



Project Number:	FP7-257123
Project Title:	CONVERGENCE
Deliverable Type:	Report
Dissemination Level	Public
Deliverable Number:	D4.1
Contractual Date of Delivery to the CEC:	28.02.2011
Actual Date of Delivery to the CEC:	16.3.2011
Title of Deliverable:	Preliminary Definition of the Versatile Digital Item
Workpackage contributing to the Deliverable:	WP 4
Nature of the Deliverable:	Report
Editor:	Giuseppe Tropea
Authors:	Nicola Blefari Melazzi, Giuseppe Tropea, Stefano Salsano (CNIT), Maria Teresa Andrade, Helder Castro (INESC), Alina Hang (LMU), Lucian Corlan (UTI), Edoardo Radica, Leonardo Chiariglione (CEDEO), Andrea Pede, Richard Walker (XIWRITE), Angelos-Christos Anadiotis, Aziz Mousas (ICCS), Panagiotis Gkonis (SIL)
Abstract:	This deliverable details preliminary decisions regarding the structure of the VDI and how to implement them within the context of the MPEG-21 standard.
Keyword List:	MPEG-21, Digital Item, Semantics, Syntax, Ontology

Executive Summary

Goals of the deliverable

The goal of this deliverable is to present the preliminary design of the Versatile Digital Item structure, and to explain the rationale behind the design choices that have been made. The deliverable will also illustrate relationships between the VDI and the Digital Item (DI) defined in the official MPEG-21 standard.

An Annex contains specific proposals for the syntax of a set of extensions to the MPEG-21 Digital Item. These extensions cover many of the enhancements offered by the VDI design as it is presented in this document. These proposals have been submitted to:

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION ORGANISATION
ISO/IEC JTC1/SC29/WG11

CODING OF MOVING PICTURES AND ASSOCIATED AUDIO

(meeting of March 2011, Geneva, CH)

We delayed the deliverable with respect to the due date to include in it this standardization proposal.

INDEX

GOALS OF THE DELIVERABLE	2
GLOSSARY	5
1 GOALS AND STRUCTURE OF THIS DOCUMENT	11
2 REQUIREMENTS GUIDING THE VDI DEFINITION PROCESS	12
2.1 UNIQUE FEATURES OF CONVERGENCE	12
2.1.1 CONVERGENCE provides synchronization	12
2.1.2 CONVERGENCE VDIs “bundle” data from different sources	12
2.1.3 CONVERGENCE provides a unique identifier for people, products, media.....	13
2.1.4 CONVERGENCE allow users to define their own description schemes for VDIs.....	13
2.1.5 CONVERGENCE provides standardized support for the annotation of digital content	13
2.2 DETAILED REQUIREMENTS FROM USER SCENARIOS	13
3 STRUCTURE OF THE MPEG-21 DIGITAL ITEM	18
3.1 WHAT IS A DIGITAL ITEM?	18
3.2 IMPORTANT ELEMENTS USED INSIDE DIGITAL ITEM	18
3.2.1 Item	18
3.2.2 Component	18
3.2.3 Descriptor.....	18
3.2.4 Statement.....	18
3.2.5 Identification	19
3.2.6 Example of a Digital Item.....	21
4 THE LIFE OF A VDI.....	26
4.1 VDI’S MAIN ACTORS	26
4.2 VDI’S LIFE SPAN	26
5 VDI DESIGN CONCEPTS.....	30
5.1 THE SELF-CONTAINED VDI	30
5.2 UPDATING VDIs AND VDI SEQUENCES.....	31
5.2.1 Semantics of VDI sequences.....	32
5.2.2 VDI sequences CoMid level operations.....	34
5.2.3 VDI sequences and CoNet level operations	34
5.3 VDI RELATIONSHIPS	35
5.3.1 Relationships’ Usefulness	35
5.3.2 Relationships’ Declaration and Types.....	36



6	STRUCTURE OF THE VDI	38
6.1	CONFORMITY WITH THE MPEG STANDARD	38
6.2	VDI IDENTIFICATION	39
6.3	SEMANTIC CONV: RELATIONSHIPS INSIDE VDIS	40
6.3.1	Inter-VDI relationships vs. inter-Item relationships	41
6.3.2	VDI-level relationships	41
6.3.3	Item-level relationships	43
6.4	PROCESSING RELATIONSHIPS INTO CONV:METADATA BLOCKS	44
6.5	EMBEDDING RESOURCES IN VDIS	45
6.6	GRANULARITY OF THE VDI	45
6.6.1	Implications of VDI Removal	45
6.7	TRUSTING VDIS	46
6.7.1	Relationships and Rights	46
7	MAPPING VDIS TO SCENARIOS	47
7.1	LECTURE PODCAST, AN EXAMPLE MAPPING	47
7.1.1	Description of the model	47
7.1.2	Example mapping procedure	49
7.1.3	Application design and VDI bundles	54
7.2	EXAMPLE SYNTAX OF VDI RELATIONSHIPS	55
8	REFERENCES AND RELEVANT LITERATURE	56

Glossary

TERM	DEFINITION
Access Rights	Criteria defining who can access a VDI under what conditions
Advertise	Procedure used by a CoNet user to make a resource accessible to other CoNet users
Application	Software, designed for a specific purpose that exploits the capabilities of the CONVERGENCE System.
Business Scenario	A scenario describing one way in which the CONVERGENCE System may be used by specific users in a specific situation; more narrowly: a scenario describing the commercial products and services bought and sold in such a situation, the actors concerned and, possibly, the associated flows of revenue.
Clean-slate architecture	The CONVERGENCE implementation of the Network functional level, totally replacing existing IP functionality.
CoApp Provider	A user offering Applications running on the Convergence Middleware (CoMid)
CoMid	The CONVERGENCE middleware.
CoMid Provider	The user providing access to CoMid services.
CoMid Resource	A virtual or physical object or service, referenced by a VDI, e.g. a video file, a Real World Object, a person, an Internet service, etc.
Community Dictionary Service	A functional block belonging to the Content level of CONVERGENCE that provides CoMid with all the matching concepts in a user's subscription, search request and publication.
CoNet	The CONVERGENCE network level.
CoNet Provider	The user providing access to CoNet services, i.e. it is the corresponding of today's Internet Service Provider.
CoNet Resource	A resource of the CoNet that can be identified by means of a name; it can be either a Named data or a Named service access point
Content-based resource discovery	A user request for resources, either through a subscription to the CONVERGENCE System or a search request to the CONVERGENCE system. The CONVERGENCE System will then return a list of VDIs compatible with the search criteria.
Content-based	A subscription based on a specification of user's preferences or

Subscription	interests, (rather than a specific event or topic). In other terms, the subscription is based on the actual content of the considered events, which are not classified according to some predefined external criterion (e.g., topic name), but according to the properties of the events themselves.
Content-centric	A network paradigm in which the network directly provides users with contents, and is aware of which content is actually transported, instead of limiting itself to providing communication channels between hosts.
CONVERGENCE Applications level	A functional level of CONVERGENCE that establishes the interaction with CONVERGENCE users. The Applications Layer interacts with the other CONVERGENCE functional levels on behalf of the user.
CONVERGENCE Content level	A functional level of CONVERGENCE that provides the means to handle resources on the basis of “what” they contain and offer. These functionalities are implemented using a set of technologies that we call CONVERGENCE Middleware (CoMid) and the Community Dictionary Service (CDS).
CONVERGENCE Core Ontology	A semantic representation of the CoReST taxonomy.
CONVERGENCE Device	A device deploys functionality implemented by a functional block.
CONVERGENCE Framework	A software framework providing interfaces to the functionality of the CONVERGENCE System
CONVERGENCE Middleware	A functional block of the CONVERGENCE system, implementing functionality belonging to the Content level.
CONVERGENCE Network	A functional block of the CONVERGENCE system, implementing functionality belonging to the Network level.
CONVERGENCE Network level	A functional level of the CONVERGENCE system that provides access to named-resources on a public or private network infrastructure. A named-resource is any resource that can be identified by means of a name; named-resources may be either data or service-access-points. Examples of named-resources include: a VDI; an electronic document, an image, a source of information with a consistent purpose, the point of access to a service, and a collection of other resources.
CONVERGENCE	A list of concepts or terms that makes it possible to categorize



Resource Semantic Type	CONVERGENCE resources, establishing a connection with the resource's semantic metadata.
CONVERGENCE System	A system built by using the technologies specified or adopted by the CONVERGENCE specification.
Describe, Discover, Distribute	The DDD paradigm uses three different axes to aggregate the pillar functionalities offered by the CoMid functional block of CONVERGENCE.
Digital forgetting	Techniques designed to ensure that VDIs do not remain accessible for indefinite periods of time, when this is not the intention of the user
Digital Item	A structured digital object with a standard representation, identification and metadata. A DI consists of content, content and context related metadata, and structure. The structure is given by a Digital Item Declaration (DID) that links content and metadata.
Discussion	A set or graph of messages each containing links to other messages. In the context of CONVERGENCE discussions and messages within discussions may be represented as VDIs.
Domain ontology	An ontology, dedicated to a specific knowledge domain or application
Elementary Service	A concept imported from the MPEG-M emerging standard. Refers to the most basic unit of functionality offered by the CoMid. ES provide CoMid's main functionality.
Entity	An object which an Elementary Service can act upon or with which it can interact with.
Expiry date	The last date on which a VDI may be legitimately used by a user of the CONVERGENCE System. The last day on which the CONVERGENCE system will allow a user to find the VDI in a search or subscription or to retrieve the VDI.
Fractal	A semantically defined virtual cluster in a distributed overlay network composed of all CONVERGENCE peers running CoMid.
Functional block	Partial or complete implementations of the functionality required by a specific level of the CONVERGENCE architecture.
Functional level	An aggregated set of conceptually similar functional blocks.
License	A machine-readable expression of Operations that may be executed by a Principal
Local named resource	A named-resource made available to the CONVERGENCE users



	through a local device, permanently connected to the network.
Metadata	Data describing a resource references by a VDI, including but not limited to provenance, classification, access rights, expiry date etc.
MPEG eXtensible Middleware (MXM)	A standard Middleware specifying a set of Application Programming Interfaces (APIs) so that MXM Applications executing on an MXM Device can access the standard multimedia technologies contained in the Middleware as MXM Engines
MPEG-M	An emerging standard proposed by MPEG as an extension of the MXM standard.
Multihoming	In the context of IP networks, the configuration of multiple network interfaces or IP addresses on a single computer.
Named data	A named-resource consisting of data.
Named resource	A CoNet resource that can be identified by means of a name. Named-resources may be either data (in the following referred to as “named-data”) or service-access-points (“named-service-access-points”)
Named service access point	A kind of named-resource, consisting of a service access point identified by a name.
Network Identifier	An identifier identifying a named resource in the CONVERGENCE Network. If the named resource is a VDI, its NID may or may not be identical to the VDI identifier (to be decided)
Node	A CONVERGENCE device that implements CoNet functionality
Overlay architecture	An implementation of CoNet as an overlay over IP.
Parallel architecture	An implementation of CoNet as a new networking layer that can be used in parallel to IP.
Peer	A CONVERGENCE device that implements CoMid and CoNet functionality.
Policy routing	In the context of IP networks, a collection of tools for forwarding and routing data packets based on policies defined by network administrators.
Publish	The act of making a VDI available to the users or to a subset of the users of the CONVERGENCE System
Publisher	A user of CONVERGENCE who advertises resources on the CONVERGENCE system, thus making them available to subscribers.
Publish-subscribe	A service model based on an asynchronous exchange of messages or



model	events. The CONVERGENCE publish subscribe model, encompasses a set of clients that publish VDIs, which are then forwarded to clients who have expressed interest in receiving them.
Real World Object	An object existing in the real (as opposed to the virtual) world.
Resource	A virtual or physical object or service referenced by a VDI, e.g. media, Real Life Objects, persons, internet services
Scope (in the context of routing)	In the context of advertising and routing, the geographical or administrative domain on which a network function operates (e.g. a well defined section of the network - a campus, a shopping mall, an airport -, or to a subset of nodes that receives advertisements from a service provider)
Search	The act through which a user requests a list of VDIs meeting a set of search criteria (e.g. specific key value pairs in the metadata, key words, free text etc.)
Service Level Agreement	An agreement between a service provider and another user of CONVERGENCE to provide the latter with a service whose quality marches parameters defined in the agreement
Subscribe	The act whereby a users requests notification every time another user publishes or updates a VDI that satisfies user-defined subscription criteria (key value pairs in the metadata, free text, key words etc.). Note: subscription criteria should be formulated in the same way as search criteria)
Subscriber	A user of CONVERGENCE who declares his/her interest in being formed about VDIs.
Subscription	The act of a User indicating his/her interests to the Content level of CONVERGENCE using keywords, free text or similarity with a resource.
Timestamp	A machine-readable representation of a date and time.
Trials	Organized tests of the CONVERGENCE System in specific business scenarios
Unnamed data	A data resource with no NID.
User	An entity outside the CONVERGENCE system, which interacts with the system and uses what the system delivers.
User ontology	An ontology (a set of concepts and their relationships), created by users of CONVERGENCE when publishing a VDI or subscribing to a VDI.



User Profile	A description of the attributes and credentials of a user of the CONVERGENCE System
VDI Browser	A tool allowing users to browse and consume VDIs on the network as allowed by their access rights
VDI Creator/Editor	A tool allowing users to create, publish, read, update and delete VDIs
VDI Identifier	A unique signifier assigned to a VDI or components of a VDI.
Versatile Digital Item	A structured, hierarchically organized, digital object containing one or more resources and metadata, including a declaration of the parts that make up the VDI and the links between them.

1 Goals and structure of this document

Deliverable 4.1 (Preliminary Definition of the Versatile Digital Item) is the first deliverable from WP4: Definition of the Versatile Digital Item. The general goal of WP4 is to define **the Versatile Digital Item**, extending the scope of the MPEG-21 DI by including new classes of objects, including Real World Objects, services and people and supporting new classes of operation.

The specific goal of this deliverable is to present the preliminary design of the VDI. The preliminary design phase focused on:

- Filtering and analysing requirements that directly influence the VDI design process
- Studying the MPEG-21 Digital Item standard to understand requirements that are already well covered
- Designing solutions for
 - the structure and number of VDI identifiers
 - explicit typed links between semantically interconnect VDIs
 - logically updating VDIs with newer versions transparently to the user and in coherence with the MPEG-21 requirements for the DI

These solutions may be considered also as extensions to the standard DI defined in MPEG-21.

This document is structured as follows:

Chapter 2 presents and discusses requirements, and input from WP2

Chapter 3 presents the MPEG-21 Digital Item

Chapter 4 gives a brief overview of the phases in the life cycle of a typical VDI

Chapter 5 presents the design challenges that were faced and outlines the high-level solution adopted

Chapter 6 presents the details of the proposed extensions and their workings

Chapter 7 maps the various concepts onto a real-world example taken from the user scenarios

This deliverable is complemented by an Annex: a preliminary document that formalizes and presents the Versatile DI extensions to the MPEG standardization community.

2 Requirements guiding the VDI definition process

2.1 Unique features of CONVERGENCE

WP2 has identified some CONVERGENCE's "Unique Selling Points", reporting its findings in D2.1. Here we cite the "selling points" that have directly influenced the design of the basic structure of the Versatile DI and its new functionality, and the proposed enhancements to the MPEG-21 Digital Item.

1. CONVERGENCE allows synchronization between content at different locations and guarantees that content will always be up to date
2. CONVERGENCE "bundles" data from different sources into a single, self-consistent package
3. CONVERGENCE provides a unique identifier for people, products, media and other real world objects
4. CONVERGENCE allow users to define their own description schemes for VDIs
5. CONVERGENCE provides standardized support for the annotation of digital content, improving interoperability between systems

In what follows, we will briefly describe these features and the way they have guided the preliminary design process.

2.1.1 *CONVERGENCE provides synchronization*

A key feature of CONVERGENCE is that VDIs can be synchronized. For example, participants in the focus groups suggested this feature could be used to synchronize libraries of video content in different locations. In the retail scenario a user manual is updated as repair shops acquire new experience about possible problems. Compared to the current situation in which user manuals and items information are mainly published in paper format, VDIs represent a great improvement.

2.1.2 *CONVERGENCE VDIs "bundle" data from different sources*

A VDI is defined by CONVERGENCE as a self-consistent bundle of information that always "travels together". This remains true even when some of the information is not physically present in the VDI but is made available via a link. VDIs can be used to package a complete library of video resources, making it easy for end-users to find documents they frequently lose, such as warranties, receipts and user manuals, or package collections of video clips, photos and slides, or entire discussions. VDIs can be logically nested inside other VDIs where they can be manipulated either as a single unit or one by one. This is a requirement for many applications in the area of collaborative work (in the e-learning domain see [IELOM] [HYTEC]).

Current technology provides several methods of bundling information (e.g. in ZIP files). However, the technology is mainly of interest to users with a technical background. This is a case, therefore, in which CONVERGENCE offers value to its users.

2.1.3 CONVERGENCE provides a unique identifier for people, products, media

Every VDI has a unique identifier, which, if necessary, can be associated with a unique real-world object. This feature plays a valuable role in several scenarios. CONVERGENCE's unique identifiers are an enabler for stock management, facilitate product recalls and allow product support and maintenance staff to identify products in the hands of customers. It is easy to imagine a very broad range of situations in which they could be useful for companies, service-providers and end-users.

2.1.4 CONVERGENCE allow users to define their own description schemes for VDIs

Many online services allow users to tag and/or provide descriptions of media resources they are making available to other users. In all cases, however, the schema for entering comments and tags is defined by the service provider, often neglecting issues of importance to particular groups of user. This means, for instance, that the only way for a media professional to develop an innovative description schema is to set up her own proprietary service. In CONVERGENCE, by contrast, this possibility is an intrinsic part of the system.

2.1.5 CONVERGENCE provides standardized support for the annotation of digital content

There are many professional and consumer applications, in which it is desirable for a user to add comments to resources produced by another user. Several online services (e.g. Google sideWiki) provide users with this kind of functionality. However, they are always proprietary and there is no standard solution. CONVERGENCE, on the other hand offers a standard solution, which can work for many different kinds of content.

2.2 Detailed requirements from User Scenarios

The table below provides a condensed list of user requirements for the VDI. Some of the requirements have been taken from the full list of CONVERGENCE requirements specified in D2.1 Others were taken from revised versions of the scenario and use case descriptions, following this numbering scheme:

1. Scenario Alinari: Managing and annotating a large photo archive
2. Scenario FMSH: Building, managing and exploiting audiovisual archives in research and education
3. Scenario LMU: Augmented Lecture Podcast
4. Scenario ICCS-SIL-UTI-WIPRO: Smart Retailing

These requirements have guided the design of the VDI structure and will continue to support design decisions in subsequent deliverables. The following table shows how these

requirements can be broadly separated into two classes: requirements which are already covered by the concept of Digital Item as it is defined in the MPEG-21 standard, and requirements which imply extensions to that standard, hence justifying the introduction of a Versatile DI. These additional requirements are shown in boldface.

#	Requirement	User Scenario			
		1	2	3	4
	VDI Properties				
1	A VDI shall be capable of referencing arbitrary classes of resources: Persons, Real World Objects, Barcode, Videos, Images, Discussion, VDIs, etc.	X	X	X	X
2	A VDI shall have a unique identifier	X	X	X	X
3	It shall be possible to define an expiry date for a VDI. If no expiry date is defined, a default expiry date shall be set	X	X	X	X
4	It shall be possible to create a template for VDIs	X	X		
5	It shall be possible to create semantically specialized versions of a VDI starting from other, more generic, VDIs		X		X
6	It shall be possible to retrieve ancestors of a nested VDI			X	
	Access Rights, Encryption, Security				
7	A VDI shall define Access/Alter Rights and Licensing mechanisms	X	X	X	X
8	It shall be possible to sign a VDI or particular fields of a VDI				X
9	It shall be possible to encrypt VDIs or particular fields of a VDI (e.g. content, comments, metadata, etc.)	X			X
10	A VDI shall define its visibility (e.g.: public, private, only visible for a certain group, etc.). There may be different levels of visibility for different parts of the VDI, depending on access rights			X	
11	It shall be possible to confirm the authenticity of the contents of VDIs				
12	It shall be possible to have tamper-proof parts of VDIs				
13	It shall be possible to use a VDI to access content and services			X	
14	A VDI may contain anonymous segments				
15	A VDI shall support Management and Protection of Intellectual Property Rights	X	X		
16	A VDI shall be able to define the ownership of a resource	X	X	X	X
17	It shall be possible to change the owner of a VDI				X
	Metadata				
18	It shall be possible to define user-defined tags for VDIs	X	X		X
19	It shall be possible to tag areas of resources (e.g. photos, or associate a video with timestamps identifying the beginning, the end or the		X		



	duration)				
20	Users shall have the possibility to generate annotations using other VDIs, hence allowing keywords, text, visual icons, or spoken text annotations		X		
21	It shall be possible to define resource-specific attributes in the VDI (e.g. author, owner, time, date, location, etc.)	X	X	X	X
22	A VDI shall be capable of describing the relationship between two VDIs (e.g. synchronization of time-dependent (video) and time-independent (slides) components; segments of videos with subtitles)		X	X	
23	A VDI can describe segments of a resource		X	X	
24	It shall be possible to classify information in a VDI				
25	It shall be possible to include different types of information (time, location)				
26	It shall be possible to include automatically generated metadata into the VDI			X	
27	A VDI may have relations with other VDIs based on generic and domain-specific ontologies		X		
28	A VDI can be used to structure / segment a resource (e.g. video)		X	X	
	Operations				
29	It shall be possible to alter/update fields of a VDI. The changes are applied to all related VDIs, also to VDIs nested in other VDIs.	X	X	X	X
30	It shall be possible to nest VDIs in another VDI			X	
31	It shall be possible to publish VDIs anonymously			X	
32	It shall be possible to republish VDIs		X		X
33	It shall be possible to search for a VDI / search in a VDI, taking into account its semantic relationships with other VDIs	X	X	X	X
34	It shall be possible to subscribe to parts of a VDI			X	
35	It shall be possible to link VDIs to other VDIs			X	
36	A VDI may have typed links to other VDIs (e.g. the VDI it is derived from, the full version of a VDI, etc.)	X	X	X	X
	History				
37	It shall be possible to track the history of a VDI (e.g. by a version number)			X	



The focus of this deliverable is on the boldface set of requirements. The aim is to propose general solutions suitable for adoption within the MPEG framework. Key features include the introduction of semantic relationships between VDIs and between parts of the same VDI, the adoption of a versioning system for VDIs which runs in the CONVERGENCE infrastructure transparently to the user, and the creation of a clean mapping between a unique VDI identifier at the Content level and the corresponding identifier at the Network level.

The following chapters will give further details about these techniques and the way CONVERGENCE proposes to use them to enhance the specification of the MPEG-21 Digital Item.

Let us consider an example. In the Lecture Podcast user scenario, a podcast is a nested VDI composed of episodes that are in turn composed of video and slides. If a student downloads the VDI representing the Podcast, she can subscribe to the whole lecture podcast or to parts of it (e.g. only video, only slides...). This is because the information is split into several VDIs linked via semantically typed relations. This scheme fulfills the requirement for “partial subscriptions” listed in the requirements Table. Details of this approach are given in the following chapters, where it is shown that nested VDIs are just separate VDIs connected by relationships. This means they do not need to sit inside the same "file". This distributed approach allows users to subscribe either to the topmost VDI or to one of the VDIs it links to. The net effect is that users can subscribe either to the whole podcast or to one of its components.

3 Structure of the MPEG-21 Digital Item

3.1 What is a Digital Item?

The MPEG-21 Digital Item is the basic unit of transaction in the MPEG-21 framework. A Digital Item is a combination of resources (video, audio tracks, images, etc...), metadata (creator, resource category, etc...), an event monitor and intellectual property information (license, encryption resource information).

3.2 Important elements used inside Digital Item

3.2.1 *Item*

An item is a grouping of sub-items and/or components that are bound to relevant descriptors. Descriptors contain information about the item. Items may contain options, which allow them to be customized or configured. Items may also be conditional (on predicates asserted by selections defined in the choices). An item that contains no sub-items can be considered an entity -- a logically indivisible work. An item that does contain sub-items can be considered a compilation -- a work composed of potentially independent sub-parts. Items may also contain annotations to their sub-parts.

The relationship between items and Digital Items (as defined in [MP21-1]) could be stated as follows: items are declarative representations of Digital Items.

3.2.2 *Component*

A component is the binding of a resource to all of its relevant descriptors. These descriptors are information related to all or part of the specific resource instance. Such descriptors will typically contain control or structural information about the resource (such as bit rate, character set, start points or encryption information) but not information describing the “content” within.

It should be noted that a component itself is not an item; components are building blocks for items.

3.2.3 *Descriptor*

A descriptor associates information with the enclosing element. This information may be a component (such as a thumbnail of an image, or a text component), or a textual statement.

3.2.4 *Statement*

A statement is a literal textual value that contains information, but not an asset. Examples of likely statements include descriptive, control, revision tracking or identifying information.

3.2.5 Identification

The 'Digital Item Identification' [MP21-3] standard defines the syntax and semantics of the Digital Item Identifier and of the 'RelatedIdentifier' elements. These elements make it possible to uniquely identify the Digital Item and associating identifiers that are related to the Digital Item, container, component, and/or fragment thereof, which do not identify the Digital Item directly. These identifiers will be in the form of a URI.

Identifier

Digital Items and their parts within the MPEG-21 Multimedia Framework are identified by encapsulating Uniform Resource Identifiers (URI) in the Identifier element.

Syntax:

```
<xsd:element name="Identifier" type="xsd:anyURI"/>
```

Semantics:

This element contains an identifier for a Digital Item, container, component, and/or fragment thereof in the form of a URI. A Registration Authority maintains a list of identification schemes compliant with ISO/IEC 21000-3. Identifiers are not required to be registered with the Registration Authority to be conformant to this clause of the specification.

This definition of the Digital Item Identifier is the basis for the definition of the VDI identifier adopted in CONVERGENCE. Each VDI has a unique identifier, the VDI ID, which follows the above definition of the DII standard.

While the MPEG standard dictates that if you change a single bit of a VDI you must change the VDI identifier, the RelatedIdentifier offers great flexibility in associating multiple identifiers to the same Digital Item when they are needed for different purposes.

RelatedIdentifiers

While the Digital Item Identifier element enables the unique identification of Digital Items (or parts thereof), the RelatedIdentifier element allows the identification of information that is *related to* the Digital Item (or parts thereof).

Syntax:

```
<xsd:element name="RelatedIdentifier" type="xsd:anyURI"/>
```

Semantic:

The RelatedIdentifier element may not be used for identifying the Digital Item (or part thereof). This shall be done by using the Digital Item Identifier element as specified above.

A Registration Authority maintains a list of identification schemes to be used within ISO/IEC 21000-3. Identifiers are not required to be registered with the Registration Authority to be conformant to this clause of this specification.

The Digital Item Identification MPEG standard contains an example, copied in the following XML snippet, which clarifies the use of this element. The example shows how to uniquely identify a resource (an MPEG Audio Layer III-coded sound recording) within a Digital Item using an International Standard Recording Code (ISRC). The example also highlights how to associate a related identifier (here: identifying the underlying music work with an International Standard Work Code (ISWC)) with such a resource.

```
<?xml version="1.0"?>
<DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS"
      xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS">
  <Item>
    <Component>
      <Descriptor>
        <Statement mimeType="text/xml">
          <dii:Identifier>urn:mpegRA:mpeg21:dii:
            isrc:US-Z03-99-32476</dii:Identifier>
          <!-- ISRC identifying the sound recording -->
        </Statement>
      </Descriptor>
      <Descriptor>
        <Statement mimeType="text/xml">
          <dii:RelatedIdentifier>urn:mpegRA:mpeg21:dii:
            iswc:T-034-524-680-1</dii:RelatedIdentifier>
          <!-- ISWC identifying the underlying musical work -->
        </Statement>
      </Descriptor>
      <Resource ref="Track01.mp3" mimeType="audio/mp3"/>
    </Component>
  </Item>
</DIDL>
```

Having multiple Resources embedded inside the same Component means that the various Resources are totally equivalent, and an agent may choose any one of them. On the other hand, having multiple Components grouped inside one Item means that the Item is composed

of several different pieces of information. Moreover, each Component can have its own Descriptors that provide additional information about the component.

The structure of the DI also allows for an Item to be composed of a sequence of sub-Items. So what is the difference between having multiple Items and having multiple Components inside an embedding Item? A hint of an answer is contained within the MPEG-21 standard itself, where it says “ITEMs are intended to be the lowest level of granularity visible to an end-user. In other words, a user interface could allow end-users to access the ITEMs within an ITEM, but not the COMPONENTs within an ITEM”.

3.2.6 Example of a Digital Item

The Digital Item contains an ITEM (top level item), called “content item”, which represents the entire resource carried by the Digital Item. This content item could be an Audio Compilation. The content item contains one or more ITEMs (inner items), called “content element item”, which represent the resources that make up the content item.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<didl:DIDL xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS"
  xmlns:ipmpdidl="urn:mpeg:mpeg21:2004:01-IPMPDIDL-NS"
  xmlns:ipmpinfo="urn:mpeg:mpeg21:2004:01-IPMPINFO-NS"
  xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
  xmlns:ipmpinfo_msx="urn:mpeg:maf:Schema:mediastreaming:
    IPMPINFOextensions:2007"
  xmlns:mpeg4ipmp="urn:mpeg:mpeg4:IPMPSchema:2002"
  xmlns:ipmpmsg="urn:mpeg:mpegB:schema:IPMP-XML-MESSAGES:2007"
  xmlns:xenc="http://www.w3.org/2001/04/xmenc#"
  xmlns:rel-r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
  xmlns:rel-sx="urn:mpeg:mpeg21:2003:01-REL-SX-NS"
  xmlns:rel-m1x="urn:mpeg:mpeg21:2005:01-REL-M1X-NS"
  xmlns:rel-mx="urn:mpeg:mpeg21:2003:01-REL-MX-NS"
  xmlns:rel-m2x="urn:mpeg:mpeg21:2006:01-REL-M2X-NS"
  xmlns:rel-m3x="urn:mpeg:mpeg21:2006:01-REL-M3X-NS"
  xmlns:rel-wimtv="urn:wimTV:2009:01-REL-WIMTV"
  xmlns:didl_msx="urn:mpeg:maf:schema:mediastreaming:
    DIDLextensions"
  xmlns:didl="urn:mpeg:mpeg21:2006:07-DIDL-NS"
  xmlns:erl="urn:mpeg:mpeg21:2005:01-ERL-NS"
  xmlns:mpeg7smp="urn:mpeg:mpeg7:smp:schema:2001"
  xmlns:didmodel="urn:mpeg:mpeg21:2002:02-DIDMODEL-NS"
  xmlns:dmprrp="urn:dmp:idp:represent:PaymentProtocol:2008"
  xmlns:mxmlbp="urn:mpeg:mpeg-m:schema:baseprotocol:2009">
  <didl:Item id="AO">
```

```
...  
<didl:Item id="B0">  
  ...  
</didl:Item>  
<didl:Item id="B1">  
  ...  
</didl:Item>  
  ...  
</didl:Item>  
</didl:DIDL>
```

Both the top level and the inner items contain more than one DESCRIPTOR and COMPONENT element.

```
<didl:DIDL>  
  <didl:Item id="A0">  
    <didl:Descriptor>  
      ...  
    </didl:Descriptor>  
    <didl:Descriptor>  
      ...  
    </didl:Descriptor>  
    <didl:Item id="B0">  
      <didl:Descriptor>  
        ...  
      </didl:Descriptor>  
      <didl:Component>  
        ...  
      </didl:Component>  
    </didl:Item>  
  </didl:Item>  
</didl:DIDL>
```

One of the DESCRIPTOR elements contains an IDENTIFIER element.

```
<didl:Descriptor>  
  <didl:Statement mimeType="text/xml">  
    <dii:Identifier>  
      urn:cedeo:wimtv:content:5a220733-edf1-47a5-9380-fb10af01ce63  
    </dii:Identifier>
```

```
</didl:Statement>
</didl:Descriptor>
```

The top level item contains a DESCRIPTOR element which provides metadata describing the content (e.g. creator, date of creation, place of creation and so on.) The metadata may be Mpeg7 compliant.

```
<didl:Descriptor>
  <didl:Statement mimeType="text/xml">
    <didl_msx:Metadata>
      <didl_msx:StructuredData ref="urn:mpeg:mpeg7:smp:schema:2001">
        <mpeg7smp:Mpeg7>
          <mpeg7smp:Description
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:type="mpeg7smp:CreationDescriptionType">
            <mpeg7smp:CreationInformation>
              <mpeg7smp:Creation>
                <mpeg7smp:Title>Casanova</mpeg7smp:Title>
                <mpeg7smp:Abstract>
                  <mpeg7smp:FreeTextAnnotation>
                    casanova presentation
                  </mpeg7smp:FreeTextAnnotation>
                </mpeg7smp:Abstract>
                <mpeg7smp:Creator>
                  <mpeg7smp:Role
                    href="urn:mpeg:mpeg7:cs:RoleCS:2001:DIRECTOR" />
                  <mpeg7smp:Agent xsi:type="mpeg7smp:PersonType">
                    <mpeg7smp:Name>
                      <mpeg7smp:GivenName>Pinco</mpeg7smp:GivenName>
                      <mpeg7smp:FamilyName>Pallino</mpeg7smp:FamilyName>
                    </mpeg7smp:Name>
                  </mpeg7smp:Agent>
                </mpeg7smp:Creator>
              </mpeg7smp:Creation>
            </mpeg7smp:CreationInformation>
          </mpeg7smp:Description>
        </mpeg7smp:Mpeg7>
      </didl_msx:StructuredData>
    </didl_msx:Metadata>
  </didl:Statement>
</didl:Descriptor>
```

Besides containing identifier and metadata the content element item also contains many other elements, in particular an Event monitor and Resource element. The Event monitor is useful to specify what actions can be monitored on the content element item. For instance the creator can be informed when a user plays a resource for a specific period of time.

Resource contains: the reference to a resource, for example a track of an audio compilation; the license (see [MP21-5]) specifies rights that may be exercised by the user on specific resources; the tool element contains useful information on the use of the resource, including encryption information (see IPMP standard [MP21-4]).

```
<didl:Component>
  <didl:Resource mimeType="application/mp21-ipmp">
    <ipmpdidl:ProtectedAsset mimeType="video/quicktime">
      <ipmpdidl:Identifier>
        urn:cedeo:wimtv:contentelement:658931e0-8a54-4d2f-8e4e-7b9560abf403
      </ipmpdidl:Identifier>
      <ipmpdidl:Info>
        <ipmpinfo:IPMPInfoDescriptor>
          <ipmpinfo:Tool>
            ...
          </ipmpinfo:Tool>
          <ipmpinfo:RightsDescriptor>
            <ipmpinfo:License>
              <rel-r:license
                licenseId="23e03f72-f55e-485c-a6d8-2f836963b47d">
                ...
              </rel-r:license>
            </ipmpinfo:License>
          </ipmpinfo:RightsDescriptor>
        </ipmpinfo:IPMPInfoDescriptor>
      </ipmpdidl:Info>
      <ipmpdidl:Contents ref="BKvrEbFDSvg.mov" />
      </ipmpdidl:ProtectedAsset>
    </didl:Resource>
  </didl:Component>
```

The creator can use licenses to define what rights to grant a user. For example the creator can grant the right to play but not to copy.

```
<rel-r:license licenseId="23e03f72-f55e-485c-a6d8-2f836963b47d">
```



```
<rel-r:grant>
  <rel-r:principal varRef="667ca6d7-74a4-4d1e-8e72-e4a196a95be4" />
  <rel-mx:play />
  <rel-m1x:protectedResource
    licensePartIdRef="bf5db91a-86ed-4bab-90b9-73a0b97996ba" />
  <rel-sx:feePerUse>
    <rel-sx:rate>
      <rel-sx:amount>10.0</rel-sx:amount>
      <rel-sx:currency>iso:IPY</rel-sx:currency>
    </rel-sx:rate>
    <rel-sx:to>
      <dmprrp:ServiceInformation>
        <dmprrp:ServiceID>205</dmprrp:ServiceID>
        <dmprrp:ServiceURL>
          http://localhost:6996/services/
        </dmprrp:ServiceURL>
      </dmprrp:ServiceInformation>
    </rel-sx:to>
  </rel-sx:feePerUse>
</rel-r:grant>
<rel-r:issuer>
  <rel-r:keyHolder>
    <rel-r:info>
      <dsig:KeyName>cre</dsig:KeyName>
      <dsig:KeyValue>
        <dsig:RSAKeyValue>
          <dsig:Modulus>AKa+vbrBcuF ... o+VXVNORpwjusGJv</dsig:Modulus>
          <dsig:Exponent>AQAB</dsig:Exponent>
        </dsig:RSAKeyValue>
      </dsig:KeyValue>
    </rel-r:info>
  </rel-r:keyHolder>
</rel-r:issuer>
</rel-r:license>
```

Using the Tool element, the creator can specify how resources have been encrypted. Within this element the creator can insert encryption keys, a reference to the algorithm used, the control point at which it was encrypted (e.g. before decoding video/audio, after decoding, ...) and so on.

4 The Life of a VDI

Below we depict the main states of a VDI, from a lifecycle viewpoint.

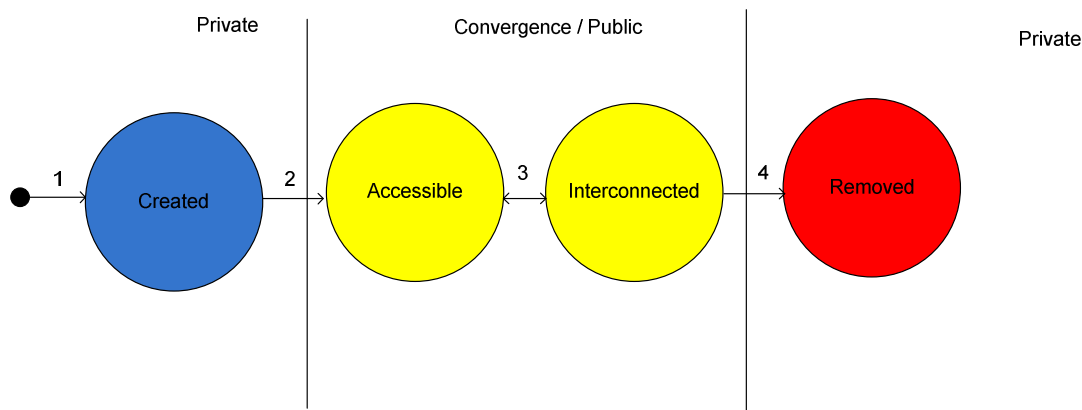


Figure 1: The VDI life cycle

The VDI is considered to be private until it is published on the CONVERGENCE system. From that moment on, the VDI will be in the public space, where the CONVERGENCE system can act on it, within the limits defined by its access rights. Once it is unpublished, the VDI will once again become visible only to the publisher.

4.1 VDI's main actors

The main actors involved in the life of a VDI are:

The Creator – creates and manages the VDI throughout its life span.

The User – searches for, subscribes to, visualizes, interconnects, and creates new versions of the VDI according to her access rights.

CONVERGENCE system – stores, retrieves and manages the VDI while it is accessible to the public.

4.2 VDI's life span

We can identify the following fundamental phases in the typical life cycle of a VDI.

Phase 1. VDI Creation

In this phase, a user wants to package certain resources into a set of VDIs. To produce a new VDI, the creator:

- defines the metadata for the VDI;
- attaches resource(s);
- includes and/or links any additional attachments and/or other VDIs;

- chooses an identification scheme and requests a Content Identification Service to assign the VDI identifier;
- defines the license and signs the VDI.

The VDI is now **created** in the user's personal space.

Once the VDI is created it contains the following structure:

VDI CONTENT	SOURCE	DETAILS
Creator	Creator	Details about the user who created the VDI
Searchable metadata	Creator	Ontology instances, keywords
Resources	Creator	Links to resources or embedded resources
Relationships	Creator	Semantic references to other VDIs
Security data	Creator/System	Creation date, expiry date, license, digital signature
VDI Identifier	Creator/System	Unique identifier for the VDI
Status data	System	Created
History	System	Links to previous versions of VDI

Phase 2. VDI Publishing

This is the phase in which the VDI moves from the private space into the CONVERGENCE realm. The creator publishes the VDI on CONVERGENCE by invoking the relevant CoMid functionalities to gossip and advertise its content in CoMid and store it in CoNet, thus making it **accessible** to the public.

The VDI is now ready to:

- be retrieved in searches and viewed by authorized users (according to the defined access rights);
- be found by user subscriptions;
- generate notifications when specific events occur.

From now on some of the information contained in the VDI (highlighted in boldface) is propagated through the System.

VDI CONTENT	SOURCE	DETAILS
Creator	Creator	Details about the user who created the VDI
Searchable metadata	Creator	Ontology instances, keywords
Resources	Creator	Links to or embedded resources
Relationships	Creator	Semantic references to other VDIs
Security data	Creator/System	Creation date, expiry date, license, digital signature
VDI Identifier	Creator/System	Unique identifier of VDI
Status data	System	Created

History	System	Links to previous versions of VDI
----------------	---------------	--

Phase 3. Interconnecting VDIs

This is the most active phase for VDIs. A web of semantic relationships can enrich the VDI throughout its lifespan, connecting it with other VDIs through such mechanisms as versioning and inter-VDI relationships (corrections, responses, extensions). These mechanisms are described in detail in the following chapters.

The VDI can be changed through a number of actions:

- the publisher modifies the VDI;
- users with the appropriate access rights modifies the VDI;
- users associate it to other VDIs (through links, comments, extensions, etc).

Each change produces a new version of the VDI, that is linked back to previous versions. The new VDIs that replace older versions can potentially hold completely different information. However some fields (Creator, History) will remain the same. In the table below, these fields are highlighted in boldface:

VDI CONTENT	SOURCE	DETAILS
Creator	Creator	Details about the user who created the VDI
Searchable metadata	Creator	Ontologies instances, keywords
Resources	Creator	Links to or embedded resources
Relationships	Creator	Semantic references to other VDIs
Security data	Creator/System	Creation date, expiry date, license, digital signature
VDI Identifier	Creator/System	Unique identifier of VDI
Status data	System	Created
History	System	Links to previous versions of VDI

Phase 4. VDI Removal

This phase marks the end of the VDI lifecycle. The VDI is **removed** from the CONVERGENCE system, and is no longer available to the public. This occurs when:

- the VDI expires;
- the publisher revokes the VDI.

Following removal, the status data is updated to specify that the VDI is inactive, and the reason (expiry or revocation, in boldface).

VDI CONTENT	SOURCE	DETAILS
Creator	Creator	Details about the user who created the VDI
Searchable metadata	Creator	Ontologies instances, keywords



Resources	Creator	Links to or embedded resources
Relationships	Creator	Semantic references to other VDIs
Security data	Creator/System	Creation date, expiry date, license, digital signature
VDI Identifier	Creator/System	Unique identifier of VDI
Status data	System	Expired / Revoked
History	System	Links to previous versions of VDI

All of the actions from the phases “2. Publishing”, “3. Interconnecting” and “4. Removal” generate notifications to subscribers. In this way subscribers can find out when a targeted VDI has been published, modified (updated with new version, interconnected) or unpublished.

5 VDI design concepts

Input to the VDI design process comes from three main sources: user-driven requirements, technical-driven requirements and coherence with existing standards.

User-driven requirements on the structure of the VDI come from the overall list of CONVERGENCE requirements in D2.1, from the use cases and from the preliminary business scenarios described in D9.1. Technical-driven requirements on the structure of VDI come from the architecture of the CONVERGENCE system, as defined in deliverable D3.1. The architecture work has been particularly important for the definition of the fields necessary to support semantic search and gossiping and for the structure of the network identifier and of its relationships with the VDI identifier. This last issue is in turn influenced by the need to maintain coherence with the MPEG-21 DI standard.

In the next chapter, we will provide details on the syntax and semantics of VDI identifiers, relationships, metadata and on semantic schemes associated with VDIs, and how they are implemented in terms of didl: and dii: elements. In this chapter, we will discuss higher level issues affecting the overall design of the VDI structure.

5.1 The self-contained VDI

The VDI is the basic information package in the CONVERGENCE system. It is a common, standard container for information about any kind of digital data, including representations of people and Real World Objects. The project's goal is to design the VDI as an extension of the MPEG-21 Digital Item (DI). In MPEG terminology, a DI is a container of information related to a resource. CONVERGENCE is fully compliant with this broad definition. However, CONVERGENCE's emphasis is on the "self-contained" nature of the VDI. It is this feature that allows the physical nodes participating in a CONVERGENCE-enabled communication network to make routing decisions based on the VDI itself (and of the network identifier) without the need of information external to the VDI, effectively building a content-centric network. In the CONVERGENCE network, IP addresses are replaced by resource names. This approach optimizes the use of the network by not necessarily serving data from the location where users might expect to find it or from a single serving point.

CONVERGENCE, uniquely identifies every VDI with a VDI identifier. This is assigned to the VDI when it is first created. In the following paragraphs this still un-specified VDI identifier is generically referred to as VDI_ID.

The flexibility provided by the MPEG-21 standards for Digital Item Identification and the Digital Item Declaration [MP21-2] makes it possible to extend the DI concept into a self-contained VDI capable of supporting the requirements of a content-centric and convergent Internet.

5.2 Updating VDIs and VDI sequences

The design of the CONVERGENCE system takes into account requirements coming from real-world use cases drafted by the members of the CONVERGENCE consortium.

All proposed scenarios require a functionality that is perceived by end-users as an update of the VDI. From a user's perspective, this is very natural: all web sites are inherently dynamic in nature - the home page of a big site will change from hour to hour even though the name of the site remains constant. If the site manager finds an error in a page, she corrects the page without changing its URL.

The user scenarios in D2.1 offer many examples, among which:

- 1) Changing the ownership field in a VDI representing an appliance
- 2) Changing the "user manual" resource in the same VDI
- 3) Adding information about a safety recall in the same VDI
- 6) Modifying a comment to a photo
- 7) Changing rights to a video

VDI updating is a key requirement for the CONVERGENCE system and challenges the design of the VDI as a self-contained package of data. MPEG standards dictate that each DI should be packaged and signed for purposes of trust and that, if a single bit changes, a new DI with a different identifier must be issued. This means that CONVERGENCE should issue new VDIs for each update, never changing a signed VDI. However, it should also give users the perception that they are updating their VDIs.

This **cannot** be implemented by designing a VDI with a changeable part and an unchangeable part. This approach would make it **impossible to sign the entire VDI**. In theory it might be possible for the original author to sign the unchangeable part and other actors to sign the parts that they introduced/changed. However such a scheme would present complex security problems that, even if solvable, would require considerable work and would lead to contrived architectural solutions.

What is needed therefore is a **level of indirection**. In what follows, updating a VDI does not involve any modification of VDIs that have already been published, though it may lead to removal of obsolete VDIs. Different levels of the CONVERGENCE architecture will contain features to make this possible:

- VDIs will contain a sequence identifier (VDI_SEQ_ID), and additional metadata indicating their "ancestor" VDI. This will make it possible to insert new VDIs, and to declare inter-VDI relationships in internal fields. This functionality will be implemented at the CoMid level.
- All VDIs in a sequence will be advertised as CoNet level named-network-resources corresponding to their VDI_IDs. At the same time an updated named-network-resource will point to the latest version corresponding to the VDI_SEQ_ID. No VDIs will ever be altered. At most they will be deleted. Old VDI_IDs will not be reused in new VDIs. This chain of advertisement operations is to be implemented at the CoNet level.

From a logical point of view, these features will require the extra VDI identifier of the sequence (which may be seen as a **sort of a VDI group identifier**) and some additional metadata fields inside the VDI. This will allow search and retrieval of the latest version of an information package or any of the preceding versions. In this way the VDI sequence can be seen as a container for evolving, dynamic information (a sequence evolving along some semantic axis, such as time, space, functionality, etc...), and a stable sequence identification, besides the “regular” VDI_IDs allows referring to the sequence as a whole.

5.2.1 *Semantics of VDI sequences*

Let us consider a VDI X, with a VDI identifier VDI_ID-X.

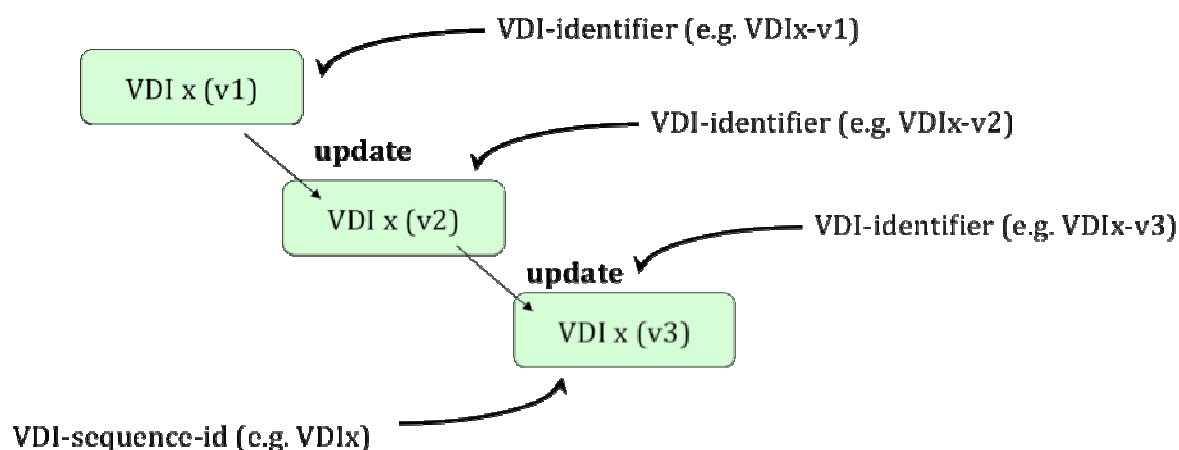
When the VDI is "updated", the system creates a new VDI Y with a new VDI identifier VDI_ID-Y. Hence, it is not possible to refer to the updated VDI with the "old" VDI identifier. Therefore we need a way to refer to the VDI with a different identifier that lets the user access the "most up to date" version of VDI X.

This is the "VDI sequence identifier", VDI_SEQ_ID. The syntax for the identifier is compatible with that of the VDI identifier, and with the MPEG-21 DII standard. The VDI sequence identifier refers to the "most up to date" instance of a group of related VDIs. When the "most up to date" VDI is retrieved, it retains its unique VDI_ID.

In this way:

- each VDI instance has a VDI identifier and a VDI sequence identifier
- end-users, CoMid, and other VDIs can refer to another VDI using the VDI identifier (if they want to refer to a VDI that will never be updated) or a VDI sequence identifier (if they want to refer to the "most up to date" VDI of a group of related VDIs)
- "old" instances of the VDI will still be accessible separately (each one has its own unique VDI identifier!), until they are erased. If it is not necessary to maintain the old copies, the CoMid will delete references to the old copies, and trying to retrieve an old copy will return a "not found" error.

This is illustrated in Figure 2.



VDI-seq-id : identifies the “latest” version of a VDI

VDI-identifier : refers to any specific previous version

Figure 2: VDI updates through VDI sequences

Below, we illustrate the general approach, assuming that identifiers are URLs, and specifically http URLs. Let us assume that we have a VDI X whose VDI identifier is: <http://vdi.org/idXidXidX>. The data for VDI X will include its VDI identifier:

```

VDI X
{
  VDI_ID: http://vdi.org/idXidXidX
  ... other data...
}
  
```

The VDI sequence identifier will be something like this: <http://vdi.org/seqXseqX>

We now assume that VDI X "understands" and includes the concept of VDI sequences. The VDIs will thus be structured like this:

```

VDI X
{
  VDI_ID: http://vdi.org/idXidXidX
  VDI_SEQ_ID: http://vdi.org/seqXseqX
  ...
}
  
```

In this way we can:

- keep the rule that a VDI identifier identifies a given VDI that cannot be changed;

- create a "new version" of the VDI with a new VDI identifier, without changing the former VDI package, which had been already encrypted, signed and validated;
- use the VDI sequence identifier as the "glue" between the different versions of a VDI;
- preserve the functionality of middleware entities that are not aware of the VDI sequence identifier (a VDI_SEQ_ID has the same structure as a VDI_ID);
- maintain security in the resolution of VDI sequence identifiers to VDI instances (the VDI_SEQ_ID is embedded in the VDI, and is therefore covered by the VDI signature)

When an update operation is requested, a new updated VDI Y is created:

```
VDI Y (update of original VDI X)
{
  VDI_ID: http://vdi.org/idYidYidY
  VDI_SEQ_ID: http://vdi.org/seqXseqX
  ...
}
```

5.2.2 VDI sequences CoMid level operations

A VDI update operation is mapped onto a CoMid level VDI insertion (for instance, of a new VDI with a brand new ID, but the same SEQ_ID and a different timestamp). If necessary, the old VDI is removed. When the new VDI's metadata are gossiped [see deliverable D3.1 for details], they are linked to the new VDI_ID. In this way, when the system finds a match for a user search query, the corresponding VDI is given back to the user through its link-back ID.

Each new VDI in the sequence, can in theory be completely different than its preceding VDI. During inter-CoMid "gossiping" interactions, the different CoMid instances running at different peers include the inter-VDI versioning identifier SEQ_ID in the information that they "gossip about". As a result, information propagates through the network and can be retrieved during the processing of user requests.

In this situation, an application can request from the CoMid:

1. A specific VDI, defined by its VDI_ID.
2. The latest version (or update) of a specific VDI, identified by its VDI_ID.

In situation (2) the CoMid uses the SEQ_ID to identify the top VDI and retrieve it. This is a relatively easy operation, despite CoMid's distributed nature.

5.2.3 VDI sequences and CoNet level operations

To implement the above-mentioned functionality, each VDI update operation at the CoNet level triggers the following operations:

- the update VDI is advertised in the CoNet with a NID that is the same as the SEQ_ID, in this way effectively replacing the old content

- the update VDI is **also** advertised in CoNet with the NID that is the same as the VDI_ID. This makes it possible to fetch VDIs either through their VDI_IDs or their SEQ_IDs.

At this point, two VDIs with the same SEQ_ID and different VDI_IDs have been injected into the network. When a request is made to <http://vdi.org/seqXseqX>, the system fetches VDI Y, the most recent advertised VDI with that identifier.

In this way the CoNet level is “aware” of all of the update operations and keeps a reference to the updated version of a VDI.

5.3 VDI Relationships

VDI relationships are logical links that connect two individual VDIs (or, more precisely, connect the resources represented by two individual VDIs), from the point of view of human consumers. As such, they describe the “meaning” of a *referencing* VDI with respect to a *referenced* VDI. In this relationship, the referencing VDI may be considered the active object in the relationship and the referenced VDI the passive object. Together with the relationship itself, the two VDIs form a semantic triple. For example, if *VDI A* “carries” a user-made video and a user of *VDI A* expresses his appreciation for another movie “contained” in *VDI B*, then *VDI A* (plays the active role and) is a comment to *VDI B* (which plays the passive role). That is, *VDI A* maintains a relationship of “*commentation*” with *VDI B*. If a user publishes a book “within” a *VDI C*, and later on she realizes that it contains errors, she may then publish a *VDI D* containing the book’s errata. In this situation, *VDI D* maintains a relationship of *correction* with *VDI C*.

5.3.1 Relationships’ Usefulness

Several literature studies including [MPOWL, LANLL, MVCOM] have proposed that Digital Item relationships should be embedded in DIs via explicit declarations within a DIDL document. This is a powerful yet lightweight way of establishing a rich semantic fabric of VDIs. Explicit, user declared links between VDIs facilitate “interpretation” of VDIs.

In terms of CONVERGENCE use cases, this strategy makes it easier to specify that some VDIs are:

- comments to other VDIs;
- responses to other VDIs;
- enhancements of other VDIs;
- additions to other VDIs (example: a VDI “containing” an X-Ray image of Mary’s lungs is an addition to the VDI that “contains” Mary’s overall medical record);
- etc.

The introduction of explicit user-declared relationships between VDIs enable users to explicitly search for comments, corrections, evaluations, related to a specific VDI. This

enables CONVERGENCE to offer a richer content searching service. This strategy ensures that the retrieval of inter-related VDIs does not depend on the system somehow identifying relationships though the semantic tags used to qualify the content (a complex and inevitably imprecise task).

Consider an example. If VDI A “contains” a movie and VDI B “contains” a review and there is no explicit declaration of the inter-VDIs relationship, the only way for CONVERGENCE to determine that the latter is a review of the former would be by examining the semantic tags of both VDIs. These would have to explicitly indicate that the resource represented by VDI B is logically a review of the content represented by VDI A. Otherwise the only way to automatically establish the relationship would be through complicated and imprecise inferential mechanisms.

In brief, the lack of formal, well-defined mechanisms to specify relations amongst VDIs, would make it difficult to provide CONVERGENCE users with effective search services. However, if VDI B carried metadata, explicitly declaring that it maintains a relationship of “reviewing” with VDI A, CONVERGENCE could offer a service allowing users to explicitly search for reviews of a specific VDI.

5.3.2 Relationships’ Declaration and Types

VDI relationships are declared within specifically defined tags, contained inside VDIs. Each relationship declaration must thus specify the type of relationship in question and the identity (the VDI identifier) of the *referenced* VDI. A VDI may declare an arbitrary number of relationships to an arbitrary number of other VDIs (as exemplified in Figure 3).

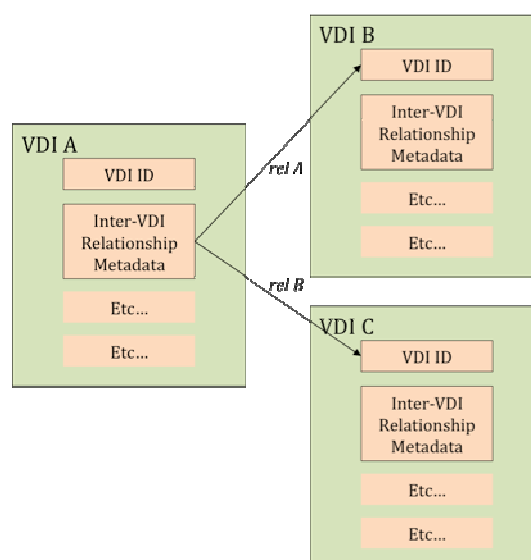


Figure 3: Graphical Depiction of Inter-VDI Relationships

The types of declarable relationships are derived from a core set of global relationships defined by CONVERGENCE. This can be extended by application-dependent relationships, which are managed and employed on an application-dependent basis, as explained in the

following chapter. Indicatively, the core set of VDI relationships should be rich enough for users to express:

- correction;
- responding;
 - commenting;
 - evaluating;
- extension;
 - enrichment;
 - addition;

These relationships are defined in a standard ontology, which may then be built upon, by extension into application-specific ontologies.

6 Structure of the VDI

As we have seen, the flexibility and expandability of the Digital Item structure is guaranteed by `didl:Descriptor` “overloading”. The MPEG-21 standard exploits the same mechanisms when it introduces a `dii:Identifier` element inside the `didl:Descriptor`.

6.1 Conformity with the MPEG standard

The CONVERGENCE VDI is a fully compliant and fully conformant MPEG-21 Digital Item DIDL document, with additional **required** fields to capture:

- A unique VDI identifier: a mandatory unique identifier at the root level of the VDI document
- Relationships with other VDIs, including versioning semantics
- Searchable metadata

The following paragraphs will describe the details of each of these additional fields.

Formally, a VDI is defined as a DI whose root element (either a `didl:Container` or a `didl:Item`) embeds mandatory `didl:Descriptors`, as represented in Figure 4.

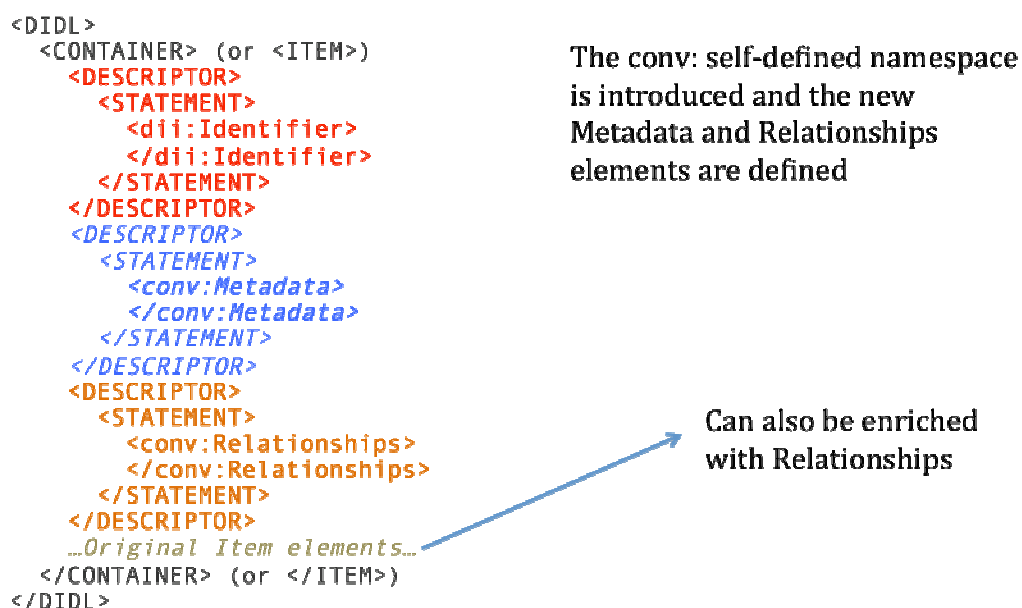


Figure 4: Conceptual representation of VDI structure

Conceptually, the VDI is thus an enrichment of the root element of a DI with a set of `didl:Descriptor` elements, which are used to wrap the mandatory VDI identifier and the relationships between this VDI and other VDIs (capturing, among other things, the versioning

information) and the semantic metadata of this VDI. The CONVERGENCE self-defined conv: namespace identifies and isolates the Relationships and Metadata blocks.

Conceptually, the VDI creation process could be seen as a:

- 1) definition of a “regular” DI
- 2) embedding identification information and conv: specific information in the root element.

Thus an existing DI can be turned into a VDI **without** opening or un-wrapping it, if this is not necessary for other reasons. All that is needed is to add didl:Descriptor fields to the root element to express the necessary semantic relationships and to identify it uniquely in the CONVERGENCE system. No other manipulation of the existing elements is required. **Unfolding** of the nested structure is necessary only if nested elements are also to be the subject of semantic relationships. Otherwise they can be left untouched. Optionally, the nested Items composing the VDI can also be enriched with Relationships so that they, in turn, contain didl:Descriptors expressing semantic relationships with other Items (either internal or external).

As explained in the previous chapters, items in the MPEG-21 DIDL can contain sub-Items. This **implies** some kind of “nesting” or “composition” relationship between the sub-Items and the embedding Item. Since we are devising means to **explicitly** express semantic relationships between Items, via the overloading of didl:Descriptors, it is not possible to define semantic relationships for **nested Items below the root level**. An equivalent effect can be achieved by defining explicit semantic relationships between Items of the same VDI. This technique can be very useful for representing RWOs, since the full range of precisely defined and named relationships can be instantiated between the parts that compose the object, instead of relying on generic nesting only.

6.2 VDI Identification

As previously mentioned, the MPEG-21 standard Digital Item Identification prescribes the dii:Identifier tag as part of didl:Descriptor statements. This implies that there is no unique identifier for the whole DI: wherever a didl:Descriptor can fit, there is also room for an identifier. In fact, each Container, each Item composing the DI, and even each Component can contain one (or more!) dii:Identifier elements, embedded within their respective didl:Descriptor elements.

This implies that CONVERGENCE needs to define the syntax of its unique VDI identifier and its position in the DI structure. Since the CoMid needs to identify VDIs, while the CoNet needs to identify named resources, we need identifiers at two different levels, VDI-identifiers and VDI-sequence-identifiers are CoMid level identifiers. Inside the VDI they are captured as **dii:Identifier** elements, related to the CoNet identifiers through mapping functions. These different levels of identifiers are depicted in Figure 5.

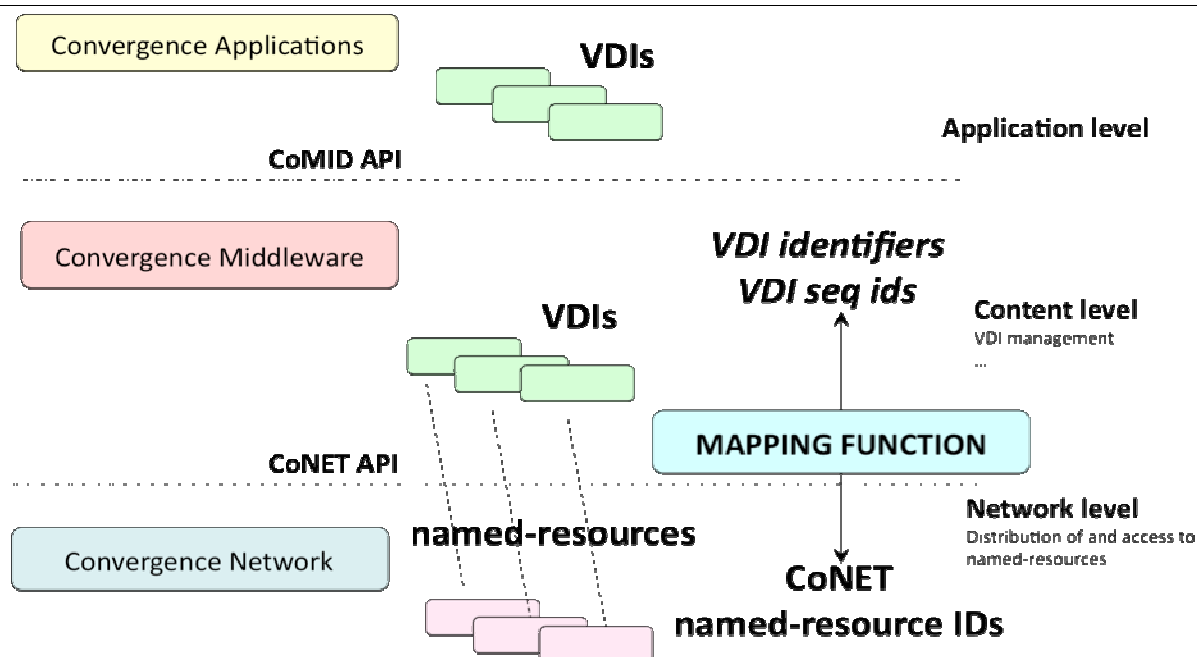


Figure 5: Various levels of identifiers

A single entity at the middleware level assigns unique VDI-sequence-identifiers and VDI-identifiers. This CoMid Content Identification Service can use CoNet resource identifiers as a base and map them to CoMid level identifiers.

Example:

Bob-Smith/publication23.pdf-XXYYZZ

is a CoNet-level id. At the CoMid level, it could become:

urn:mpegRA:mpeg21:conv:Bob-Smith/publication23.pdf-XXYYZZ

On the other hand, if the VDIs remain in the user's local device or network, they are assigned a **local address space**.

6.3 Semantic conv: Relationships inside VDIs

Semantic relationships are defined via a custom overloading of the didl:Descriptor. This is accomplished by introducing a self-defined CONVERGENCE conv: XML namespace, containing XML statements about VDIs. A preliminary analysis of possible solutions for different syntaxes representing relationships suggests that RDF/XML could be a viable choice for expressing these relationship blocks. In this case, conv:Relationships elements would be snippets expressed in RDF [RDF] language, based on a vocabulary defined in OWL [OWL]. To date this choice has not been finalized. The final decision is tightly connected to the outcome of CONVERGENCE's standardization proposals to MPEG-21. This issue is further discussed in an annex to this deliverable.

Relationships are defined at two levels:

- At the VDI level: global information for the VDI as a whole
- At the Item level: information about parts of the VDI

Both levels have a model that captures the distinction between **Context Information** and **Representation Information**, as defined in the OAIS Open Archival Information System [OAIS] and similar approaches:

- Context Information: how the object relates to its environment and other objects
- Representation Information: describes the internal structure and hierarchy of the object.

For example, the complex details of the physical structure of a RWO can be modelled by introducing semantic relationships between the Items composing the VDI that represents the whole object.

6.3.1 Inter-VDI relationships vs. inter-Item relationships

Relationships between VDIs are expressed in terms of the **VDI-level** Ontology. This defines the attributes of a VDI as a whole, as well as core systems concepts that are vital for the functioning of the CONVERGENCE framework. Nevertheless it is a dynamic ontology. Where useful, it is possible to add application-specific relationships to describe the domain specific behaviour of VDIs used by a CONVERGENCE-enabled Application. User searches and queries are expressed in terms of restrictions formulated over the terminology of the VDI-level ontology and its extensions.

Inter-Item relationships are expressed in terms of Application ontologies. These may take the form of a root **Item-level** (Upper) Ontology such as the ontology introduced in paragraph 6.3.3. Alternatively it is possible to introduce completely custom taxonomies to capture the internal structure of specific VDIs. Applications/Domain ontologies at the Item level also define terminology custom for a specific application. However these ontologies are conceptually associated with a single VDI, and users can only fully exploit them when they acquire the VDI and browse the individual Items in the VDI.

6.3.2 VDI-level relationships

The taxonomy of possible relationships between VDIs and their meaning is encoded in the VDI-level ontology. The conceptual design process is depicted in Figure 6.

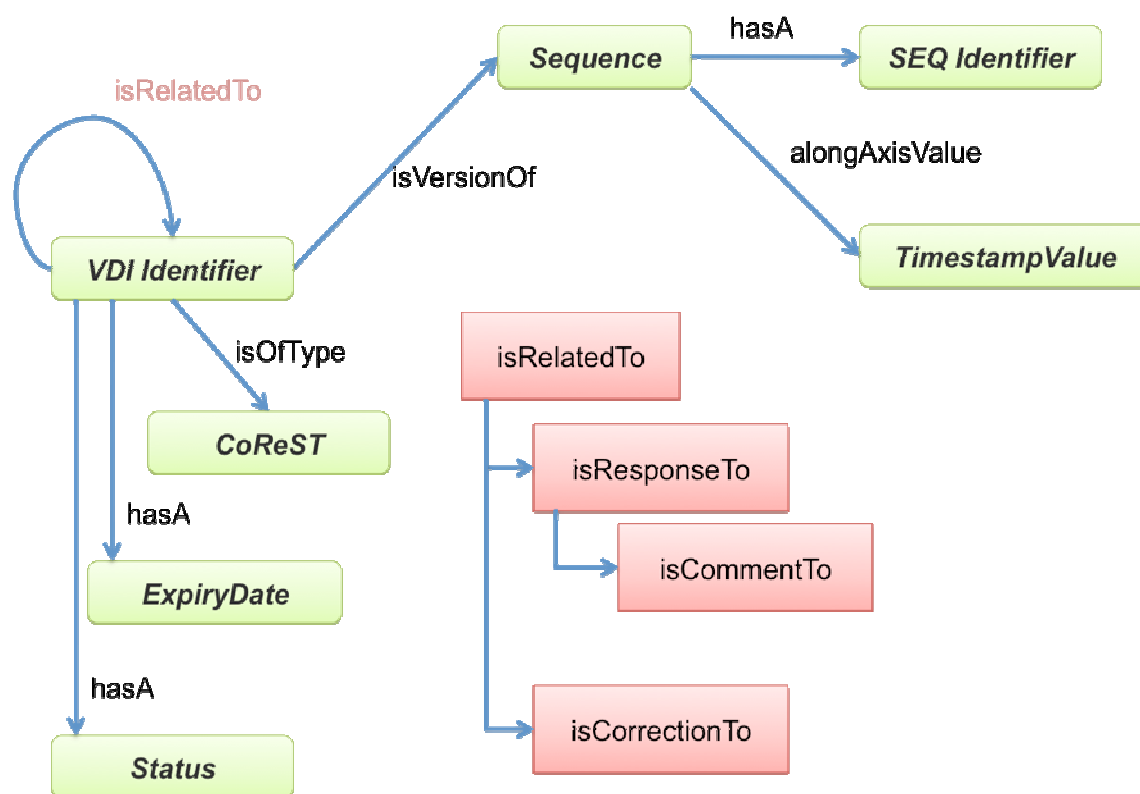


Figure 6: VDI-level ontology model

The green-coloured (boldface) *italics* classes (centred on the VDI Identifier class) capture the Context Information for a VDI as a whole. Each VDI is assigned a CONVERGENCE Resource Semantic Type (see D3.1). The explicit ExpiryDate class is used to enforce “not found” behaviour when searching for expired objects at the Content Level of the CONVERGENCE architecture, and for progressive garbage collecting at the Network Level.

Each VDI has a unique identifier, and is assigned to a Sequence, which represents a ternary relationship between a specific VDI, a global identifier for the whole sequence and a “semantic axis” along which the Sequence has to be interpreted. The default approach is to unfold the versioning along the time axis, so that each update to the sequence gets a progressive timestamp. Nevertheless, more sophisticated approaches can also be envisioned. For instance successive versions of a VDI could be interpreted as “geographic location updates” along a spatial axis or as updates in functionality along a functionality axis.

The red-coloured areas (normal text) (hierarchy under isRelatedTo property) capture the Representation Information for a VDI as a whole, specializing the isRelatedTo generic bonding between two distinct VDIs. This captures generic relationships attached to the VDI package, and is useful for all kinds of VDIs. Important relationships include thisVDI “isSimilarTo” “isPartOf” “isConnectedTo” “extends” “embeds” “replaces” “comments” “follows” “precedes” “derivesFrom” “leadsTo” anotherVDI.

These are generic relationships that every human can use, at VDI creation time, by giving them an **intuitive** semantic meaning that can differ between domains without changing or

further specifying the base ontology. They map onto logical connections between the objects represented by the VDI package as a whole, rather than the multiple individual Resources within.

They facilitate the expression of assertions such as:

- This VDI is an enrichment to another VD
- This VDI is a comment to another VDI
- This VDI is an addition to another VDI

VDI-level relationships make it possible to search for comments or corrections to a specific VDI. They can be searched in combination with data such as sequence version, date of creation, geographic place of creation of the VDI, author and the such. As mentioned before, they are freely extendable by means of domain and application-specific terminology.

6.3.3 Item-level relationships

Relationships between VDIs are expressed in terms of the VDI-level ontology, as detailed in the previous section. Inter-Item relationships are expressed in terms of Application ontologies, which define a terminology customized for a specific application domain.

The taxonomy of possible relationships between parts of a VDI, represented by Items specifications contained within a VDI document, is thus encoded in Application ontologies that specialize an Item-level ontology such as the one depicted in Figure 7.

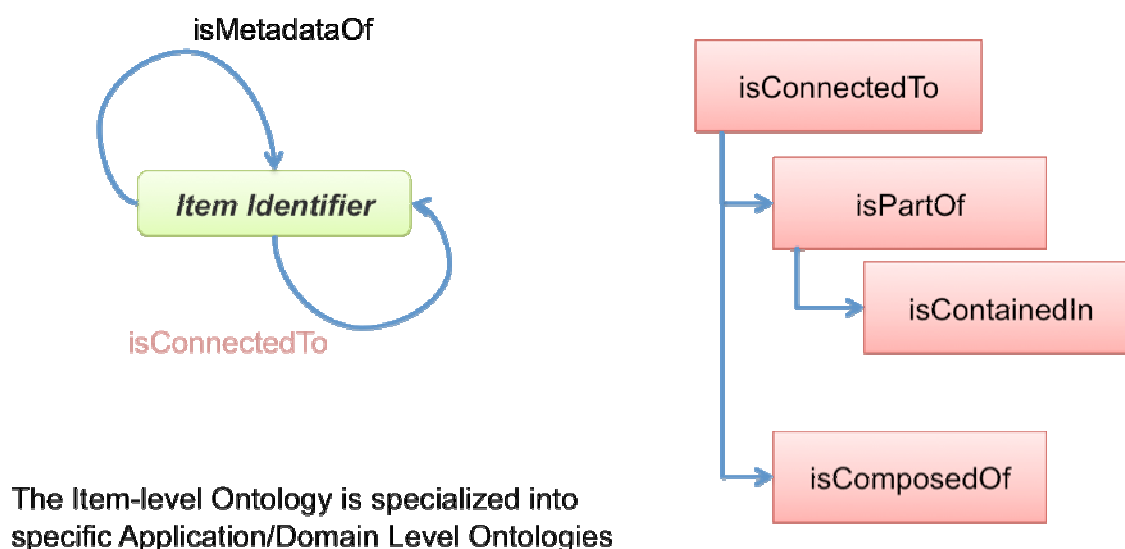


Figure 7: Item-level ontology model

In this picture, the green-coloured (boldface italics) class Item Identifier and the red coloured (normal text) hierarchy of properties under isConnectedTo, distinguish between Context and Representation information, again. Since Item-level relationships apply to single Items within the VDI, each didl:Item must be uniquely identified.

The simplest approach is to use the VDI `dii:Identifier` as a base reference (`xml:base`) and then use standard XML fragment identification (XPointer specification [XPOIN]) for the Items. This approach has the advantage of being independent of other possible identifiers of the Item already present inside its definition. Since existing `didl:Item` elements are identified by their XPointer semantics within the VDI document, no explicit additional identifiers are necessary.

Metadata require human effort to be created, organized and referenced and are often as important as data. In many cases, they are already available as external Resources. In the model proposed here, any Item can represent metadata (i.e. context descriptive information) about other Items. In other words, metadata receives the same treatment as data.

Some relationships at the VDI-level and at the Item-level may have the same name, such as “`isPartOf`”. Nevertheless they pertain to different namespaces and different ontology models and as such are distinguishable.

6.4 Processing Relationships into `conv:Metadata` blocks

Reading Metadata blocks by the CoMid needs to be efficient: when a VDI arrives at a CoMid instance it is not feasible to run a SPARQL query through the whole VDI to collect its Metadata. Metadata is thus pre-packaged.

At VDI packaging time, the Packaging Engine scans the VDI, collects and interprets Descriptor for all Items and normalizes the information in `conv:Metadata` elements.

These `conv:Metadata` elements will most likely be designed as RDF/XML snippets, but could also be custom-designed for efficiency. This issue will be further investigated in later stages of our work. In the `conv:Metadata` block they will be packed in a “normalized” form allowing efficient parsing and searching. Normalization operations include

- Creating an expanded list of triples
- Resolving References
- Making literal values explicit

The metadata contained in the VDI `conv:Metadata` block is information about the VDI as a whole, and not just about the individual Items in the original DI. The metadata is extracted during VDI creation:

- from the `conv:Relationships` elements of all Items at the Item-level
- and also from `conv:Relationships` at VDI-level (the root element)

The `conv:Metadata` syntax will depend on the technical solutions adopted for CoMid engines and will be specified in future deliverables.

At publishing time, the CoMid parses the VDI, extracts the `conv:Metadata` information, (which now contains the semantic characterization of the VDI and the characterization of the relationships in which the VDI is involved), and performs the appropriate dissemination and storage of the information. Users will be able to search the data both for specific concepts (keyword search) and for VDIs which maintain specific relationships with other VDIs.

When a user searches for VDIs by keyword, the local CoMid sends the query to the peer CoMid instance responsible for the concept in question (that is, the CoMid peer that keeps the list of all the VDIs involved as active or passive part in that concept), which then sends the appropriate response. When, a user searches for VDIs in a given relationship to another VDI (for instance, all VDIs that *comment* on VDI XYZ), CoMid will search through the information describing relationships of the type in question, looking for *commenting* relationships which link to VDI XYZ. To perform such a search, CoMid directs the query to the CoMid instance responsible for the relationship concept *comment*, that is, the CoMid peer that keeps the list of all VDIs involved in a relationship of that specific type. The peer then provides the appropriate answer. This way, the system handles the two types of search in the same way, that is, it directs the query to an appropriate CoMid instance.

6.5 Embedding Resources in VDIs

The MPEG-21 standard describes a number of ways of embedding Resources in Digital Items. Multiple Components in the same Item can be used to offer the same Resource in different formats. For example two Components in the same Item could hold song.mp3 and song.wav, that is the same song in compressed and raw formats. The same Component can contain multiple resources. This technique can be used when redundancy or load balancing is desired. By including a link to an external server holding a copy of the Resource, it is possible to offer the same Resource twice in the same Component, once by value and once by reference.

Although syntactically both the `didl:Component` and the `didl:Resource` elements can be tagged with `didl:Descriptors`, these Descriptors may not be used for embedding semantic relationships at the level of the Component or Resource.

6.6 Granularity of the VDI

By granularity we mean the degree to which information is packaged within the same VDI and the degree to which it is distributed across several distinct VDIs. Choosing the optimal granularity is a process depending on the application and the user. As defined in CONVERGENCE, the VDI is the atomic searchable entity of the CONVERGENCE network. All Items within the same VDI are classified under the same set of metadata within the CoMid and are distributed together as a self-contained information package.

6.6.1 Implications of VDI Removal

In most Inter-VDI relationships, the removal of one of the VDIs in the relationship (specifically the VDI that occupies the passive role in the relationship), does not present problems. For instance, if *VDI B comments VDI A* and the latter is removed, the former will become a dangling comment that “points” to a non-existent VDI. In most cases this will not be a problem (just like, in most cases, a broken link does not disable a web site). There are

some cases, however, where a broken link could be problematic. This can occur when Inter-VDI relationships form logical sequences, where the knowledge of the entire chain is necessary for the proper operation of the system. Two typical examples are *versioning* and *updating* Inter-VDI relationships (*VDI C* updates *VDI B* which updates *VDI A*). In both cases, removal of one of the VDIs isolates the two resulting halves. This is why VDI sequences are managed by means of a common Sequence Identifier which is part of the context information of all VDIs.

The explicit declaration of Inter-VDI relationships thus helps the implementation of digital forgetting, making it possible to delete VDIs without impeding the regular operation of the system. It also facilitates digital forensics. Even if a specific VDI is removed, all the other VDIs that commented and criticized it will still be available. This means that “traces” of the VDI will remain.

6.7 Trusting VDIs

Each VDI may also include a License. The license to a VDI will state who can modify it and how. The license could state, for example, that only the authenticated author can modify the VDI, that other authors can add resources to the VDI without deleting the original resources or that other authors can freely replace resources in the VDI. In the first case, the author will sign any updated version. In the second case, the new author will add (and perhaps sign) a new resource, in the third case the author will replace the old resource with a new, signed resource. In each of these cases, the middleware responsible for publishing the updated VDI will recalculate a digital signature on the whole package of bits, so that the VDI can be trusted, based on the author of the change.

Where the goal is to ensure non-repudiation, it is enough to state in the license that the VDI cannot be modified by anyone (including the author). These certification and authentication mechanisms are a powerful guarantee that the VDI has not been tampered with after publication. CONVERGENCE will also support anonymous communication. This is essential, if we want to use VDIs for free discussion on controversial topics or in countries where free discussion is unpopular with the government.

6.7.1 Relationships and Rights

Authors of VDIs are **not** free to declare any kind of relationships to other VDIs. For instance, if Mary is not the owner of *VDI A*, she is not allowed to insert a *VDI B*, with *errata* for *VDI A*. On the other hand, Mary is perfectly free to insert a *VDI C*, which she declares as a *comment* to *VDI A*. Before accepting the VDI the system has to analyze and validate its VDI-level Relationship tags. If a user searches for and retrieves *VDI C*, she only receives Mary’s comment, not *VDI A* itself. Nevertheless when browsing *VDI C* she will be able to reach into *VDI A*, if this is allowed by the rights expressed by the VDI.

Rights management and rights expression will be the topic of forthcoming deliverables.

7 Mapping VDIs to Scenarios

This chapter will propose strategies and guidelines allowing the partners responsible for use cases to map VDIs onto the conceptual models described in the previous sections. These guidelines will make it easier to define the boundaries of VDIs and the distinction between a VDI and its constituents. This will facilitate the adoption of the VDI concept in real-world scenarios and make it easier to design the CONVERGENCE middleware, which obviously has to be independent of the content of individual VDIs.

7.1 Lecture Podcast, an example mapping

7.1.1 *Description of the model*

The main entities of this scenario are: the lecture podcast, the podcast application, and comments.

Lecture Podcast

A lecture podcast is composed of one or more podcast episodes. Each episode is composed of synchronized slides and videos, which are segmented into chapters. Chapters of videos are associated with timestamps. Slides are composed of one or more slide(s). A slide contains content and references to other content.

The podcast can be viewed in a simple video application (e.g. Quicktime) or combined with a web-based podcast application, which supports students when they revise the lectures.

Podcast Application / Comments

The podcast application offers an individual and a collaborative mode.

a) Individual Mode

Based on the content of a slide, the individual mode automatically suggests related content (e.g. further readings) to students. It also allows students to comment content for their individual learning. A comment always refers to a certain timestamp and has a specific position on a slide.

b) Collaborative Mode

In collaborative mode, comments may be made (semi-) publicly, in order to start a discussion. Other users have the possibility to reply to comments or create new ones. Comments may be published anonymously or with a username.

Figure 8 shows these relationships.



Figure 8: Semantic model of the Lecture Podcast Application domain

7.1.2 Example mapping procedure

Alina has some OpenOffice slides of her lesson inside a slides.pdf file. Assume she wants to make this file available to the public by publishing it inside the network and making it searchable. The first thing she does is to create a standard MPEG-21 Digital Item out of the file.

```
<ITEM>
  <!-- identifier of the Item, not of the VDI. -->
  <It can be anything, like file path -->
  <IDENTIFIER>urn:foo:bar:alinaHD/slides.pdf</IDENTIFIER>
  <CONTENT>
    <RESOURCE encoding="base64" mimeType="application/pdf">
      ZpZWxkPgOK9yZWQgSjJlj5jMTk5My48L3iVVR6LTgiPz4NCjxb2xs...
    </RESOURCE>
  </CONTENT>
</ITEM>
```

The whole file of the slides is embedded into the Item as an uuencoded mime-type.

Afterwards, the Slides DI is turned into a proper VDI, either by embedding it into an outer root Container element, or by adding CONVERGENCE identifiers (Sequence ID and VDI ID) and metadata (generated automatically or defined by the user).

The first case produces the VDI shown below.

```
<ITEM>
  <SEQ_ID> sequence1 </SEQ_ID>
  <VDI_ID> vdi7 </VDI_ID>
  <METADATA> slides; lessons; human-computer interfaces </METADATA>
  <!-- identifier of the Item, not of the VDI. -->
  <It can be anything, like file path -->
  <IDENTIFIER>urn:foo:bar:alinaHD/slides.pdf</IDENTIFIER>
  <CONTENT>
    <RESOURCE encoding="base64" mimeType="application/pdf">
      ZpZWxkPgOK9yZWQgSjJlj5jMTk5My48L3iVVR6LTgiPz4NCjxb2xs...
    </RESOURCE>
  </CONTENT>
</ITEM>
```

In the second case, the resulting VDI is:

```
<CONTAINER>
  <SEQ_ID> sequence1 </SEQ_ID>
  <VDI_ID> vdi1 </VDI_ID>
  <METADATA> slides; lessons; human-computer interfaces </METADATA>
  <ITEM>
    ... Digital Item with slides file here ...
  </ITEM>
</CONTAINER>
```

In what follows we will assume that the VDI is in the second form.

The file with the slides is now ready to be packaged inside a well-formed CONVERGENCE Slides VDI. Some days later Alina is asked to provide a more complete podcast episode, that includes the slides and some video footage of her talk. She thus decides to assemble some video and the slides inside a Podcast Episode VDI and distribute it.

She now faces two alternatives:

A) Create a Podcast Episode VDI, which embeds the original digital item of the slides files and a new digital item for the video. This will be an entirely new packaging of the same content/resources, completely independent of the other Slides VDI, which just contains the slides. In this case the Podcast Episode VDI would look like:

```
<CONTAINER>
  <SEQ_ID> sequence2 </SEQ_ID>
  <VDI_ID> vdi2 </VDI_ID>
  <METADATA>
    podcasts; multimedia lessons; human-computer interfaces
  </METADATA>
  <ITEM>
    ... video Digital Item here ...
  </ITEM>
  <ITEM>
    ... Digital Item with slides file here ...
  </ITEM>
</CONTAINER>
```

B) Create a Podcast Episode VDI that just contains the video and put it in a semantic relationship that links this Podcast Episode VDI with the former Slides VDI. In this case, the Podcast Episode VDI would look like this.

```
<CONTAINER>
  <SEQ_ID> sequence2 </SEQ_ID>
  <VDI_ID> vdi2 </VDI_ID>
  <METADATA>
    podcasts; multimedia lessons; human-computer interfaces
  </METADATA>
  <RELATIONSHIPS>
    <TYPE> isComposedOfSlides </TYPE>
    <TARGET_VDI> sequence1 </TARGET_VDI>
  </RELATIONSHIPS>
  <ITEM>
    ... video Digital Item here ...
  </ITEM>
</CONTAINER>
```

This second approach is more flexible. We now have two independent but interlinked information packages that can be searched for and retrieved independently. This approach encourages the user to distribute the various pieces of information across several distinct containers and to maintain them in a linked browsable format. However, the balance between what to keep together and what to spread into various VDIs is domain- and constraint-dependent.

- In case A) if the original Slides VDI is updated, this is not reflected in the Podcast Episode VDI.
- However, in case B) the whole Podcast Episode (meaning Podcast Episode VDI + Slides VDI) is no longer distributed as a bundle containing the two VDIs. Browsing all the VDIs in the Podcast domain requires CoNet. Using CoNet, the user can start from the root, i.e. the Podcast itself, and then retrieve the Slides. If the slides are updated, another Slides VDI is generated, with a different VDI_ID, but the same SEQ_ID. Since the Podcast Episode VDI uses the SEQ_ID as a target, it will point to the latest version by default. Conversely, the Podcast Episode VDI might have any of the specific VDI_IDs of the sequence as a target. This means that the episode is bound to a specific version of the slides. In this latter case, it is still possible to retrieve the latest version, since all VDIs of the sequence know what the latest version is.

In the Podcast scenario depicted here, the use of the Slides VDI is motivated by the idea of providing an independent package of information that does not depend on the existence of the Podcast.

The scenario also contains references to several other VDIs.

- The Podcast VDI representing a group of Podcast Episodes.

- A Discussion VDI, containing Comments: here again the VDI designer can choose between making a Comment Item part of a Discussion VDI and creating a standalone Comment VDI. This latter solution might be acceptable. However, the model suggests that comments should be versioned. This can only be achieved by taking the Comment DIs out of the Discussion DI and linking the two entities together as standalone VDIs.
- The User, represented by a User VDI.
- Other sources of information, such as a VDI representing a Book. This possibility shows how Items can have relationships with external Items, as in the case of the Item named Content (it is part of a Slide Item, which in turn is part of a Chapter Item of Slides VDI). This item has a relationship of type *providesKeywordsFor* towards an external Item named Similar Content of an external Book VDI.
- The Podcast Application itself, coarsely represented by yet another VDI. Of course it is difficult to fully represent an articulated object such as the Application itself as a VDI, since it has states, and even the *didl:Conditions* are not enough to capture such complexity. However, it is unlikely that such a representation will be needed.

In the following Figure 9, the same semantic domain as the previous Figure is shown, but a possible mapping of the domain into VDIs is depicted, summarizing what has been discussed up to now. VDI-level relationships are enclosed in (blue) circles. All other relationships are Item-level relationships. VDI boundaries are shown with superimposed (yellow) boxes.

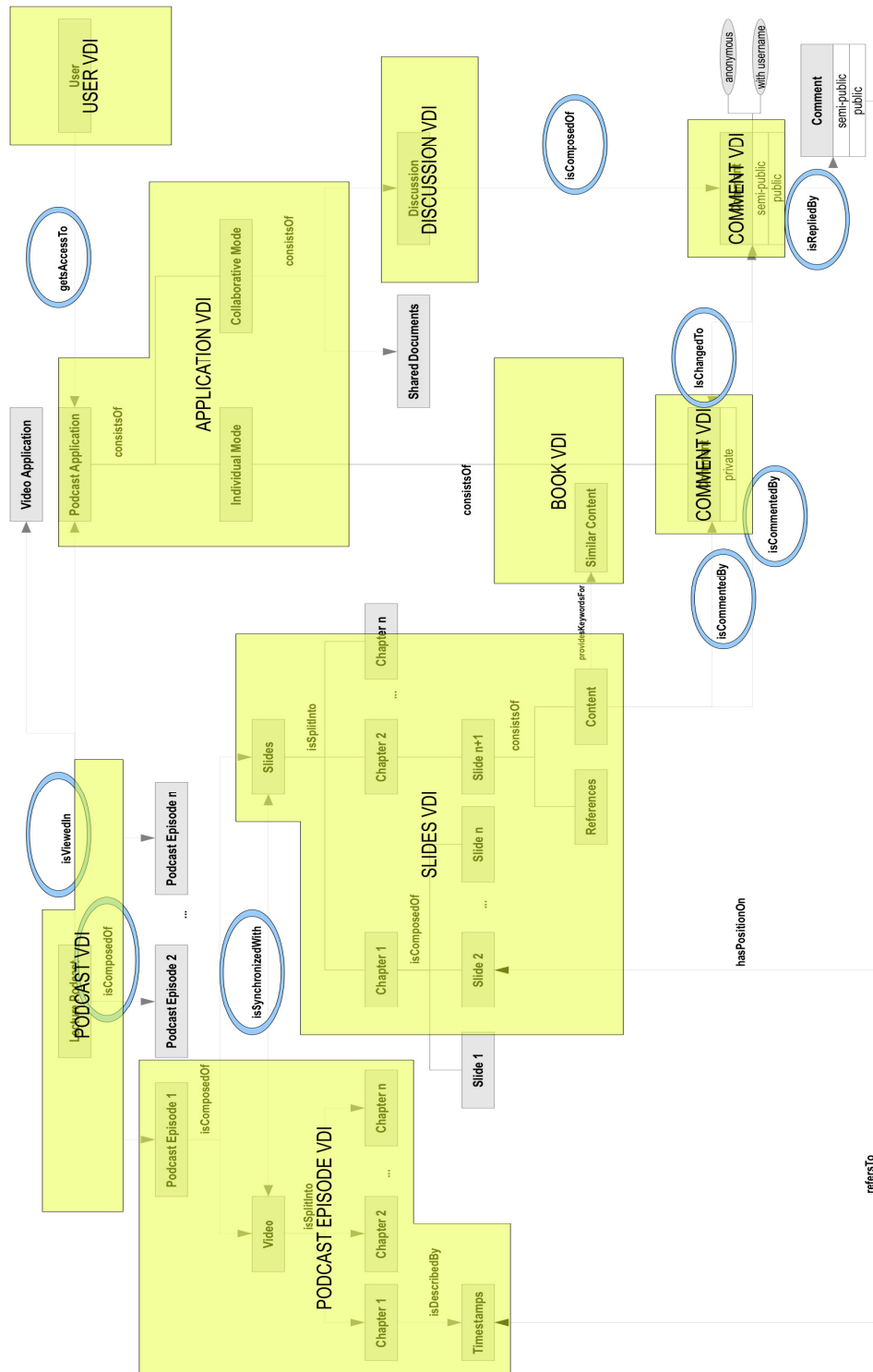


Figure 9: The Lecture Podcast domain mapped to VDIs

7.1.3 Application design and VDI bundles

The Podcast Application is conceived so that it offers students the possibility of installing a lecture podcast to their local device (in this case a connection to the CONVERGENCE Network is not strictly needed; e.g. a USB stick can be used), or of studying the podcast in a Web-based learning community. In this second mode, it is natural to follow approach B) (using the SEQ_ID to link to the slides VDI).

In the first mode, however, it may be more natural to follow approach A), creating an entirely new package that contains everything. This would make it easier to move the podcast for offline fruition. The drawback would be that students wouldn't be working on the latest material, since links to "external" slide VDIs would be missing.

The right choice is to increase the VDI granularity and distribute the information regarding the most dynamic part of this domain, i.e. the Slides, into a separate VDI. All the VDIs could then be distributed e.g. on the same USB stick, while maintaining their status as separate VDIs.

What makes this possible is the fact that semantic links between VDIs are explicitly stated in the metadata block of each VDI. This means that the Browser application can scan for a list of locally available VDIs and aggregate them in the correct order.

Obviously, the students will get any update as soon as the Application goes online. If a student stays permanently offline once she has installed the podcast, she will begin by browsing the root Podcast VDI. When she accesses the Slides, she will receive a message from the Application level informing her she is offline and asking whether she wishes to browse for the files only locally. This means however that the Application will require appropriate mechanisms to automatically scan for referenced VDIs. Whenever the Application is connected, it acquires updates and allows the user to use them when she is offline. This is analogous to having a local cache on a web browser.

As a general design paradigm, the Application level should offer the possibility of searching for VDIs, downloading the results to a local cache and keeping them browsable, in a coherent overall view which includes all semantically related VDIs, even when the user is offline. However, this automatic behaviour cannot be a feature of the CONVERGENCE middleware. Given that referenced VDIs can reference other VDIs it is not possible for the middleware to fetch all related VDIs. It is up to the application designer to code the application so that it can follow the semantic relationships between VDIs and assist the user in browsing the web of VDIs that constitutes the semantic domain of the application. This is normal since it is the Application designer who decides the granularity of the VDIs in the specific domain, based on her knowledge of what parts are more dynamic and need frequent updates and what pieces of information can be safely packaged together with the root VDI.

7.2 Example syntax of VDI relationships

In an annex to this document, we suggest a possible format for the VDI relationship. This annex has been produced in order to formally initiate a process of interaction with MPEG, and specifically MPEG-21. The goal is to promote the extension of the DID towards support of Inter-DI semantic relationships.

8 References and relevant literature

- [IELOM] IEEE LOM (2000). LOM final draft, <http://ltsc.ieee.org/wg12/20020612-Final-LOM-Draft.html>
- [HYTEC] Reyes, E., Saleh, I. (2004). HyperTectol, an assistant and authoring tool for using multimedia in Learning Objects creation. Proceedings of the 5th. International Conference on Information Technology Based Higher Education and Training. Istanbul: IEEE
- [MP21-1] ISO/IEC 21000-1, MPEG-21 Part 1 - Vision, Technologies and Strategy
- [MP21-2] ISO/IEC 21000-2, MPEG-21 Part 2 - Digital Item Declaration
- [MP21-3] ISO/IEC 21000-3, MPEG-21 Part 3 - Digital Item Identification
- [MP21-4] ISO/IEC 21000-4, MPEG-21 Part 4 – Intellectual Property Management and Protection (IPMP)
- [MP21-5] ISO/IEC 21000-5, MPEG-21 Part 5 – Rights Expression Language (REL)
- [SPRQL] SPARQL Query Language for RDF, W3C Recommendation, January 2008, available at: <http://www.w3.org/TR/rdf-sparql-query/>
- [XPOIN] XML Pointer Language (XPointer) Version 1.0, January 2001, <http://www.w3.org/TR/xptr>
- [RDF] Resource Description Framework (RDF), <http://www.w3.org/RDF/>
- [OWL] OWL Web Ontology Language ☐ Reference, February 2004, <http://www.w3.org/TR/owl-ref/>
- [OAIS] Consultative Committee for Space Data Systems, CCSDS 650.0-B-1 BLUE BOOK, Reference Model for an Open Archival Information System (OAIS), January 2002
- [MPOWL] Chrisa Tsinaraki, Panagiotis Polydoros, and Stavros Christodoulakis; Interoperability Support between MPEG-7/21 and OWL in DS-MIRF
- [LANLL] Using MPEG-21 DIDL to Represent Complex Digital Objects in the Los Alamos National Laboratory Digital Library; http://www.dlib.org/dlib/november03/bekaert/11bekaert.html#Appendix_A
- [MVCOM] Victor Rodriguez-Doncel, Jaime Delgado; A Media Value Chain Ontology for MPEG-21, IEEE