sIPEntity): String	
Java sample code	
`public String getOrigin(String sIPEntity) throws BadIdException, ExistingIdException;`	
Method description	
This method returns the IPEntity that has given rise to the current IPEntity. If the IPEntity is a Work, a null will be returned. This method is redundant and given for convenience, as the query could be performed by means of the Query method.	
Parameters	
sIPEntity	IPEntity to be queried
Return value	
IPEntity id from which the given IPEntity id comes from.	
Exceptions	
ExistingIdException	The given id points to a non-existing User, Action or IPEntity.
BadIdException	An id string is malformed, either in the User, the Action or the IPEntity

A.2 Extension of the MVCO with MPEG-21 REL rights

ISO/IEC 21000-5 defines 14 rights in its Multimedia Extension (whose namespace prefix is usually denoted mx)

1. mx:adapt – subclassOf of mvco:ModifyCopy
2. mx:delete - subclassOf of mvco:MoveContent
3. mx:diminish - subclassOf of mvco:ModifyCopy
4. mx:embed - subclassOf of mvco:Synchronize
5. mx:enhance - subclassOf of mvco:ModifyCopy
6. mx:enlarge - subclassOf of mvco:ModifyCopy
7. mx:execute - subclassOf of mvco:Render

8. mx:install - subclassOf of mvco:MoveContent

9. mx:modify - subclassOf of mvco:ModifyCopy

10. mx:move - subclassOf of mvco:MoveContent

11. mx:play - subclassOf of mvco:Render

12. mx:print - subclassOf of mvco:Render

13. mx:reduce - subclassOf of mvco:ModifyCopy

14. mx:uninstall - subclassOf of mvco:MoveContent

A.3 Sample Scenario of Use

This Subclause describes a scenario where the MVCO can play a key role certifying the adherence of some DRM operations in a Content Management System to the MVCO. This scenario is not intended to represent the sole use of MVCO but is given to facilitate understanding of how to practically exploit the MVCO.

Consider a Content Management System with the following components:

- A Content Repository, possibly storing MPEG-21 Digital Items.
- A Licensing Server, possibly storing and authorising MPEG-21 REL licenses.
- A User database, including content providers, instantiators, broadcasters, end users etc.
- A MVCO Server

Consider Bob an identified User Creator of an original Work with a digital Manifestation encapsulated in a DI.

1. The Work is registered in the MVCO Server, being Bob the sole Rights owner.
2. The Manifestation is registered in the MVCO Server, being linked with the Work it is of.
3. The DI embodying the Manifestation is registered and uploaded in the Content Repository.
4. A REL license template is uploaded into the Licensing Server.

Now consider a second User of the system, called Alice, who is an Instantiator and wants to make an instance of Bob´s work.

1. Alice obtains the DI representing Bob´s Work Manifestation with a REL license only for creating an Instance (an appropriate Right can be defined).
2. Alice uploads the Instance embodied in a Digital Item.

Finally we consider a third User of the system, BroadcasterUnion, a broadcaster who is interested in making a Public Communication of Alice´s performance of Bob´s Work. Note that in order to make a Public Communication, both the consent of the performer and that of the author are required.

1. BroadcasterUnion obtains the DI, by buying a REL license.
2. BroadcasterUnion obtains a Permission from the MVCO Server, issued by Alice as the Instantiator.
3. BroadcasterUnion obtains a Permission from the MVCO Server, issued by Bob as the Creator.

The MVCO Server can be used for a number of purposes:

- Certify that licenses and license templates adhere to the Value Chain Model. In particular:
  - Certify that Bob´s REL license template is valid, as it is issued for a Manifestation whose Work belongs to him.
  - Certify that Alice´s Instances belongs to her.
- Validate operations whose authorisation mechanism is out of MPEG-21 REL scope. In particular:
  - Validate that BroadcasterUnion's broadcasting counts with the permissions of both interpreter and author and thus is conformant to the IP Value Chain.
- Respond to queries in some of the richest expression formats, like SPARQL, taking advantage of MVCO Server reasoning capabilities. Thanks to MVCO inferencing potential, intelligent responses can be given. For example:
  - Answer queries about which kind of IP Entity a Digital Item represents.
  - Answer queries about which kind of Role a User plays with respect to a certain IP Entity.
  - Answer queries about relationships between IP Entities like provenance, existing permissions and derivative works.
- Offer the best interfaces for interoperability and ontology extension. For example:
  - Offer an interface of the system to other platforms in the easiest possible fashion. Semantic Web expressions are thought and intended for interoperation with other Semantic Web models, thanks to its semantic expression capabilibies. Mappings are expressed as RDF / OWL and seamlessly integrated in the system.
  - Grant compatibility between custom MVCO extensions refining existing classes or properties.

Note that these functions cannot be performed by virture of any of the existing parts of MPEG-21 or combinations thereof.

A.4 Class individuals in several examples of Use)

For several cases, this section describes which class individuals are present at a given moment in the ontology.

Bob creates a Work

OWL individuals in the MVCO represent real Users, IP Entities, Actions and Permissions. When Bob creates a Work, the ontology represents it with a minimum of two MVCO class individuals: Bob and the created Work. But the Action itself can also be represented and stored as an individual. This Action may trigger an Event Report (ISO/IEC 21000-15).

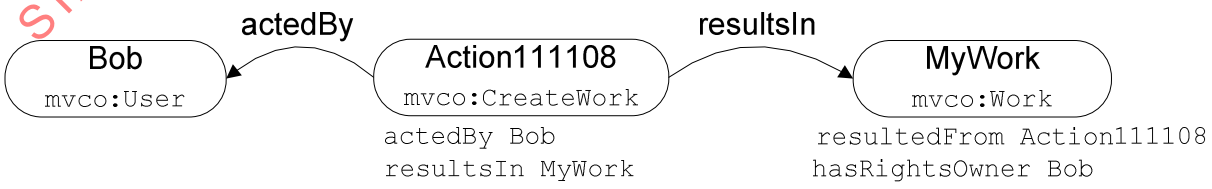


Figure A.1 — Three individuals in the Ontology

Figure A.1 — Three individuals in the Ontology represents the three individuals, “Bob” ,“Action111108” and “MyWork” and the class they belong (User, CreateWork, Work). The arrows represents the relations (object properties) by which they are linked, and below the values taken by the object properties are given. Note that not all of the possible object properties are represented here.

Alice makes an Instance

Figure A.2 — Individuals when Alice makes an Instance

Once Bob has created a Work and a Manifestation (the latter not represented here), Alice´s Action of creating an Instance is represented in Figure A.2 — Individuals when Alice makes an Instance. Note that in MVCO some properties have exactly one value. Thus, the object property “actedBy” has to receive a value for each existing Action, while “actedOver” may or may not.

Also note that the figure represents the “Action111110” as an individual of MakeInstance.

Charles and Claire make a PublicCommunication

Finally, Charles and Claire, acting as a Collective (called BroadcasterUnion) make a PublicCommunication of Bob´s Work, instantiated by Alice (Figure A.3 — Alice's Instance is broadcasted).

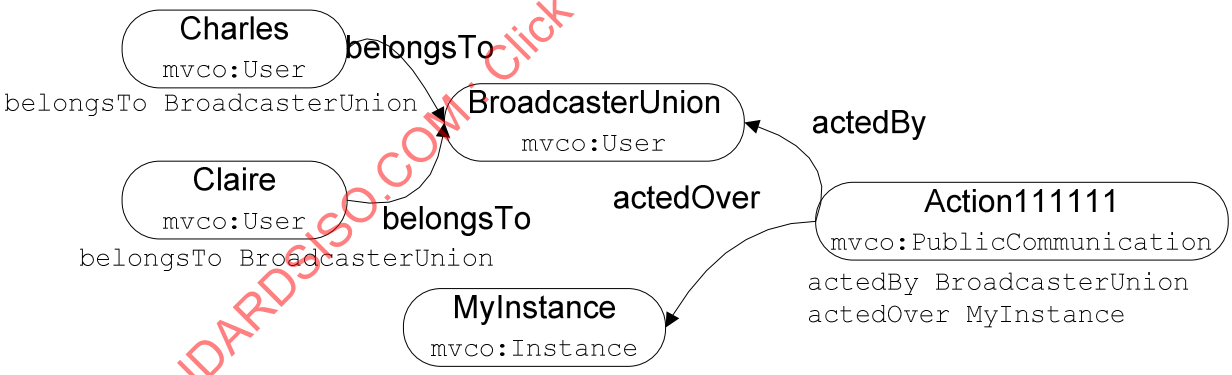


Figure A.3 — Alice's Instance is broadcasted

Charles and Clair have Permission to make the PublicCommunication

For the Public Communication of Bob´s Work to adhere to the Value Chain Model, there must be two Permissions; not only the one from Alice as Interpreter but also from Bob as the Creator.

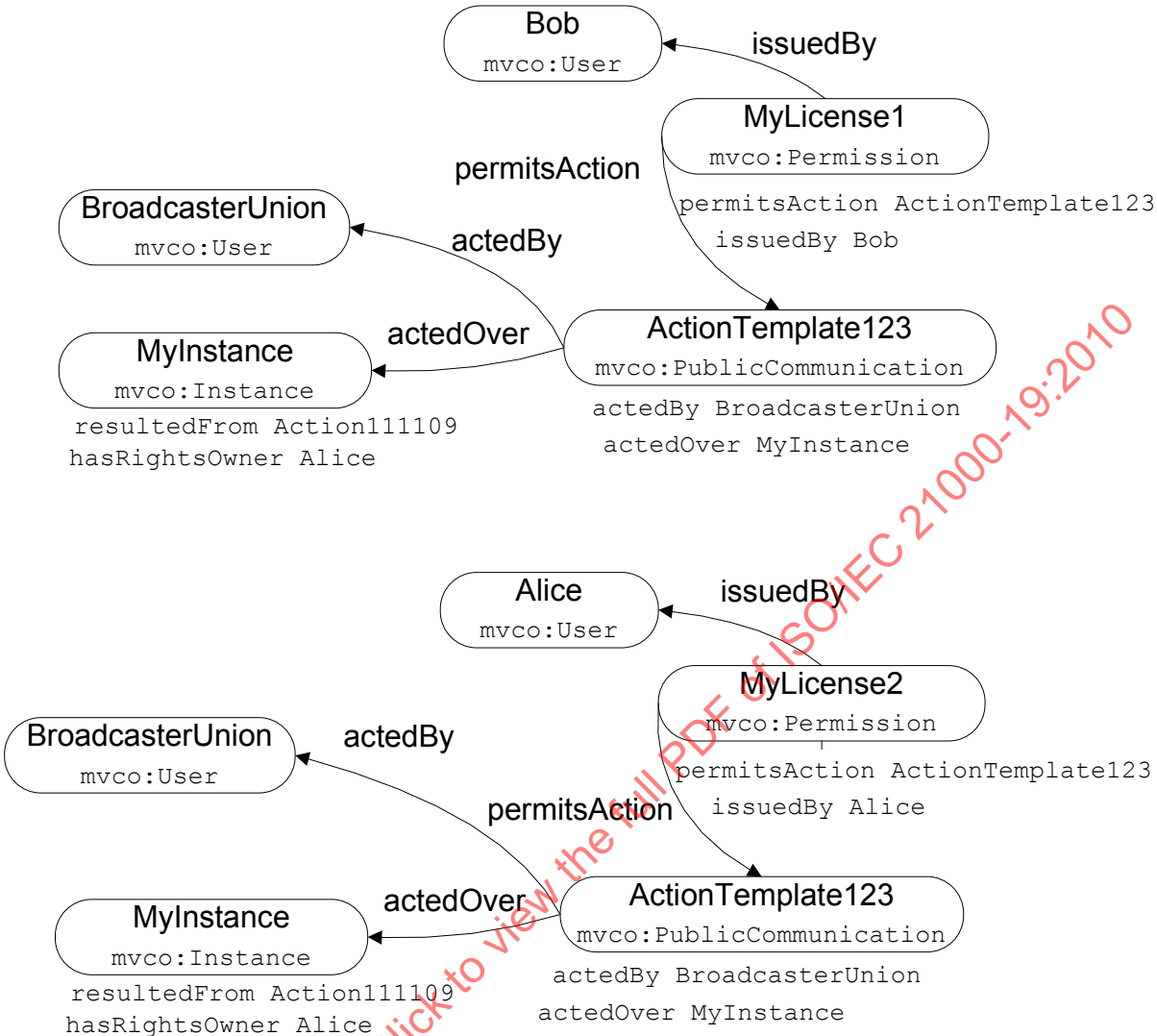


Figure A.4 — Two permissions are needed to make a Public Communication

Annex B
(normative)

MVCO OWL

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:mvco="http://purl.oclc.org/NET/mvco.owl#"
    xmlns:mx="urn:mpeg:mpeg21:2003:01-REL-MX-NS#"
    xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
  xml:base="http://purl.oclc.org/NET/mvco.owl">
  <owl:Ontology rdf:about="">
    <rdfs:comment xml:lang="en">ISO/IEC 21000-19 Media Value Chain Ontology
(MVCO)</rdfs:comment>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
  </owl:Ontology>
  <owl:Class rdf:ID="Instantiator">
    <rdfs:comment xml:lang="en">A User who interprets a Manifestation of a Work
making an Instance</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="User"/>
    </rdfs:subClassOf>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <owl:equivalentClass>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="acts"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="MakeInstance"/>
        </owl:someValuesFrom>
      </owl:Restriction>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:ID="MakeWorkInstance">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#MakeInstance"/>
          <owl:Restriction>
            <owl:someValuesFrom>
              <owl:Class rdf:ID="WorkManifestation"/>
            </owl:someValuesFrom>
            <owl:onProperty>
              <owl:ObjectProperty rdf:ID="actedOver"/>
            </owl:onProperty>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
```

```
    </owl:equivalentClass>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:ID="resultsIn"/>
        </owl:onProperty>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="WorkInstance"/>
        </owl:allValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">The Action of making an Instance from a Work
Manifestation.</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="#WorkInstance">
    <rdfs:comment xml:lang="en">An object or event which is an example of an
Identified Manifestation of a Work (e.g. a File)</rdfs:comment>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:ID="Instance"/>
          <owl:Restriction>
            <owl:someValuesFrom rdf:resource="#MakeWorkInstance"/>
            <owl:onProperty>
              <owl:ObjectProperty rdf:ID="resultedFrom"/>
            </owl:onProperty>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:ID="Collective">
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#User"/>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">Set of two or more Users.</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="CopyrightExceptionPermission">
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Permission"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasRequired"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="CopyrightExceptionFact"/>
        </owl:someValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Permission to invoke one right exceptionally.</rdfs:comment>
```

```
  </owl:Class>
  <owl:Class rdf:ID="Copy">
    <rdfs:comment xml:lang="en">A mechanical reproduction of analogue or digital
representations of a given IP Entity. In the case of digital Copies the result is
virtually identical  while in the case of analogue Copies the results can vary
considerably in quality.</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="IPEntity"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#resultedFrom"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="MakeCopy"/>
        </owl:someValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <owl:disjointWith>
      <owl:Class rdf:ID="Work"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Product"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Manifestation"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Instance"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#User">
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <rdfs:comment xml:lang="en">Any person or legal entity in a Value-Chain
connecting (and including) Creator and End-User.</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:TransitiveProperty rdf:ID="belongsTo"/>
        </owl:onProperty>
        <owl:allValuesFrom rdf:resource="#Collective"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="Action"/>
        </owl:allValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#acts"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
    </rdfs:subClassOf>
  </owl:Class>
```

```
  <owl:Class rdf:ID="EndUserAction">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Action performed by an EndUser</rdfs:comment>
    <owl:disjointWith>
      <owl:Class rdf:ID="PublicCommunication"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class rdf:about="#Product"/>
        </owl:allValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#actedOver"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <mvco:rightGivenBy rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://purl.oclc.org/NET/mvco.owl#Distributor</mvco:rightGivenBy>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <owl:disjointWith>
      <owl:Class rdf:ID="Produce"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Action"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:about="#MakeInstance"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="MakeAdaptation"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Synchronise"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="MakeManifestation"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#MakeCopy"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Fact">
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <owl:disjointWith rdf:resource="#User"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Positive proposition.</rdfs:comment>
    <owl:equivalentClass>
      <owl:Restriction>
        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:ID="isTrue"/>
        </owl:onProperty>
      </owl:Restriction>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="#Synchronise">
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
    >1.0</owl:versionInfo>
    <mvco:impliesAlso rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://purl.oclc.org/NET/mvco.owl#Render</mvco:impliesAlso>
    <mvco:impliesAlso rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://purl.oclc.org/NET/mvco.owl#ModifyCopy</mvco:impliesAlso>
    <rdfs:comment xml:lang="en">The Action of concurrently performing/displaying
two distinct IP Entities each for a different human sense e.g. text and audio or
video and song</rdfs:comment>
    <mvco:rightGivenBy rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://purl.oclc.org/NET/mvco.owl#Creator</mvco:rightGivenBy>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#actedOver"/>
        </owl:onProperty>
        <owl:allValuesFrom>
          <owl:Class rdf:about="#Work"/>
        </owl:allValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Action"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Permission">
    <rdfs:comment xml:lang="en">Authorisation from one RightsOwner to one or more
Users to perform one or more Actions on a given IPEntity.</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom>
          <owl:Class rdf:about="#Action"/>
        </owl:someValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="permitsAction"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="issuedBy"/>
        </owl:onProperty>
        <owl:allValuesFrom rdf:resource="#User"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom rdf:resource="#Fact"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasRequired"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class rdf:about="#Action"/>
        </owl:allValuesFrom>
```

```
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#permitsAction"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#User"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#issuedBy"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Distributor">
    <rdfs:comment xml:lang="en">A User who distributes a Product</rdfs:comment>
    <owl:equivalentClass>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#acts"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="Distribute"/>
        </owl:someValuesFrom>
      </owl:Restriction>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#User"/>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
  </owl:Class>
  <owl:Class rdf:ID="MakeAdaptationInstance">
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="AdaptationInstance"/>
        </owl:allValuesFrom>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#resultsIn"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#MakeInstance"/>
          <owl:Restriction>
            <owl:onProperty>
              <owl:ObjectProperty rdf:about="#actedOver"/>
            </owl:onProperty>
            <owl:someValuesFrom>
              <owl:Class rdf:ID="AdaptationManifestation"/>
            </owl:someValuesFrom>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
```

```
    <rdfs:comment xml:lang="en">The Action of making an Instance from an
AdaptationManifestation</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="MoveContent">
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <rdfs:comment xml:lang="en">The action of moving the location of a
content</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#EndUserAction"/>
  </owl:Class>
  <owl:Class rdf:about="#MakeInstance">
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <owl:disjointWith>
      <owl:Class rdf:about="#Produce"/>
    </owl:disjointWith>
    <rdfs:comment xml:lang="en">The Action of making an Instance from a
Manifestation.</rdfs:comment>
    <owl:disjointWith>
      <owl:Class rdf:about="#PublicCommunication"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Synchronise"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Action"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#actedOver"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:about="#Manifestation"/>
        </owl:someValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:about="#MakeManifestation"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#resultsIn"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:about="#Instance"/>
        </owl:someValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Adaptation">
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <owl:disjointWith>
      <owl:Class rdf:about="#Work"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Manifestation"/>
    </owl:disjointWith>
    <rdfs:comment xml:lang="en">A Work that is derived from another
Work</rdfs:comment>
    <owl:disjointWith>
```

```
      <owl:Class rdf:about="#Instance"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#IPEntity"/>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#Copy"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
              <owl:Class rdf:about="#MakeAdaptation"/>
              <owl:Class rdf:about="#Synchronise"/>
            </owl:unionOf>
          </owl:Class>
        </owl:allValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#resultedFrom"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:about="#Product"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="UseData">
    <owl:disjointWith rdf:resource="#User"/>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Data documenting the functions Actions performed by a Device User on a
content item and the associated context</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="#MakeAdaptation">
    <owl:disjointWith>
      <owl:Class rdf:about="#MakeCopy"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Synchronise"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
              <owl:Class rdf:about="#Adaptation"/>
              <owl:Class rdf:about="#Work"/>
            </owl:unionOf>
          </owl:Class>
        </owl:allValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#actedOver"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#MakeInstance"/>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <owl:disjointWith>
      <owl:Class rdf:about="#MakeManifestation"/>
    </owl:disjointWith>
    <mvco:rightGivenBy rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://purl.oclc.org/NET/mvco.owl#Adaptor</mvco:rightGivenBy>
```

```
    <owl:disjointWith>
      <owl:Class rdf:about="#Produce"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#resultsIn"/>
        </owl:onProperty>
        <owl:allValuesFrom rdf:resource="#Adaptation"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">The Action of making an Adaptation</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Action"/>
    </rdfs:subClassOf>
    <mvco:rightGivenBy rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://purl.oclc.org/NET/mvco.owl#Creator</mvco:rightGivenBy>
    <owl:disjointWith>
      <owl:Class rdf:about="#PublicCommunication"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="MakeWorkInstanceCopy">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#MakeCopy"/>
          <owl:Restriction>
            <owl:onProperty>
              <owl:ObjectProperty rdf:about="#actedOver"/>
            </owl:onProperty>
            <owl:someValuesFrom rdf:resource="#WorkInstance"/>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <rdfs:comment xml:lang="en">The Action of making a
WorkInstanceCopy</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#resultsIn"/>
        </owl:onProperty>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="WorkInstanceCopy"/>
        </owl:allValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#IPEntity">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:ID="hasRightsOwner"/>
        </owl:onProperty>
      </owl:Restriction>
```

```
    </rdfs:subClassOf>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <rdfs:comment xml:lang="en">Types of IP Represented as Content: Work,
Adaptation, Manifestation, Instance...</rdfs:comment>
    <owl:disjointWith rdf:resource="#User"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class rdf:about="#Action"/>
        </owl:allValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#resultedFrom"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#Permission"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#hasRightsOwner"/>
        </owl:onProperty>
        <owl:allValuesFrom rdf:resource="#User"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="isMadeUpOf"/>
        </owl:onProperty>
        <owl:allValuesFrom rdf:resource="#IPEntity"/>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="ModifyCopy">
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <rdfs:comment xml:lang="en">Action of modifying a copy.</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#EndUserAction"/>
  </owl:Class>
  <owl:Class rdf:ID="MakeAdaptationManifestation">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class rdf:about="#AdaptationManifestation"/>
        </owl:allValuesFrom>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#resultsIn"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#MakeManifestation"/>
          <owl:Restriction>
            <owl:onProperty>
              <owl:ObjectProperty rdf:about="#actedOver"/>
            </owl:onProperty>
            <owl:someValuesFrom rdf:resource="#Adaptation"/>
          </owl:Restriction>
```

```
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <rdfs:comment xml:lang="en">The action of making an AdaptationManifestation
from an Adaptation.</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="#WorkManifestation">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Manifestation"/>
          <owl:Restriction>
            <owl:onProperty>
              <owl:ObjectProperty rdf:about="#resultedFrom"/>
            </owl:onProperty>
            <owl:someValuesFrom>
              <owl:Class rdf:ID="MakeWorkManifestation"/>
            </owl:someValuesFrom>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <rdfs:comment xml:lang="en">An object or event which is an expression of a
Manifestation of a Work</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="#CopyrightExceptionFact">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Fact related to the invokation of a CopyrightException</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#Fact"/>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
  </owl:Class>
  <owl:Class rdf:ID="MakeAdaptationInstanceCopy">
    <rdfs:comment xml:lang="en">The Action of making an
AdaptationInstanceCopy</rdfs:comment>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="AdaptationInstanceCopy"/>
        </owl:allValuesFrom>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#resultsIn"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#MakeCopy"/>
          <owl:Restriction>
            <owl:onProperty>
              <owl:ObjectProperty rdf:about="#actedOver"/>
            </owl:onProperty>
            <owl:someValuesFrom>
              <owl:Class rdf:about="#AdaptationInstance"/>
```

```
            </owl:someValuesFrom>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="#Produce">
    <mvco:rightGivenBy rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://purl.oclc.org/NET/mvco.owl#Instantiator</mvco:rightGivenBy>
    <mvco:rightGivenBy rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://purl.oclc.org/NET/mvco.owl#Creator</mvco:rightGivenBy>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#actedOver"/>
        </owl:onProperty>
        <owl:allValuesFrom>
          <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
              <owl:Class rdf:about="#Copy"/>
              <owl:Class rdf:about="#Instance"/>
              <owl:Class rdf:about="#Manifestation"/>
            </owl:unionOf>
          </owl:Class>
        </owl:allValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class rdf:about="#Product"/>
        </owl:allValuesFrom>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#resultsIn"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">The Function of making Products</rdfs:comment>
    <owl:disjointWith>
      <owl:Class rdf:about="#PublicCommunication"/>
    </owl:disjointWith>
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >1.0</owl:versionInfo>
    <owl:disjointWith rdf:resource="#Synchronise"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Action"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="WorkManifestationCopy">
    <rdfs:comment xml:lang="en">A copy of a WorkManifestation.</rdfs:comment>
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Copy"/>
          <owl:Restriction>
            <owl:someValuesFrom>
              <owl:Class rdf:ID="MakeWorkManifestationCopy"/>
            </owl:someValuesFrom>
            <owl:onProperty>
              <owl:ObjectProperty rdf:about="#resultedFrom"/>
            </owl:onProperty>
```