Project Number: FP7-257123

Project Title: CONVERGENCE

Dissemination Level: Public

Deliverable Number: D7.3

Contractual Date of Delivery to the CEC: 30.09.2012

Actual Date of Delivery to the CEC: 08.11.2012

Title of Deliverable: Tools and Sample Applications Final Release

Work-package contributing to the Deliverable: WP 7

Nature of the Deliverable: Prototype

Editors: Stelios Pantelopoulos and Panagiotis Gkonis

Author(s): Stelios Pantelopoulos, Panagiotis Gkonis (SIL), Aziz Mousas, Angelos – Christos Anadiotis, Charalampos Patrikakis (ICCS), Fernando Almeida (INESC), Daniel Sequeira (WIPRO), Alina Hang (LMU), Francis Lemaitre (FMSH), Bogdan Ardeleanu, Mihai Tanase (UTI), Silke Geisen, Carsten Rust (MOPRHO), Sam Minelli (Alinari)

Abstract:

D7.3 is classified in the DoW as a prototype. Following the completion of the D7.3 prototype, comprising Tools and Applications, the consortium decided to issue this report, considered complementary to the prototype.

The report, bearing the same name as the prototype, describes the final release of the Tools and Applications developed in the CONVERGENCE project, based on the technical specifications described in deliverables 7.1 and 7.2, and the reference implementation of the CONVERGENCE Network and Middleware.

Keyword List: Tools, applications, APIs, use cases, implementation

Executive Summary

D7.3 is classified in the DoW as a prototype. Following the completion of the D7.3 prototype, comprising Tools and Applications, the consortium decided to issue this report, considered complementary to the prototype.

The report, bearing the same name as the prototype, describes the final release of the Tools and Applications developed in the CONVERGENCE project, based on the technical specifications described in deliverables 7.1 and 7.2, and the reference implementation of the CONVERGENCE Network and Middleware [3], [4]. The first part (chapters 2 and 3), presents the CONVERGENCE Tools in terms of their APIs and associated functionality. Additional material presents testing and verification methods and describes the workflow for the development of a CoApp (chapter 3). The second part (chapters 4, 5, 6 and 7) describes the development of the applications. The final part (chapter 8) presents an integrated application that combines features from the Alinari Photographic Archive Management (PAM) application and the two retailing applications from UTI and WIPRO. The goal is to show interoperability among individual applications and identify practical issues for integration and exploitation.

TABLE OF CONTENTS

LIST OF FIGURES

Glossary

Term	Definition
Access Rights	Criteria defining who can access a VDI or its components 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 a way in which the CONVERGENCE System may be used by specific users in a specific context or, more narrowly, a scenario describing the products and services bought and sold, the actors concerned and, possibly, the associated flows of revenue in such a context.
CA	Central Authority
CCN	Content Centric Network
Cl_Auth_SC	Client Authentication with Smart Card (Challenge Response)
Cl_Auth_User_Pw	Client Authentication with Username and Password
Clean-slate architecture	The CONVERGENCE implementation of the Network Level, totally replacing existing IP functionality. See “Integration Architecture” and ”“Overlay Architecture” and “Parallel Architecture”.
CoApp	The CONVERGENCE Application Level.
CoApp Provider	A user providing Applications running on the CONVERGENCE Middleware Level (CoMid).
CoMid	The CONVERGENCE Middleware Level.
CoMid Provider	A user providing access to a single or an aggregation of CoMid services.
CoMid Resource	A virtual or physical object or service referenced by a VDI, e.g. media, Real World Objects, persons, internet services. It has the same meaning of “Resource” and it is used only to better specify the term “Resource” when there is a risk of a misunderstanding with the term “CoNet Resource”.
Community Dictionary	A CoMid Technology Engine that provides all the matching

Service (CDS)	concepts in a user’s subscription, search request and publication.
CoNet Provider	A user providing access to CoNet services, i.e. the equivalent of an Internet Service Provider.
CoNet Resource	A resource of the CoNet that can be identified by means of a name; resources may be either Named-data or a Named service access point.
Content-based resource discovery	A user request for resources, either through a subscription or a search request to the CONVERGENCE system (from literature). See “subscription” and “search”.
Content-based Subscription	A subscription based on a specification of user’s preferences or interests, (rather than a specific event or topic). The subscription is based on the actual content, which is not classified according to some predefined external criterion (e.g., topic name), but according to the properties of the content itself. See “Subscription” and “Publish-subscribe model”.
Content-centric	A network paradigm in which the network directly provides users with content, and is aware of the content it transports, (unlike networks that limit themselves to providing communication channels between hosts).
CONVERGENCE Applications level (CoApp)	The level of the CONVERGENCE architecture that establishes the interaction with CONVERGENCE users. The Applications Level interacts with the other CONVERGENCE levels on behalf of the user.
CONVERGENCE Computing Platform level (CoComp)	The Computing Platform level provides content-centric networking (CoNet), secure handling (CoSec) of resources within CONVERGENCE and computing resources of peers and nodes.
CONVERGENCE Core Ontology (CCO)	A semantic representation of the CoReST taxonomy. See “CONVERGENCE Resource Semantic Type (CoReST)”
CONVERGENCE Device	A combination of hardware and software or a software instance that allows a user to access Convergence functionalities
CONVERGENCE Engine	A collection of technologies assembled to deliver specific functionality and made available to Applications and to other Engines via an API
CONVERGENCE Middleware level (CoMid)	The level of the CONVERGENCE architecture that provides the means to handle VDIs and their components.
CONVERGENCE	The Content Centric component of the CONVERGENCE

Network (CoNet)	Computing Platform level. The CoNet provides access to named-resources on a public or private network infrastructure.
CONVERGENCE node	A CONVERGENCE device that implements CoNet functionality and/or CoSec functionality.
CONVERGENCE peer	A CONVERGENCE device that implements CoApp, CoMid, and CoComp (CoNet and CoSec) functionality.
CONVERGENCE Resource Semantic Type (CoReST)	A list of concepts or terms that makes it possible to categorize a resource, establishing a connection with the resource’s semantic metadata.
CONVERGENCE Security element (CoSec)	A component of the CONVERGENCE Computing Platform level implementing basic security functionality such as storage of private keys, basic cryptography, etc.
CONVERGENCE System	A system consisting of a set of interconnected devices - peers and nodes - connected to each other built by using the technologies specified or adopted by the CONVERGENCE specification. See “Node” and “Peer”.
Dec_Key_Unwrap	Key Unwrapping and Content Decryption
DIDL	Digital Item Description Language
Digital forgetting	A CONVERGENCE system functionality ensuring that VDIs do not remain accessible for indefinite periods of time, when this is not the intention of the user.
Digital Item (DI)	A structured digital object with a standard representation, identification and metadata. A DI consists of resource, resource and context related metadata, and structure. The structure is given by a Digital Item Declaration (DID) that links resource and metadata.
Domain ontology	An ontology, dedicated to a specific domain of knowledge or application, e.g. the W3C Time Ontology and the GeoNames ontology.
Elementary Service (ES)	The most basic service functionality offered by the CoMid.
Enc_Key_Wrap	Encryption and Key Wrapping
Entity	An object, e.g. VDIs, resources, devices, events, group, licenses/contracts, services and users, that an Elementary Service can act upon or with which it can interact.
Expiry date	The last date on which a VDI is accessible by a user of the CONVERGENCE System.

Fractal	A semantically defined virtual cluster of CONVERGENCE peers.
Group_Sig	Group Signature
ICN	Information Centric Network
Identifier	A unique signifier assigned to a VDI or components of a VDI.
Integration Architecture	An implementation of CoNet designed to integrate CoNet functionality in the IP protocol by means of a novel IPv4 option or by means of an IPv6 extension header, making IP content-aware. See “Clean-state Architecture”, “Overlay Architecture”, “Parallel Architecture”
IP	Identity Provider
License	A machine-readable expression of Operations that may be executed by a Principal.
Local named resource	A named-resource made available to CONVERGENCE users through a local device, permanently connected to the network. Users have two options to make named-resources available to other users: 1) store the resource in a device, with a permanent connection to the network; 2) use a hosting service. In the event she chooses the former option, the resource is referred to as a local named-resource.
Metadata	Data describing a resource, including but not limited to provenance, classification, 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 ISO/IEC standard that includes the previous MXM standard.
Multi-homing	In the context of IP networks, the configuration of multiple network interfaces or IP addresses on a single computer.
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. A named-service-access-point is a network endpoint identified by its name rather than by the Internet port numbering mechanism.

Named-data	A named-resource consisting of data.
Network Identifier (NID)	An identifier identifying a named resource in the CONVERGENCE Network. If the named resource is a VDI or an identified VDI component, its NID may be derived from the Identifier (see “Identifier”).
Overlay architecture	An implementation of CoNet as an overlay over IP. See “Clean-state Architecture” and “Integration Architecture” and “Parallel Architecture”
Parallel architecture	An implementation of CoNet as a new networking layer that can be used in parallel to IP. See “Clean-state Architecture” and “Integration Architecture” and ”“Overlay Architecture”
PKI	Public Key Infrastructure
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.
Principal (CoNet)	The user who is granted the right to use a *CoNet Principal Identifier* for naming its named resources. For example, the principal could be the provider of a service, the publisher or the author of a book, the controller of a traffic lights infrastructure, or, in general, the publisher of a VDI. A Principal may have several Principal Identifiers in the CoNet.
Principal (Rights Expression Language)	The User to whom Permissions are Granted in a License.
Principal Identifier (CoNet)	The Principal identifier is a string that is used in the Network Identifiers (NID) of a CoNet resource, when the NID has the form: NID = <namespace ID, hash (Principal Identifier), hash (Label)> In this approach, hash (Principal Identifier) must be unique in the namespace ID, and Label is a string chosen by the principal in such a way that hash(Label) is unique for in the context of the Principal Identifier.
Publish	The act of informing an identified subset of users of the CONVERGENCE System that a VDI is available.
Publisher	A user of CONVERGENCE who performs the act of publishing.
Publish-subscribe model	CONVERGENCE uses a content-based approach for the publish-subscribe model, in which notifications about VDIs are delivered to a subscriber only if the metadata / content of those VDIs match

	constraints defined by the subscriber in his Subscription VDI.
Real World Object	A physical object that may be referenced by a VDI.
REL	Rights Expression Language
Resource	A virtual or physical object or service referenced by a VDI, e.g. media, Real World 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.).
Serv_Auth	Server Authentication without Smart Card
Service Level Agreement (SLA)	An agreement between a service provider and another user or another service provider of CONVERGENCE to provide the latter with a service whose quality matches parameters defined in the agreement.
Sig	Signature
Smart_Card Role_Auth_SC	Role Authentication towards Smart Card
SP	Service Provider
Subscribe	The act whereby a user requests notification every time another user publishes or updates a VDI that satisfies the subscription criteria defined by the former user (key value pairs in the metadata, free text, key words etc.).
Subscriber	A user of CONVERGENCE who performs the act of subscribing.
Timestamp	A machine-readable representation of a date and time.
Tool	Software providing a specific functionality that can be re-used in several applications.
Trials	Organized tests of the CONVERGENCE System in specific business scenarios.
Un-named-data	A data resource with no NID.
Us_Reg_IP	User Registration to Identity Provider
Us_Reg_SP	User Registration to Service Provider

User	Any person or legal entity in a Value-Chain connecting (and including) Creator and End-User possibly via other Users.
User (in OSI sense)	In a layered architecture, the term is used to identify an entity exploiting the service provided by a layer (e.g. CoNet user).
User ontology	An ontology created by CONVERGENCE users when publishing or subscribing to a VDI.
User Profile	A description of the attributes and credentials of a user of the CONVERGENCE System.
Versatile Digital Item (VDI)	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 Introduction

D7.3 is classified in the DoW as a prototype. Following the completion of the D7.3 prototype, comprising Tools and Applications, the consortium decided to issue this report, considered complementary to the prototype.
The report, bearing the same name as the prototype, describes the final release of the CONVERGENCE Tools and Applications (CoApps), providing a detailed description of the APIs for the Tools, along with testing and verification methods and examples. We then provide a detailed description of the five CoApps that were used in the first two CONVERGENCE Trials: Photos in the Cloud and Down to Earth, Videos in the Cloud and Down to Earth, Augmented Lecture Podcast, Smart Retailing (WIPRO and UTI). Finally we describe an integrated application that combines three separate CoApps. The rest of the deliverable is organized as follows:

- **Chapter 2 - Description of the APIs for CONVERGENCE Tools**. This chapter describes the final version of the APIs for the tools. For each API, we provide a short description and testing methods.

- **Chapter 3 - Building a CONVERGENCE Application**. This section describes the step-by-step development and deployment of a CoApp.

- **Chapters 4, 5, 6 and 7**. Each of these chapters is devoted to one of the CoApps used in the first two CONVERGENCE Trials. Each chapter begins with a short description of the use case and the supported applications. The analysis goes on to provide a functional description of the application (i.e. Methods for creating application VDIs and Web interface analysis).

- **Chapter 8.** This chapter presents an integrated application based on the Alinari, WIPRO and UTI CoApps. The application facilitates search and retrieval of digital content related to photos, while at the same time allowing users to purchase components related to these photos (i.e. lenses, cameras, etc.).

- **Chapter 9 - Concluding remarks.** This chapter summarises the key findings of the deliverable and discusses an additional integrated scenario.

2 Description of the APIs for CONVERGENCE Tools

2.1 Introduction

The CONVERGENCE framework includes five basic Tools: User Registration, Resource VDI, Publish VDI, Subscribe to VDI, Revoke VDI. Other operations such as Creation of License, Metadata, or Event Report can be viewed as additional APIs to these Tools. For example, the APIs for the creation of license, ERR and metadata are shared between the Resource VDI, Publication VDI and Subscription to VDI Tools.

This chapter describes the APIs for the Tools, along with testing and verification methods. Test datasets can be found in ANNEX B - Test Datasets. Given that the tools described here are the final version of the tools already presented in D7.2, the descriptions inevitably contain a large amount of duplicate material from the previous report.

2.2 User registration

The first step in each scenario is for the user to register with an identity provider, who provides her with a certificate and a personal smart card, which stores an automatically generated key pair and the certificate. The user also receives a unique identifier. After registering with the identity provider, the user can register with a service provider, using a pseudonym (if this is allowed by the application). After registering with the Identity Provider and the Service Provider, the user can authenticate herself, either using a username and password or the smart card. More details can be found in D7.2 [2], section 3.

2.2.1 User registration APIs

Method Summary	
IdentifyUserResponse	**identifyUser** (IdentifyUserRequest request): Registers the user
String	**getUserId**(PublicKey): returns the id connected to the given public key
AuthenticateUserResponse	**AuthenticateUser**(AuthenticateUserRequest): Returns the returned response has valid SAML response from the identity provider

2.2.2 Testing and verification

2.2.2.1 **identifyUserTest**

Name of test case	testIdentifyUserRequestCreation
Initial conditions	Pin number set to the security engine via set property method

Description of testing functionality	This test case test the user registration Mysql database running
Engines involved	Security engine
Detailed steps	1. Create user entity [containing certificate, user details] 2. Call identify user 3. Get the id and check if it is valid id i.e. not -1
Input	IdentifyUserRequest
Output	User id
Expected result	User id is not -1

authenticateUserTest

Name of test case	testAuthenticateUserRequest
Initial conditions	Pin number set to the security engine via set property method Mysql data base running
Description of testing functionality	This test case the authenticate the user request
Engines involved	Security engine
Detailed steps	1. User has to receive a valid SAML assertion from the identity provider [For this user has to be make request for saml response authenticateUser(saml request), this will return identifyProviderResponse.This response contains the signed assertion] 2. Take this assertion and create AuthenticateUserRequest 3. call authenticateUser(authenticateUserRequest) 4. Service provider return with saml response along with decision for access granted or not based on the attached assertion in the request
Input	AuthenticateUserRequest
Output	AuthenticateUserResponse
Expected result	AuthenticateUserResponse with specific data assertTrue(serviceProviderResponse.getApplicationSpecificData().size()>0)

2.3 Security Tool

The Security Tool provides general confidentiality methods as well as methods for signature handling. A configuration file (MXMConfiguration.xml) allows the user to choose whether to use the CONVERGENCE smart card as a crypto module or to use a less secure off-card solution. The main functionality of the Security Tool is to generate and verify signatures. Other important (internal) functions include the generation of symmetric keys, symmetric/hybrid decryption and encryption, the generation of key pairs and the wrapping and unwrapping of keys. For more details see D7.2 [2], section 3.

2.3.1 Security Tool APIs

Method Summary	
java.security.PublicKey	**generateKeyPair**(java.lang.String keyIdentifier, int keyLength) Generates the RSA key pair and returns the public key from the pair
org.iso.mpeg.mxm.tEngine.securityTE.schemahandler.dsig.Signature	**generateSignature**(byte[] message) generates the signature of given byte array
org.iso.mpeg.mxm.tEngine.securityTE.schemahandler.dsig.Signature	**generateSignature**(org.w3c.dom.Document doc) Generates the signature of given doc.
javax.crypto.SecretKey	**generateSymmetricKey**(int keyLength) Generates symmetric AES key of given length
static SecurityTool	**getInstance**()
byte[]	**hybridDecrypt**(byte[] data) Decrypts the given encrypted hybrid key with the private key of type "conf"
byte[]	**hybridEncrypt**(byte[] data) Encrypts the data with wrapped key and returns the wrapped key and also stores it in .
boolean	**StreamHybridDecrypt**(byte[] data) Decrypts the data encrypted with wrapped key that was returned in hybridEncrypt(data) method
byte[]	**symmetricDecrypt**(byte[] data, javax.crypto.SecretKey symmetricKey) Decrypts the data with provided symmetric AES key
byte[]	**symmetricEncrypt**(byte[] data, javax.crypto.SecretKey symmetricKey) Encrypts the data with provided symmetric AES key
javax.crypto.SecretKey	**unWrap**(org.iso.mpeg.mxm.tEngine.securityTE.schemahandler.xenc.EncryptedKey wrappedKey) Unwraps the given encrypted key and return the Secret key [private key of type "conf" is used]
boolean	**verifySignature**(byte[] message, org.iso.mpeg.mxm.tEngine.securityTE.schemahandler.dsig.Signature signature)

	Verifies the signature of given byte array [private key of type "sig" is used for this purpose]
boolean	**verifySignature**(org.w3c.dom.Document doc, org.iso.mpeg.mxm.tEngine.securityTE.schemahandler.dsig.Signature signature) Verifies the signature of given document [private key of type "sig" is used]
org.iso.mpeg.mxm.tEngine.securityTE.schemahandler.xenc.EncryptedKey	**wrap**(javax.crypto.SecretKey secretKey) Wraps the given secret key and return the encrypted key [public key of type "conf" is used]

2.3.2 Testing and verification

2.3.2.1 securityToolTest

Name of test case	testSecurityToolParsing
Initial conditions	Pin number is set to the security engine via set property method Mysql data base running
Description of testing functionality	This test examines the basic key generation and wrapping and unwrapping of symmetric key
Engines involved	Security engine
Detailed steps	A: General encryption and decryption 1.Generate key pair 2. Generate AES symmetric key 3. Encrypt data with symmetric key 3. Wrap the symmetric key with public key to test wrap mechanism 4. Unwrap the wrapped key 5. Decrypt the with unwrapped key and check the decrypted data with original B: Begin with XML signing and verifying 1. Reade xml file from a location 2. Get org.w3c.dom.Document object 3. Generate signature 4. Verify signature
Input	AuthenticateUserRequest
Output	AuthenticateUserResponse
Expected result	AuthenticateUserResponse with specific data assertTrue(serviceProviderResponse.getApplicationSpecificData().size()>0)

2.4 VDI Tool

In the final version of the CONVERGENE applications, the Resource VDI (R-VDI), Publication VDI (P-VDI) and Subscription VDI (S-VDI) tools have been merged in a new VDI Tool. This change makes the tools easier to manage during the development process, avoiding the need to fix the same bug in multiple tools. The new tool provides the complete functionality required to create and parse R-VDIs, P-VDIs and S-VDIs. The tool thus provides six different interfaces, as well as the R/P/SvdiCreator and the R/P/SvdiParser. The effects on VDI Tool interfaces are limited: the main changes, described in 2.4.1.1, concern the way the tool is initialized.

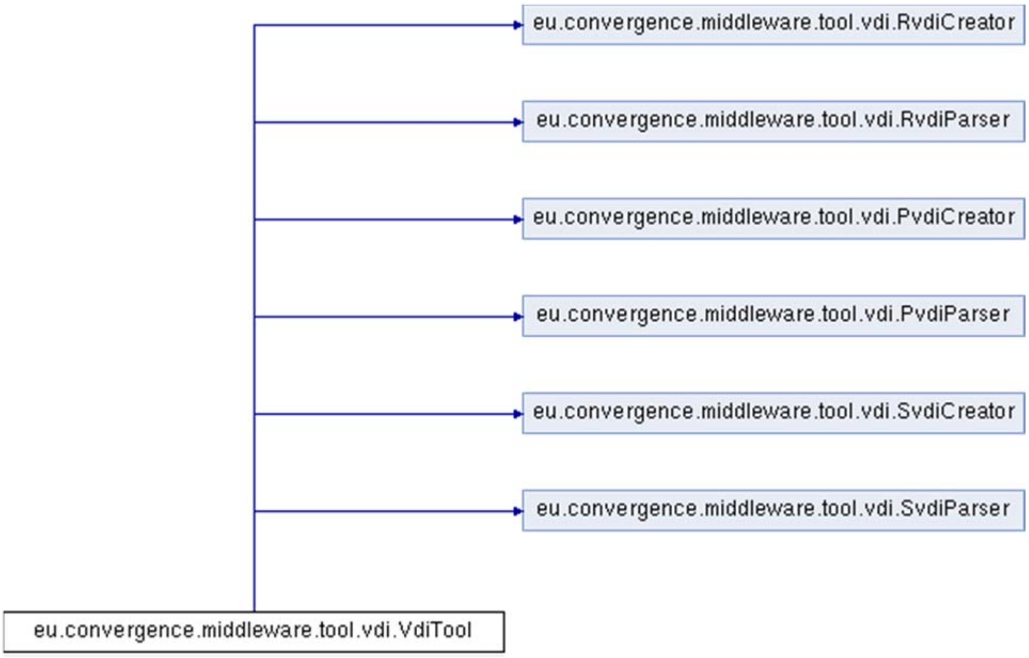

eu.convergence.middleware.tool.vdi.RvdiCreator
eu.convergence.middleware.tool.vdi.RvdiParser
eu.convergence.middleware.tool.vdi.PvdiCreator
eu.convergence.middleware.tool.vdi.PvdiParser
eu.convergence.middleware.tool.vdi.SvdiCreator
eu.convergence.middleware.tool.vdi.SvdiParser
eu.convergence.middleware.tool.vdi.VdiTool

Figure 2-1: Inheritance diagram of the VDI Tool

2.4.1 VDI Tool APIs

2.4.1.1 **VDI Tool constructors**

The VDI Tool can be instantiated using one of the following ten static constructors, providing the RvdiCreator/Parser, PvdiCreator/Parser and SvdiCreator/Parser interfaces.

Constructor Summary	
RvdiCreator	**newRvdiCreator**() Constructs a new R-VDI Creator instance.
RvdiParser	**newRvdiParser(String rVdiId)** Constructs a new R-VDI Parser instance for the R-VDI with id rVdiId.
RvdiParser	**newRvdiParser(File rVdiFile)** Constructs a new R-VDI Parser instance for the specified R-VDI file.

PvdiCreator	**newPvdiCreator(String rVdiId)** Constructs a new P-VDI Creator instance for publishing the R-VDI with id rVdiId.
PvdiCreator	**newPvdiCreator(File rVdiFile)** Constructs a new P-VDI Creator instance for publishing the specified R-VDI file.
PvdiParser	**newPvdiParser(String pVdiId)** Constructs a new P-VDI Parser instance for the P-VDI with id pVdiId.
PvdiParser	**newPvdiParser(File pVdiFile)** Constructs a new P-VDI Parser instance for the specified P-VDI file.
SvdiCreator	**newSvdiCreator**() Constructs a new S-VDI Creator instance.
SvdiParser	**newSvdiParser(String sVdiId)** Constructs a new S-VDI Parser instance for the S-VDI with id sVdiId.
SvdiParser	**newSvdiParser(File sVdiFile)** Constructs a new S-VDI Parser instance for the specified S-VDI file.

2.4.1.2 RvdiCreator

This interface specifies the creation of a R-VDI, providing methods for adding metadata, licenses, resource references and relationships to other R-VDIs. It also exposes methods to package an R-VDI with resources, add a digital signature and store it.

Method Summary	
String	**getIdentifier**() Returns the identifier of the generated VDI.
void	**setStartDate**(Date startDate) Sets the start date of validity of the VDI.
void	**setExpiryDate**(Date expiryDate) Sets the expiration date of the VDI.
void	**addKeyword**(String keyword) Adds a keyword description to the VDI, e.g. comedy.
void	**addTag**(String tag) This method adds a tag description to the VDI. The tag differs to the keyword description in the way that tags are URIs of a controlled vocabulary.

void	**addFieldValue**(String field, Object value) Adds a field/value pair description to the VDI, e.g. genre=comedy. Note that value is of type object. This enables the use of non-String values, namely double, integer or date values. This way the value preserves its data type.
void	**addStructuredData**(StructuredDatastructuredData) The method adds structured metadata descriptions to the VDI. The structured metadata objects can be created using the Metadata Tool, producing RDF/XML descriptions based on ontologies.
Resource	**createResource**(String resourceURL, String mimeType, String content) This method creates a resource reference descriptor. An optional mimeType parameter gives the option of specifying the content type.
void	**addAsset**(Resource resource, List<License> licenses, List<ERR>eventReportRequests) This method adds a resource in a R-VDI. The user may specify a list of licenses regarding the resource along with event report requests e.g for download events.
void	**addRelationship**(String relationshipType, String vdiId) Adds a relationship descriptor stating a semantic relationship between another VDI and the VDI to be created. The relationshipType may be a URI identifying an ontology relationship.
void	**addSignature**(Signature signature) Adds a signature to the VDI. Please note that the signature is not a simple hash of the VDI, but a complex XML signature.
DIDL	**generateRVDI**() This method allows the creation of the R-VDI, provided that metadata, licenses, etc. have been inserted.
void	**pack**(byte[][] resources) Packs the VDI with an array of resources.
boolean	**store**(String filePath) Stores the VDI to the specified filePath location. The method first generates the VDI, if not already done by the user, and stores it to the passed location.
boolean	**advertise**(String filePath) Advertises the R-VDI that is located in filePathin CoNet. The filepath is optional. If no valid string is provided, then the store method is first triggered to generate and store the R-VDI and then the tool employs CoNet TE to advertise it.

2.4.1.3 RvdiParser

This interface specifies methods for accessing a R-VDI and provides methods for extracting metadata, licenses, resource references and relationships to other VDIs.

Method Summary	
String	**getIdentifier**() Returns the identifier of the generated VDI.
Date	**getStartDate**() Returns the start validity date of the VDI.
Date	**getExpiryDate**() Returns the expiration date of the VDI.
List<String>	**getKeywordList**() The method returns all the keywords that describe the VDI.
List<String>	**getTagList**() The method returns all the tags that describe the VDI.
Map<String, List<Object>>	**getFieldValueMap**() The method returns all field value descriptions in a map object. Please note that for a given field more than one values can be added, hence the corresponding list.
List<StructuredData>	**getStructuredDataList**() Returns a list of all the structured data descriptors of the VDI.
Map<String, String>	**getResourceMap**() Returns a map object containing resource references and their mime type.
Map<String, String>	**getRelationshipsMap**() Returns a map object containing the identifiers and the relationship to them.
List<License>	**getLicenseList**() Returns a list containing all the licenses that are embedded in the VDI.
List<ERR>	**getERRList**() Returns a list containing all the event report requests that are embedded in the VDI.
List<Signature>	**getSignaturesList**() Returns a list containing all the signatures that are embedded in the VDI.

List<byte[]>	**unpack**() Returns a list of the embedded resources.

2.4.1.4 PvdiCreator

This interface specifies methods for creating a P-VDI and provides methods for adding metadata, licenses, and event report requests. It also exposes methods for injecting a P-VDI into the overlay, adding a digital signature and storing it. The interface also contains the addKeyword, addTag, addFieldValue, addStructuredMetadata, addSignature, setStartDate, setExpiryDate methods. Given that these methods are defined in the same way as in RvdiCreator, they are not described here.

Method Summary	
void	**setFractal**(String fractal) Sets the VDI destination fractal, which is identified by a string (e.g. photo fractal). This is later used by the Overlay TE to propagate the VDI to the semantic overlay.
void	**addERR**(ERR eventReportRequest) Adds an event report request associated with the VDI.
void	**addLicense**(License license) Adds a license to the VDI. Please note that the VDI can have multiple licenses embedded in it.
DIDL	**generatePVDI**() This method generates the P-VDI. It should have already been passed, metadata, licenses, ERRs etc. The Publish VDI Tool employs the Orchestrator TE to build a chain of engines and fuse all the submitted data and formulate a valid VDI object.
boolean	**publish**() Publishes the VDI to the CONVERGENCE cloud

2.4.1.5 PvdiParser

This interface specifies methods for accessing a P-VDI including methods for extracting metadata, licenses, and event report requests from a P-VDI. The API also contains getIdentifier, getSequenceIdentfier, getKeywordList, getTagList, getFieldValueMap, getStructuredMetadataList, getStartDate, getExpiryDate, getLicenseList, getERRList, getSignatureList methods. Given that these methods are defined in the same way as in RvdiParser, they are not described here.

Method Summary

String	**getFractal**() The method returns the fractals that the VDI belongs to.

2.4.1.6 SvdiCreator

This interface specifies methods for creating a S-VDI, including methods for adding conditions, licenses, and event report requests. It also exposes methods for actions that can be performed to a S-VDI, namely injecting it into the semantic overlay, adding a digital signature or storing it. The API also contains the addLicense, addSignature, setStartDate, setExpiryDate methods, which are defined in the same way as in RvdiCreator.

Method Summary	
void	**setFractal**(String fractal) Sets the VDI destination fractal, which is identified by a string (e.g. photo fractal).
void	**addERR**(ERR eventReportRequest) Adds an event report request associated with the VDI.
void	**addFieldValueCondition**(String field, String operator, Object value) This method adds a field value condition to the subscription query. It is possible to express also inequality conditions using a graterThan or a lessThan operator. Also notice that the value is of object type, preserving this way its datatype e.g. Date.
void	**addKeywordCondition**(String keyword) Adds a keyword condition to the subscription query
void	**addTagCondition**(String tag) Adds a tag condition to the subscription query
DIDL	**generateSVDI**() This method generates the S-VDI. It should have already been passed, metadata, licenses, errs etc. The Subscription VDI Tool employs the Orchestrator TE to build a chain of engines and fuse all the submitted data and formulate a valid VDI object.
String	**subscribe**() Sends the S-VDI to the CONVERGENCE cloud
void	**addSparqlTripleCondition**(String subject, String predicate, String object) Adds a triple condition to the sparql query. Variables e.g. ?x can also be stated here
void	**addSparqlLiteralCondition**(String subject, String predicate, Object value) Adds a triple literal condition to the sparql query. That is, the object of the condition has a datatype e.g. integer.

void	**addSparqlFilterCondition**(String object, String operator, Object value) This method enables the definition of a filter condition based on the value of a property. The object must be a variable that is already stated as an object in a triple condition pattern.
void	**addSparqlReturnVariable**(String varName) This method sets the variable that will be returned from the query.
void	**setResultLimit**(longlimit) This method sets a limit for the returned results.

2.4.1.7 SvdiParser

This interface specifies methods for accessing a S-VDI, including methods for extracting subscription conditions, licenses, and event report requests. It also exposes methods for actions that can be performed on a S-VDI, namely injecting it into the semantic overlay, adding a digital signature and storing it. The API also contains the getIdentifier, getStartDate, getExpiryDate, getLicenseList, getERRList and getSignatureList methods, which are defined in the same way as in the RvdiParser.

Method Summary	
String	**getFractal**() The method returns the fractals that the VDI belongs to.
List<String>	**getKeywordConditions**() Returns subscription’s keyword conditions.
List<String>	**getTagConditions**() Returns subscription’s tag conditions.
List<String>	**getFieldValueConditions**() Returns subscription’s tag conditions.
String	**getSparqlQuery**() Returns subscription’s SPARQL query.

2.4.2 VDI Tool testing and verification

2.4.2.1 R-VDI creation

Name of test case	testRVDICreation
Initial conditions	*RVDI file, REL license object, StructuredData metadata object*
Description of testing functionality	This test case verifies the proper creation of an R-VDI file, containing a resource reference, metadata about the resource and a license. The R-VDI is then stored and advertised in CoNet.

Tools involved	RvdiCreator
Detailed steps	1. Create Resource a. Set resourceUrl b. Set mimeType 2. Add an Asset a. Set Resource b. Set License 3. Set metadata a. Add keywords b. Add tags c. Add StructuredData 4. Set start and expiry date 5. Generate the R-VDI 6. Store R-VDI to disk 7. Check if the stored R-VDI and the dataset's RVDI file are identical 8. Advertise R-VDI file in CoNet 9. Check if the advertisement was successful
Dataset [12.1.1]	RVDI file, REL license file, RDF metadata file
Output	RVDI file.
Expected result	The generated RVDI file must be identical with the datasets RVDI file.
Variations	*<none>*

2.4.2.2 R-VDI parsing

Name of test case	testRVDIParsing
Initial conditions	*RVDI file, REL license file, StructuredData metadata*
Description of testing functionality	This test case verifies the proper parsing of a R-VDI file, which contains a resource, metadata about the resource and a license.
Tools involved	RvdiParser
Detailed steps	1. Parse R-VDI file a. Check and print R-VDI identifier b. Check and print start data c. Check and print expiry date d. Check and print keywords e. Extract and check StructuredData with dataset file. f. Extract Resources 2. For each Resource a. Check and print resourceUrl b. Check and print mimeType c. Extract and check License with dataset file
Dataset [12.1.1]	RVDI file, REL license file, RDF metadata file
Output	Checks and prints information of the RVDI.
Expected result	The parsing of the RVDI and the various checks must be successful

Variations	*<none>*

2.4.2.3 P-VDI creation

Name of test case	testPVDICreation
Initial conditions	*RVDI file, PVDI file, ERR object*
Description of testing functionality	This test case verifies the proper creation of a P-VDI file. The P-VDI is then published in the specified fractal.
Tools involved	PvdiCreator
Detailed steps	1. Parse R-VDI file 2. Set fractal 3. Add ERR match object 4. Generate the P-VDI 5. Check if the generated P-VDI and the dataset's PVDI file are identical 6. Publish P-VDI file. 7. Check if P-VDI is successfully published
Dataset [12.1.2]	RVDI file, REL license file, RDF metadata file
Output	PVDI file.
Expected result	The generated PVDI file must be identical with the datasets PVDI file and the publication procedure should be successfully performed.
Variations	*<none>*

2.4.2.4 S-VDI creation

Name of test case	testSVDICreation
Initial conditions	*SVDI file, ERR object*
Description of testing functionality	This test case verifies the proper creation of an S-VDI. The generated S-VDI contains a SPARQL query and is announced in the specified fractal.
Tools involved	SvdiCreator
Detailed steps	1. Set fractal 2. Add ERR match object 3. Create SPARQL subscription a. Set the returned variable b. Set a triple condition c. Set a triple literal condition d. Set a filter condition 4. Generate the S-VDI 5. Check if the generated S-VDI and the dataset's SVDI file are identical 6. Announce S-VDI to the fractal. 7. Check if S-VDI announcement is successfully carried out
Dataset [12.1.3]	SVDI file

Output	SVDI file
Expected result	The generated SVDI file must be identical with the datasets SVDI file and the subscription procedure should be successfully performed.
Variations	*<none>*

2.4.2.5 S-VDI parsing

Name of test case	testSVDIParsing
Initial conditions	*SVDI file*
Description of testing functionality	This test case verifies the proper parsing of aS-VDI file, which contains a mixed MPQF query, with both keyword conditions and a SPARQL query.
Tools involved	SvdiParser
Detailed steps	1. Parse S-VDI file a. Check and print S-VDI identifier b. Check and print start data c. Check and print expiry date d. Check and print keyword conditions e. Check and print SPARQL query
Dataset [12.1.3]	SVDI file
Output	Checks and prints information of the SVDI.
Expected result	The parsing of the SVDI and the various checks must be successful.
Variations	*<none>*

2.5 Revoke VDI

This tool allows CONVERGENCE users to revoke a P-VDI or an S-VDI from CoNet or CoMid. The tool uses the CoNet TE for CoNet revocation and the Overlay TE to propagate the revocation request to peers. The user can specify the time and date at which the VDI will expire during the publishing process. It is also possible for an authorized user to invoke the digital forgetting mechanism to unpublish a P-VDI (published at some time in the past), for which he/she has "unpublish rights".

The Revoke Tool has been refactored since the last release of the tools, integrating various changes in the middleware engines. The new version makes it possible to revoke a VDI from the middleware. The gorest message is automatically sent to the subscribed pears, as described in section Revoke CoMid test case.

2.5.1 Revoke Tool APIs

Method Summary	
boolean	**revokeFromConet**(String nid) Revokes a VDI with nid from CoNet.

boolean	**unpublishFromComid**(String vdiId) Unpublish a VDI with vdiId from CoMid.

2.5.2 Revoke CoNet test case

Name of test case	testRevokeConetMethods
Initial conditions	Digital Item (VDI), NID object reference
Description of testing functionality	This test case verifies the revocation of a VDI from CoNet.
Tools involved	Revoke PE, Conet TE
Detailed steps	1. Creation of NID object 2. Store and Advertise of named-resource
Dataset	VDI file
Output	VDI file
Expected result	Output VDI file should not include NID object
Variations	*<not applied>*

The data input/output files are available in section 12.4.

2.5.3 Revoke CoMid test case

Name of test case	testRevokeComidMethods
Initial conditions	Digital Item (VDI), VDI Item Identifier
Description of testing functionality	This test case verifies the revocation of a VDI Item from the CoMid infrastructure
Tools involved	Revoke PE, VDI TE, Overlay TE
Detailed steps	1. Process Digital Item 2. Creation of unpublish DI 3. Creation of a new Gorest Message 4. Gets fractals locations 5. Propagate over Overlay Engine
Dataset	VDI file
Output	VDI file
Expected result	Output VDI file should not include the Digital Item
Variations	*<not applied>*

The data input/output files are available in section 12.4.

2.6 Event Report Tool

The Event Report Tool provides methods to create and parse event report (ER) and event report request (ERR) objects. The Event Report Tool has been refactored since the last release of the tools, integrating various changes in the middleware engines. It now combines the browse event tool and the create event tool in a single tool. These changes have only a limited effect on event reporting operations.

2.6.1 Event Report Tool APIs

These APIs allow a user to create and parse an ER (Event Report) and/or an ERR (Event Report Request). The user begins by creating the ERR and then associates the ERR with a Resource, Subscription or Publication VDI.

Method Summary	
ERR	The user provides the verb as the event trigger type. The user may use the Event Report Tool to create the ERR and after that will associate the ERR with the VDI (Resource, Subscription or Publication). The ERR contains the peer service access point name. In order to create an ERR, the tool will do the following operation: 1. Create an ERRobject 2. Set the identifier (internal generated) 3. Set the verb (provided as method parameter) 4. set the SAP (get it from the EREngine)
String	**getDIDLIId**(ER eventReport) Parses an ER and returns the DIDL identifier
String	**getERRIdentifier**(ER eventReport) Parses an ER and returns the ERR identifier.
String	**getERRIdentifier**(ERR eventReportRequest) Parses an ERR and returns the ERR identifier.
static EventReportTool	**getInstance**() Gets the EventReportTool instance.
String	**getVerb**(ERR eventReportRequest) Parses an ER and returns the verb (match, notify etc.)

2.6.2 Event Report Tool testing and verification

2.6.2.1 Create event report request

Name of test case	testERRCreation
Initial conditions	-
Description of testing functionality	This test case verifies proper creation of event report request object (ERR)
Tools involved	EventReportTool
Detailed steps	1. Instantiate EventReportTool 2. Create ERR using EventTriggerType: match

	3. Parse ERR to get ERR identifier 4. Check ERR creation without any errors
Dataset	-
Output	Prints a message with the ERR identifier generated during the creation.
Expected result	An ERR object gets created without any exception
Variations	*<not applied>*

2.7 License Tool

The License Tool provides methods to create and parse a license expressed in the Rights Expression Language (REL) with the help of the REL TE. Additional methods support the verification of a license's validity and querying a license for authorization purpose.

The License Tool has been refactored since the last release of the tools. The new version takes account of changes to the middleware engines and now offers two separate interfaces: the License Creator for creating licenses and the License Parser for parsing and querying existing licenses. These modifications, described in section 2.7.1.1, have only a limited effect on the License Tool interfaces and mainly concern the way the tool is initialized.

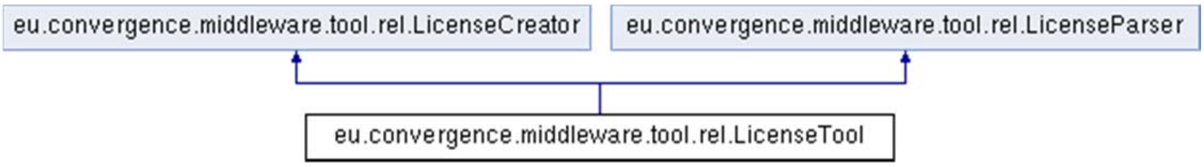

eu.convergence.middleware.tool.rel.LicenseCreator
eu.convergence.middleware.tool.rel.LicenseParser
eu.convergence.middleware.tool.rel.LicenseTool

Figure 2-2: Inheritance diagram of the License Tool

2.7.1 License Tool APIs

2.7.1.1 License Tool constructors

The License Tool can be instantiated using one of the following two static constructors, providing the LicenseCreator and LicenseParser interfaces.

Constructor Summary	
LicenseCreator	**newLicenseCreator**() Constructs a new LicenseCreator instance.
LicenseParser	**newLicenseParser(FilelicenseFile)** Constructs a new License Parser instance parsing a given license file.

2.7.1.2 LicenseCreator

This interface specifies creation methods for generating a REL license. It provides methods for creating principals, resources and conditions. These methods can be used to define grants and add them to the license.

Method Summary	
Principal	**createKeyHolder**(X509Data certificate) Creates a new KeyHolder who is identified by his certificate.
Principal	**createIdentityHolder**(String principalIdentifier) Creates a new IdentityHolderwho identified his principal Identifier.
PropertyPossessor	**createPropertyPossessor**(String propertyURI) Creates a Principal of type PropertyPossessor. Unlike KeyHolderand IdentityHolder principals, PropertyPossessorsare generic descriptors for principals who possess a particular property.
Void	**setLicenseIdentifier**(String licenseId) Sets the identifier of the license.
Resource	**createDIreference**(String diIdentifier) Creates a new Resource, which references the Digital Item with the given identifier.
Resource	**createDigitalResource**(Stringurl) Creates a new Digital Resource, which can be accessed with the given url.
Resource	**createProtectedResource**(String url, byte[] key) Creates a new Protected Recource, which can be accessed with the given url and decrypted with the given key.
Condition	**createValidityIntervalCondition**(Date notBefore, Date notAfter) Creates a Validity Interval Condition during which the license allows the exercise of the right over the resource.
Void	**setIssuer**(Principalissuer) Sets the issuer of the license.
Void	**setPrincipal**(Principalprincipal) Sets the principal of the license.
Void	**setForAll**(PropertyPossessorpropertyPossessor) Sets a generic principal for the license, who is identified by the possession of a property.
Void	**addGrant**(RELRightright, Condition condition) Adds a new Grant to the license. Each Grant refers to a Principal, or a

	PropertyPossessor, who is granted a Right over the specified Resource under specified Conditions. The list of Rights that a can be included in the license are specified in the definition of the REL e.g. rights to adapt, copy or play a resource.
Void	**setResource**(Resourceresource) Sets the resource that is governed by this license.
License	**generateLicense**() Generates a new license that contains all submitted data.
Boolean	**storeLicense**(String filePath) Stores the license to the filePath.
Void	**addSignature**(Signature signature) Adds a signature to the license.

2.7.1.3 LicenseParser

This interface provides methods for parsing a REL license i.e. methods for extracting the grants, the issuer and the resource of a license.

Method Summary	
Principal	**getIssuer**() Returns the issuer the license.
Principal	**getPrincipal**() Returns the Principal of the license.
List<Grant>	**getGrants**() Returns the grants stated in the license.
String	**getResourceUrl**() Returns the url of the resource that is governed by the license.
byte[]	**getEncryptedKey**() Returns the decryption key of a protected resource.
List<String>	**getPropertyPossessorProperties**() Returns the properties of the generic principals inside the license.
List<RELRight>	**getPropertyPossessorRights**(String propertyURI) Returns the rights that a PropertyPossessor has over the resource.
Signature	**getSignature**() Returns the signature of the license.

boolean	**verifyLicense**() Verifies the validity of the REL statements contained in the license.
AuthorizationResponse	**checkLicense**(Principal principal, RELRight right) Processes the license and checks the privileges of a principal over a certain resource. The query consists of a principal and a right, which are checked by the Authorization Manager of the REL TE to evaluate the Authorization Response.

2.7.2 License Tool testing and verification

2.7.2.1 **License creation**

Name of test case	testLicenseCreation
Initial conditions	*REL license file*
Description of testing functionality	This test case verifies the proper creation of a REL license. The license is stored on disk and is checked with the REL license file of the dataset.
Tools involved	License Tool
Detailed steps	1. Instantiate LicenseCreator. 2. Create KeyHolder1 a. Set his public key 3. Create PropertyPossessor1 a. Set the PropertyURI he should possess 4. Create a Reference to a DI a. Set the DI identifier 5. Create a Validity Interval Condition a. Set notBefore date b. Set notAfter date 6. AddPropertyPossessor1as ForAll principal 7. Set KeyHolder1 as Issuer 8. Set Reference to a DI as Resource 9. Add Grant a. Set Right b. Set Validity Interval Condition as Condition 10. Generate the License 11. Store License to disk 12. Check if the stored license and the datasets REL license files are identical
Dataset [12.2.1]	REL license file
Output	REL license file
Expected result	The generated license file must be identical with the datasets license file.
Variations	*This test has many variations, adding in the license more grants, specifying different conditions and rights.*

2.7.2.2 License parsing and authorization

Name of test case	testLicenseParsingAndAuthorization
Initial conditions	*REL license file*
Description of testing functionality	This test case verifies the proper parsing of a REL license which is stored in a local file. During the test, all grants of the license are parsed, printed and checked with expected values for principal, right, resource and conditions. The license is then queried if it grants specific rights.
Tools involved	License Tool
Detailed steps	1. Instantiate LicenseParser. 2. Parse the local REL license file a. Verify license b. Check and print Issuer public key c. Check and print Resource’s url and encrypted key d. Check and print PropertyPossessor’s propertyURI e. Extract Grants 3. For each Grant a. Check and print REL Right b. Check and print Condition 4. Check AuthorizationResponse of authorization queries a. Set Principal b. Set RELRight c. Check and print response.
Dataset [12.2.1]	REL license file
Output	Parses a REL license, verifies it and prints information stated inside it. It also prints AutorizationResults of existing queries.
Expected result	The license file must be successfully parsed and verified. The information inside it must be checked and have the expected values and the license queries should be answered correctly.
Variations	*This test has many variations, using as input file a license with more grants, different conditions, rights and resources.*

2.8 Metadata Tool

The Metadata Tool provides methods to create and parse metadata descriptions expressed with the Resource Description Framework (RDF). Additional methods expose CDS TE functionality for requesting ontology entities and for building rich semantic descriptions.

The Metadata Tool has been refactored since the last release of the tools, to take account of various changes in the middleware engines. It now offers three separate interfaces: the MetadataCreator for creating descriptions, the MetadataWrapper for parsing and wrapping RDF metadata and the MetadataRequester for querying the CDS TE about ontology entities. These modifications, described in section 2.8.1.1. have only a limited effect on the Metadata Tool interfaces and mainly concern the way the tool is initialized.

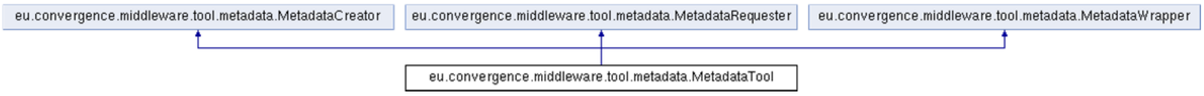

eu.convergence.middleware.tool.metadata.MetadataCreator
eu.convergence.middleware.tool.metadata.MetadataRequester
eu.convergence.middleware.tool.metadata.MetadataWrapper
eu.convergence.middleware.tool.metadata.MetadataTool

Figure 2-3: Inheritance diagram of the MetadataTool

2.8.1 Metadata Tool APIs

2.8.1.1 **Metadata Tool constructors**

The Metadata Tool can be instantiated using one of the following three static constructors, which provide the MetadataCreator, the MetadataWrapper or the MetadataRequester interfaces.

Constructor Summary	
MetadataCreator	**newMetadataCreator**() Constructs a new MetadataCreator instance.
MetadataWrapper	**newMetadataWrapper()** Constructs a new MetadataWrapper instance.
MetadataRequester	**newMetadataRequester()** Constructs a new MetadataRequester instance.

2.8.1.2 **MetadataCreator**

The MetadataCreator interface provides methods for creating RDF metadata structures and wraps them in a StructuredData element that can be used to describe a VDI.

Method Summary	
void	**addRDFTriple**(String subject, String predicate, String object) Adds a RDF triple to the description. The subject, relationship and object provided as parameters are transformed to a RDF statement. Note that subject, predicate and object must be valid URIs.
void	**addRDFTriple**(String subject, String predicate, Object value) Adds a RDF triple to the description. The subject, relationship and object are transformed to a RDF statement. Both subject and predicate must be valid URIs. Notice that the object is of anyType, allowing submitting also non-String values for datatype properties.
StructuredData	**Generate StructuredData()** The method returns a StructuredMetadata object containing the full description submitted by the user.

2.8.1.3 MetadataWrapper

This interface specifies methods for parsing RDF/XML metadata and wrapping the data in a StructuredData object, which can later be added to a VDI.

Method Summary	
void	**parseRDF**(String rdfFilePath, String baseURI) This method enables the parsing of existing rdf/xml files.
StructuredData	**generateStructuredData()** The method returns a StructuredMetadata object containing the full description submitted by the user.

2.8.1.4 MetadataRequester

This interface specifies the methods making it possible to request the entities in an ontology model, based on their labels.

Method Summary	
List<Map<String,List<String>>>	**requestMetadata**(String label, String type, String modelURI) Returns a list of metadata entities that satisfy the parameters passed. The modelURI parameter specifies the model where the entity will be searched. The label parameter specifies the prefix of the entity label. The type parameter specifies the type of the requested ontology entity that should be returned.

2.8.2 Metadata Tool testing and verification

2.8.2.1 Create metadata

Name of test case	testMetadataCreation
Initial conditions	StructuredData with RDF metadata file
Description of testing functionality	This test case verifies proper creation of RDF metadata with the MetadataCreator and the generation of the StructuredData object.
Tools involved	MetadataTool
Detailed steps	1. Instantiate MetadataCreator 2. Add a RDF triple 3. Add a RDF literal triple 4. Generate StructuredData 5. Check StructuredData with file of the Dataset
Dataset [12.3.1]	StructuredData with RDF metadarta file
Output	Prints a message if the file is not identical with the generated StructuredData.
Expected result	The StructuredData generated by the tool must be identical with the dataset file.

Variations	*<not applied>*

2.8.2.2 Wrap metadata

Name of test case	testMetadataWrapper
Initial conditions	RDF metadata file, StructuredData with RDF metadarta file
Description of testing functionality	This test case verifies proper parsing of RDF metadata with the MetadataWrapper and the generation of the StructuredDataobject
Tools involved	MetadataTool
Detailed steps	1. Instantiate MetadataWrapper 2. Read RDF metadata file 3. Generate StructuredData 4. Check StructuredData with file of the Dataset
Dataset	RDF metadata file, StructuredData with RDF metadarta file
Output	Prints a message if the file is not identical with the generated StructuredData.
Expected result	The StructuredData generated by the tool must be identical with the dataset file.
Variations	*<not applied>*

2.8.2.3 Request metadata

Name of test case	testMetadataRequester
Initial conditions	CDS should have the Wordnet ontology loaded
Description of testing functionality	This test case verifies the results returned by the tool and the CDS engine regarding ontology entity queries
Tools involved	Metadata Tool
Detailed steps	1. Instantiate MetadataRequester 2. Run multiple queries for ontology entities 1. Set prefix of entity label 2. Set entity type 3. Set ontologyURI 3. Check returned results with expected results
Dataset	<none>
Output	Prints a message if the query results returned by the tool are the expected results.
Expected result	The ontology entities returned as query results by the tool, must be identical to the expected results.
Variations	*<not applied>*

3 Building a CONVERGENCE Application

This chapter aims to provide application developers with the information necessary to develop and deploy a CONVERGENCE application and to initialize the middleware.

3.1 Developing the application

The CONVERGENCE middleware and tools are developed in the JAVA programming language and can be used by any JAVA application. The application developer should include the MXM and CONVERGENCE middleware binaries in the classpath and import the CONVERGENCE-tools library to access the tools. Before deploying the application, the developer prepares a valid MXMConfiguration file, which specifies the middleware engines to be loaded at runtime. More information on the MXMConfiguration file can be found in the deliverable 6.2 [3]. A sample configuration can be found in the ANNEX C – Sample MXM Configuration file. Examples of the functionality provided by specific tools are described in the chapters of this report, describing specific CONVERGENCE applications.

3.2 Deploying the application

The only requirement on the deployment environment for CONVERGENCE applications is that it should include the JAVA runtime environment and that the classpath should include the binaries for the CONVERGENCE middleware and tools. If the application is deployed in an application server, it is advisable to package the middleware and tools binaries inside the web archive and not in the server's library folder. This avoids potential classpath issues. The paragraphs below describe the fundamental steps that the application developer should perform to properly initialize the middleware, when starting up an application.

3.2.1 Middleware initialization

Before deploying the application, the developer should pack it with a valid MXMConfiguartion file, enabling CONVERGENCE to automatically load the middleware engines. The initialization of the middleware is performed by calling the singleton constructor and passing the MXMConfiguration file as a parameter.

```
MXM coMid = MXM.getInstance(new File(“MXMConfiguration.xml)”);
```

If the initiation is successful, the list of middleware engines is printed, as shown below.

```
INFO [main] (MXM.java:84)- ------- Multimedia Extensible Middleware ------
INFO [main] (MXM.java:85)- - Version 0.0.1
INFO [main] (MXMEngineLoader.java:88)- Loaded engine: MetadataTE
INFO [main] (MXMEngineLoader.java:88)- Loaded engine: RELTE
INFO [main] (MXMEngineLoader.java:88)- Loaded engine: SecurityTE
INFO [main] (MXMEngineLoader.java:88)- Loaded engine: EventReportTE
INFO [main] (MXMEngineLoader.java:113)- Loaded engine: IPMPTE
INFO [main] (MXM.java:145)- Middleware initialization successful
```

3.2.2 Orchestrator initialization

After the initialization of the middleware core and engines, it is necessary to configure the operation of the orchestrator engine. The orchestrator engine consists of various orchestrations that handle MXMEvents, support protocol engines, schedule tasks and build chains of engines. The orchestrator engine also configures the operation of technology engines, providing parameters that are not available in the MXMConfiguration file. The code fragment below, demonstrates the initialization of the Event Report TE, Match TE and Overlay TE repositories. If the Match TE repository is not empty, the “true” parameter will enforce a clean start. The ReceiverOrchestrator, responsible for handling MatchEventReport MXMEvents, is then booted, firing PubSubMatchEvents, as described in the next paragraph. Finally, the Overlay TE registries are scheduled to be updated every 30 seconds and every 10 seconds pending publications or subscriptions.

```
CONVERGENCEOrchestratorTE orchestrator = (CONVERGENCEOrchestratorTE)
MXM.getInstance().getDefaultEngine(MXMEngineName.Orchestrator.value());

// boot er, overlay, match, receiver
orchestrator.getEROrchestrator().bootEREngine();
orchestrator.getOverlayOrchestrator().bootOverlayEngine();
orchestrator.getMatchOrchestrator().bootMatchEngine(true);
orchestrator.getReceiverOrchestrator().bootReceiver();

// start overlay tasks
orchestrator.getTimerOrchestrator(OrchestratorTasks.GossipTask).startTaskEx
ecution(10000);
orchestrator.getTimerOrchestrator(OrchestratorTasks.RegistryAdvertisementTa
sk).startTaskExecution(30000);
```

3.2.3 Listener initialization

The last step that the application developer is required to perform is to implement a MXMeventListener to listen for PubSubMatchEvents. These events are fired by the middleware when there is a publication or subscription match. The developer can exploit the event to handle incoming notifications for matches and record matched publications and subscriptions. The code below registers the MatchListener for PubSubMatchEvents and prints a message when there has been a match.

```
public class MatchListener implements MXMEventListener
{
	public MatchListener()
	{
	MXM.getInstance().getMXMEventHandler().registerMXMEventListener(MatchingPu
blicationReceivedEvent.class.getName(), this);
	}

	public void mxmEventOccurred(MXMEvent event)
	{
		System.out.println("NOTIFICATION ARRIVED !!!!!!!!");
	}
}
```

4 Implementation of “Photos in the Cloud and Analyses on the Earth” (real world trial 1)

4.1 Brief description of use case

The purpose of this trial is to test whether CONVERGENCE can provide useful support for new business models in the photography business. The goal is to make it easier for photographers to contribute and describe photos, improve access for users and generally facilitate the management of relevant services. All photographs are represented by VDIs, each containing:

1. The photo itself (or a link to the photo) accompanied by a low-resolution version of the photo
2. Metadata describing the photo, including the date, time and place where the photo was taken, legal data on the author and owner of the photo, technical data about the photo (camera, lens, shutter time, aperture, ISO etc.), historical data about the site represented in the photo, and other metadata contributed by Alinari staff and third parties
3. Licensing information representing the conditions under which photos can be licensed by a photographer to Alinari and by Alinari to an end user. Licensing information is represented using the CONVERGENCE REL [5].

A detailed description of the functionalities supported in this use case scenario can be found in the corresponding sections of D7.1 and D7.2 [1][2].

4.2 Functional Description

This section describes the creation of the VDIs required by the application, with the help of the CONVERGENCE Tools described in Chapter 2.

4.2.1 Creation of Application VDIs

4.2.1.1 Photographer uploads Photo VDI to Alinari server

In this first application, a freelance photographer wants to create a new photo VDI. Each such VDI will contain metadata related either to the photo (e.g. tags, date of capture, info related to the analysis of the image, etc.) or to the photographer. Additionally, the photographer can use the License Tool to define rights for the photo. After successful registration, she launches the Create Photo VDI application. The procedure for creating a new R-VDI is described in the following steps:

1. Initialize a new RvdiCreator instance
   RvdiCreatorrvdiTool = VdiTool.newRvdiCreator();
2. Use the environment to fill in the appropriate metadata (tags)
   rvdiTool.addTag(tag);
3. Create an image reference to the resource VDI

Resource imageRef = rvdiTool.createResource(url, "image/" + photo.getImageFormat(), null);

4. Set an expiry date for the VDI

   rvdiTool.setExpiryDate(new Date());

5. Add asset to the resource VDI with its licenses (if they exist)

rvdiTool.addAsset(imageRef, List<License>, null);

6. Store the VDI

   rvdiTool.store(filePath);

7. Advertise the VDI

   rvdiTool.advertise(filePath);

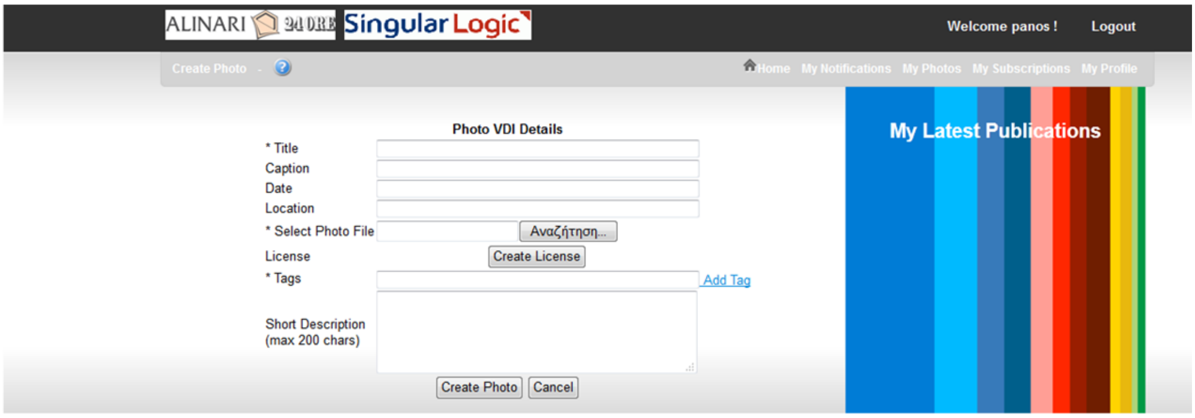


Figure 4-1: Creation of a Photo VDI in the Alinari Web Application

The procedure for creating a new REL license is described in the following steps:

1. For each PropertyPossessor generate a license
2. Create a new LicenseCreator instance

   LicenseCreatorlicenseCreator= LicenseTool.newLicenseCreator();
3. Create Everyone as a PropertyPossessor

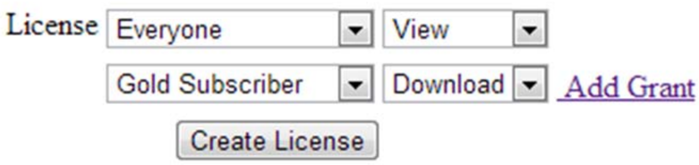
   PropertyPossessor EVERYONE = licenseCreator.createPropertyPossessor(“alinari:everyone”);
4. Set ForAllPropertyPossessors

   licenseCreator.setForAll(EVERYONE);
5. Add Grant

   licenseCreator.addGrant(RELRight.play, null);
6. Generate the license

   License license = licenseCreator.generateLicense();

Once a user clicks on the “Create Photo” option of the web environment shown in **Figure 4-1**, she can access the online form for creating a new photo VDI. At the same time, a new instance of the R-VDI Tool is generated (step 1). The user may then add all related metadata (step 2), upload the actual photo (step 3) and set an expiry data for the VDI (step 4). Once she clicks on the “Create Photo” button, the actual R-VDI is created and stored in CoNet (steps 5 and 6).



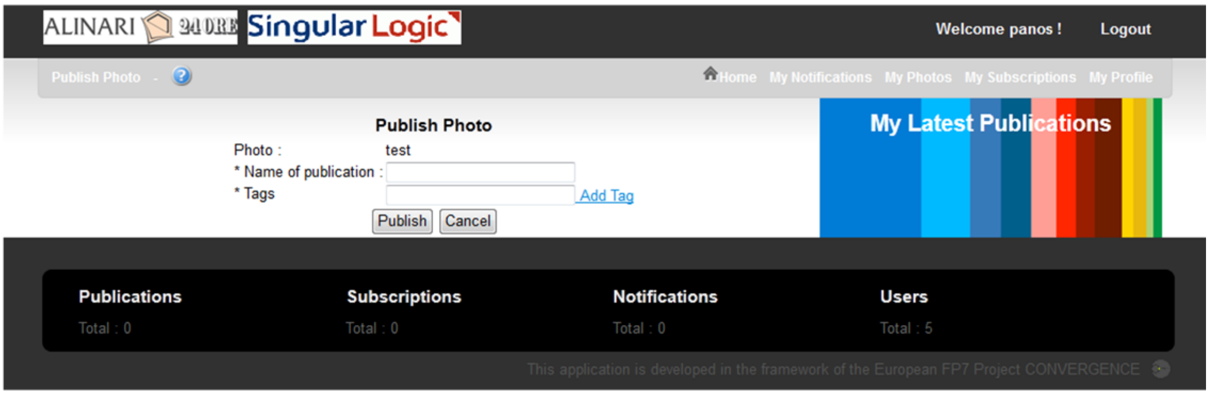
License: Everyone – View

Gold Subscriber – Download

4.2.1.2 Photographer publishes Photo VDI

In this second application, a photographer publishes a VDI. This involves the following steps:

1. Initialize a new instance of the PvdiCreator, based on the R-VDI identifier
   PvdiCreatorpubVdiTool = VdiTool.newPvdiCreator(photo.getR_VDI_Id());
2. Set an expiry date for the VDI
   pubVdiTool.setExpiryDate(finalDate.getTime());
3. Add the VDI to a specified fractal
   pubVdiTool.setFractal(fractal);
4. Generate and publish the VDI in the cloud
   pubVdiTool.generatePVDI();
   pubVdiTool.publish();



Publish Photo

Photo : test

* Name of publication :

* Tags

Publications Total : 0

Subscriptions Total : 0

Notifications Total : 0

Users Total : 5

This application is developed in the framework of the European FP7 Project CONVERGENCE

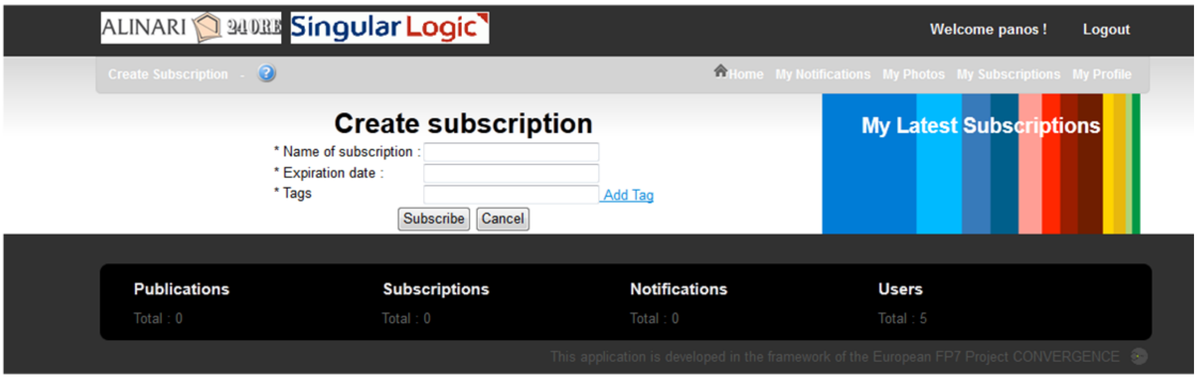
Figure 4-2: Creation of publication in the Alinari Web Application

When the user selects the “Publish Photo” from the main menu of the application (see **Figure 4-2**), a new instance of the Publication Tool is initialized (Step 1). Note that this new VDI is related to the original R-VDI created in the previous step. The user can then add metadata as and an expiration date to the VDI. Once she clicks on the Publish button, the P-VDI is created and published in CONVERGENCE (step 4).

4.2.1.3 Alinari Photo Archive Manager subscribes to and downloads photos

The third application allows users to search for specific content by injecting S-VDIs into the cloud. This involves the following steps.

1. Initialize a new instance of the SvdiCreatoras and a new event report
   SvdiCreatorsvdiTool = VdiTool.newSviCreator();
   ERR err = EventReportTool.getInstance().createERR(EventTriggerType.MATCH);
2. Set a fractal for the S-VDI
   svdiTool.setFractal(request.getParameter("fractal"));
3. Set an expiry date to the S-VDI
   svdiTool.setExpiryDate(finalDate.getTime());
4. Add the keywords and search tags specified by the user
   svdiTool.addKeywordCondition(tags);
5. Add an event report (an event is triggered on every match)
   svdiTool.addERR(err);
6. Publish the S-VDI
   svdiTool.generateSVDI();
   String svdi_id = svdiTool.publish();



Create subscription

* Name of subscription :
* Expiration date :
* Tags

Publications
Total : 0

Subscriptions
Total : 0

Notifications
Total : 0

Users
Total : 5

This application is developed in the framework of the European FP7 Project CONVERGENCE

Figure 4-3: Creation of subscription in the Alinari Web Application

The user selects “Create Subscription” from the main menu of the application (see **Figure 4-3**) initializing a new instance of the Subscription Tool (Step 1). The user can then add metadata and an expiration date to the VDI. When she clicks on the Subscribe button, the S-VDI is created and published in CONVERGENCE (Step 6).

5 Specifications and Technical Implementation of "Videos in the Cloud and Analyses on the Earth" (real world trial 2)

5.1 Brief description of use case

The "Videos on the Cloud and Analyses on the Earth" scenario has been designed to improve the management of audiovisual archives and to exploit the potential of semantic techniques, when the same video resources are exploited several times in different contexts of use (analyses using different domain ontologies, postings on different video channels).

This scenario involves the following users.

1. Video Material Owners (VMOs) provide videos to the archive. They encrypt videos, upload them on CoNet and advertise the existence of the videos to specific analysts and video distributors. They want to be notified when their videos are manipulated.
2. Video Distributors (VDs) are notified of new videos, download and decrypt them, then make the videos available for streaming.
3. Analysts describe and interpret the content of a video resource, i.e. through the analysis of audiovisual topics (or subjects), the analysis of visual and/or acoustic frames, the linguistic and cultural adaptation of source videos to a chosen (French-speaking) target public, etc. They subscribe to videos, download them from CoNet and decrypt them. They upload their analyses on CoNet and notify certain VCOs of the existence of their analyses. They want to be notified when their analyses are manipulated.
4. Video Channel Owners (VCOs) are responsible for video channels, and manage content posted on their channel. They subscribe to analyses and post them on their channel. They want to notify certain VCUs and CHs of these posts.
5. Channel Holders (CHs) are notified of new channels and new posts on channels, update the channels and make them available on the Internet.
6. Video Channel Users (VCUs) explore the content of Channels, subscribe to new posts on channels and browse channels.

5.2 Functional Description

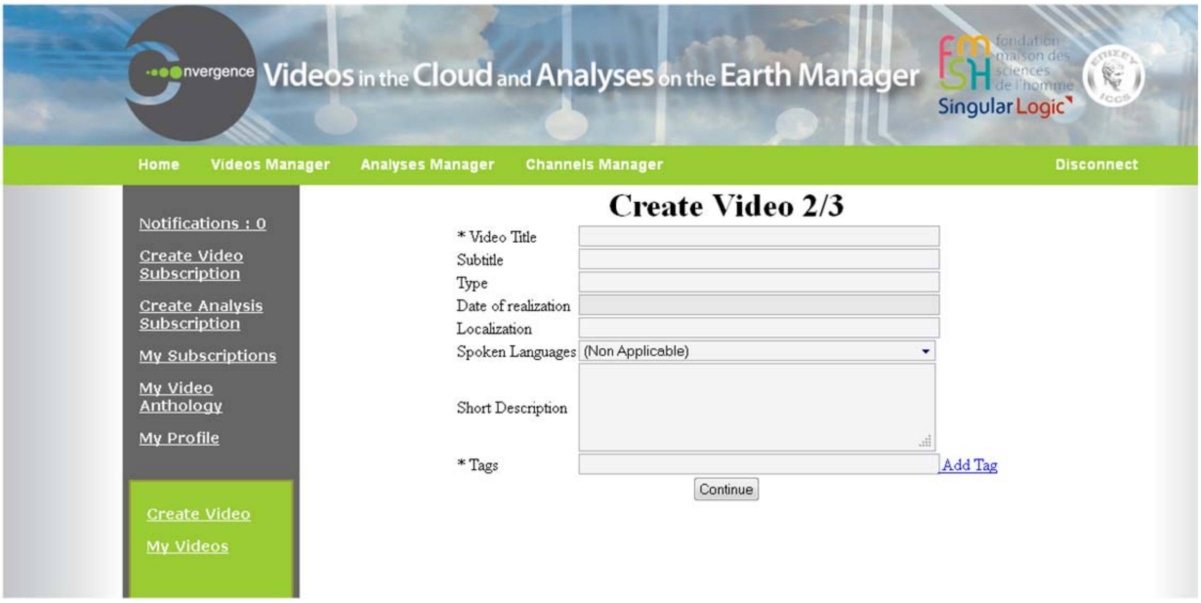
5.2.1 *VMO creates a Video VDI*

In the first application, a VMO wants to create a new Video VDI. Each such VDI will contain metadata related either to the video (e.g. title, tags, date of capture, short description, authors, etc.) or to the producer. After successful registration, he/she launches the Create Video VDI application. The procedure for creating a new R-VDI consists of the following steps:

1. Initialize a new RvdiCreator instance
2. Using the environment, the user fills the appropriate metadata (tags entry)
3. Create a video reference to the resource VDI
4. Add the resource with its licenses (if they exist)
5. Set an expiry data for the VDI
6. Store the VDI

7. Advertise the VDI

The code is identical to the code for the Alinari scenario described in section 4.2.1.1.



Create Video 2/3

* Video Title

Subtitle

Type

Date of realization

Localization

Spoken Languages (Non Applicable)

Short Description

* Tags

Figure 5-1: Creation of Video VDI in the FMSH Web Application

5.2.2 VMO publishes a Video VDI

In the second application, a VMO creates and publishes a VDI, in the following steps:

1. Initialize a new instance of the Publication VDI Tool, based on the R-VDI identifier
2. Set an expiry date for the VDI
3. Add the VDI to a specified fractal
4. Publish the VDI in the cloud

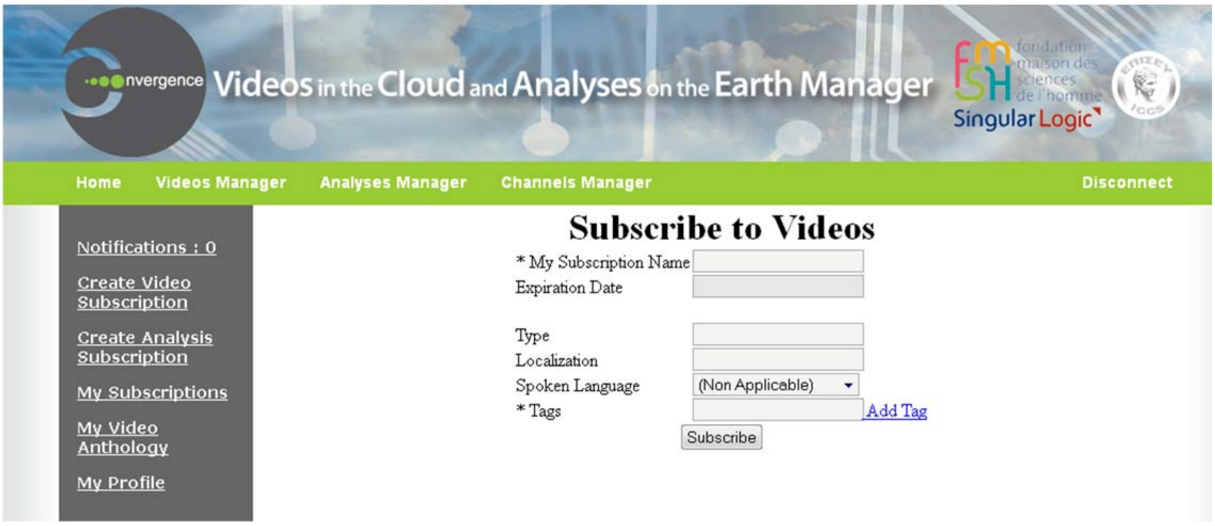
The code is identical to the code for the Alinari scenario described in section 4.2.1.2 (Photographer publishes Photo VDI).

5.2.3 Analyst subscribes to a Video VDI

In the third application, an Analyst wants to create an S-VDI and inject it into the cloud. This involves the following steps:

1. Initialize a new instance of the SvdiCreator
2. Set a fractal for the S-VDI
3. Set an expiry date for the S-VDI
4. Add the keywords and search tags specified by the user
5. Add an event report (an event is triggered on every match)
6. Generate the S-VDI
7. *Inject the S-VDI*

The code is identical to the code for the Alinari scenario described in section 4.2.1.3 (Alinari Photo Archive Manager subscribes to and downloads photos).


Convergence Videos in the Cloud and Analyses on the Earth Manager
fondation maison des sciences de l'homme
SingularLogic
Home
Videos Manager
Analyses Manager
Channels Manager
Disconnect
Notifications : 0
Create Video Subscription
Create Analysis Subscription
My Subscriptions
My Video Anthology
My Profile
Add Tag
Subscribe

Subscribe to Videos

* My Subscription Name

Expiration Date

Type

Localization

Spoken Language (Non Applicable)

* Tags

Figure 5-2: Creation of Subscription VDI in the FMSH Web Application

5.2.4 Analyst creates an Analysis VDI

This application creates and publishes an Analysis of a Video. This involves the following steps:

1. User adds the RDF analysis of the video
2. A new MetadataWrapper instance is created
   MetadataWrappermetaWrapper = MetadataTool.newMetadataWrapper();
3. MetadataWrapper parses the metadata file.
   metaWrapper.parseRDF(rdfFile);
4. A StructuredData object is created.
   StructuredDatasmetadataObject = metaWrapper.generateStructuredData();
5. A new RvdiCreator instance is initialized.
6. Set as expiry date the expiration date of the video
7. Add license if the user has created one.
8. Generate AnalysisVDI
9. Advertise the AnalysisVDI
10. A new PvdiCreator instance is initialized for the AnalysisVDI.
11. Set as fractal the ANALYSIS_FRACTAL.
12. Inject Analysis VDI to the ANALYSIS_FRACTAL

The code is identical to the code for the Alinari scenario described in section 4.2.1.1.

5.2.5 Analyst publishes an Analysis VDI

This application publishes an Analysis VDI, in the following steps:

1. Initialize a new instance of the PvdiCreator, based on the Analysis R-VDI identifier
2. Set an expiry date for the VDI
3. Add the VDI to the ANALYSIS_FRACTAL
4. Publish the VDI in the cloud

The code is identical to the code for the Alinari scenario described in section 4.2.1.3 (Alinari Photo Archive Manager subscribes to and downloads photos).

5.2.6 VCU subscribes to an Analysis VDI

This application creates a subscription for Analyses and injects it into the cloud. This involves the following steps:

1. Initialize a new instance of the SvdiCreator.
2. Set the ANALYSIS_FRACTAL as fractal for the S-VDI
3. Set an expiry date
4. For each submitted object property add to the SPARQL subscription a triple condition

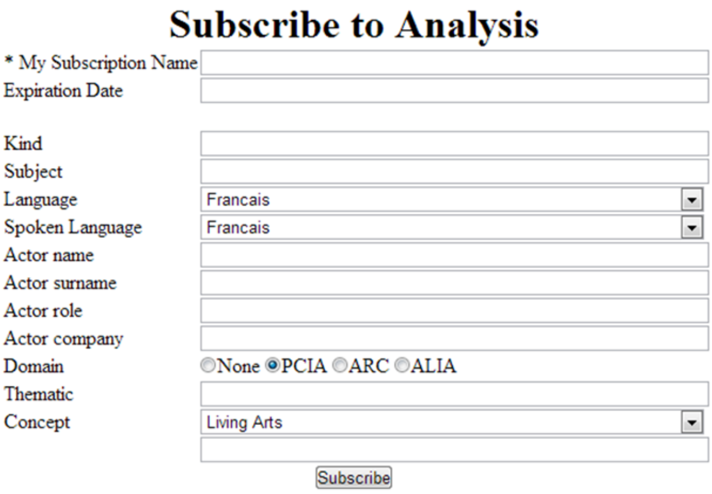
svdiCreator.addSparqlTripleCondition(subject, field/predicateURI, entityURI);

5. For each submitted datatype property add to the SPARQL subscription a literal triple condition

svdiCreator.addSparqlLiteralCondition(subject,field/predicateURI, value);

6. Generate the S-VDI
7. Inject the S-VDI

The code for steps 1-3 and 6-7 is identical to the code for the Alinari scenario described in section 4.2.1.3 (Alinari Photo Archive Manager subscribes to and downloads photos).



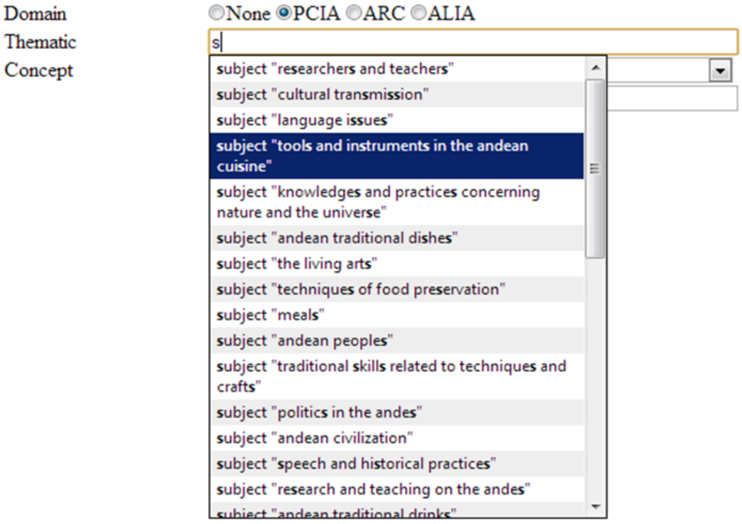
Subscribe to Analysis

* My Subscription Name	
Expiration Date	
Kind	
Subject	
Language	Francais
Spoken Language	Francais
Actor name	
Actor surname	
Actor role	
Actor company	
Domain	○None ●PCIA ○ARC ○ALIA
Thematic	
Concept	Living Arts

Figure 5-3: Subscription to Analysis form

When a user wishes to fill in object properties, she uses the Metadata Tool. More specifically, she uses the MetadataRequester to query the ontologies loaded in the CDS TE for ontology entities (see also ANNEX A).

1. A new instance of the MetadataRequester is instantiated
   MetdataRequestermetaRequester = MetadataTool.newMetadataRequester();
2. User selects the ontology model e.g. PCIA, ALIA
3. User types in the field a prefix of the requested entity's label
   metaRequester.requestMetadata(prefix, type, ontologyURI);
4. A popup window with the results is displayed to the user.
5. The label of the entity is copied in the field and its URI to a hidden field.



Domain ○None ●PCIA ○ARC ○ALIA

Thematic s

Concept

subject "researchers and teachers"
subject "cultural transmission"
subject "language issues"
subject "tools and instruments in the andean cuisine"
subject "knowledges and practices concerning nature and the universe"
subject "andean traditional dishes"
subject "the living arts"
subject "techniques of food preservation"
subject "meals"
subject "andean peoples"
subject "traditional skills related to techniques and crafts"
subject "politics in the andes"
subject "andean civilization"
subject "speech and historical practices"
subject "research and teaching on the andes"
subject "andean traditional drinks"

Figure 5-4: Using ontologies in the FMSH application

6 Technical Implementation of “Augmented Lecture Podcast (ALP)” (real world trial 3)

6.1 Brief description of use case

The augmented lecture podcast scenario describes a social learning environment that augments traditional podcasts with interactive features. Students watch and annotate podcasts, sharing and creating new knowledge. CONVERGENCE offers a number of advantages over current systems.

- Content is automatically updated as soon as changes occur. This ensures the quality of learning
- Content caching offers a more reliable service
- The CONVERGENCE Standard guarantees inter-operability.

At the same time, CONVERGENCE completely changes the way applications are developed.

To demonstrate these features, we implemented two applications – the podcast creator application and the augmented podcast service. The podcast creator is used by lecturers to create and publish podcasts; the augmented podcast service primarily targets students. **Figure 6-1** provides an overview of the way the two applications are used.

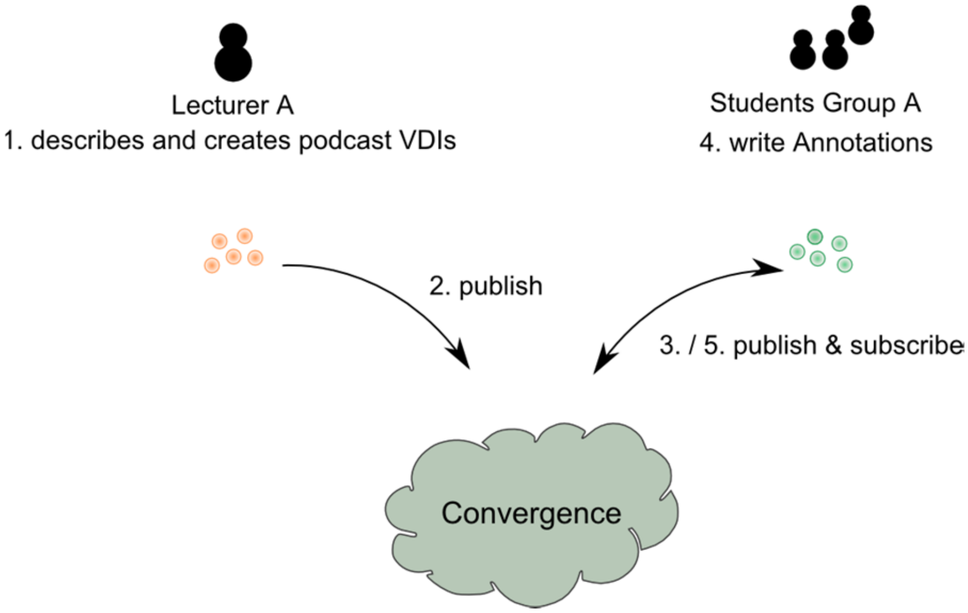
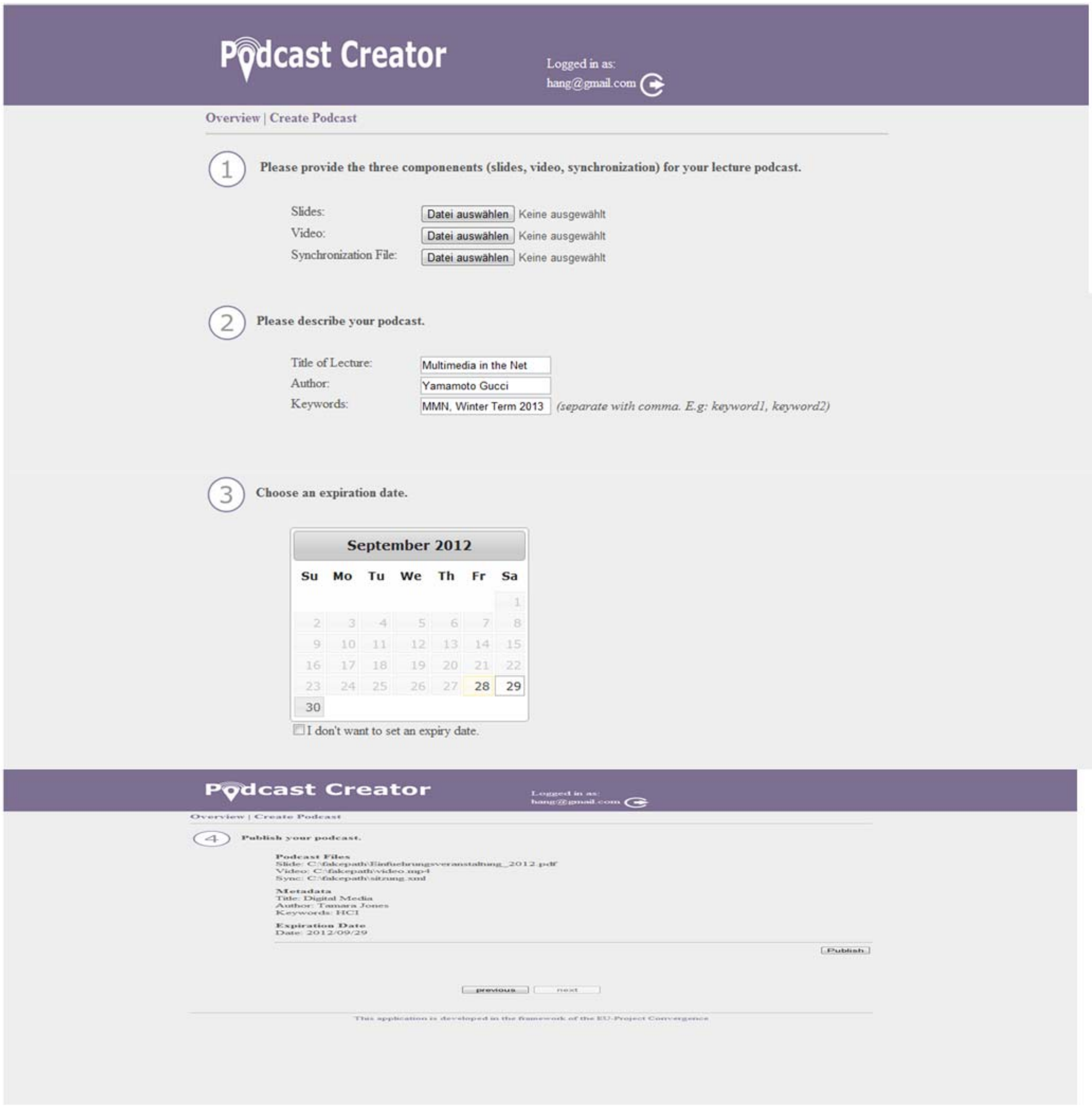


Figure 6-1: Workflow between the podcast creator application and the augmented lecture podcast service

6.2 Creation of Application VDIs

6.2.1 Create Resource-VDI& Create Publication VDI

Figure *6-2* provides an overview of the podcast creation process. The web interface enables lecturers to define podcast resources, describe them and set expiration dates. Once the publish button is selected, the CONVERGENCE process begins. This consists of the creation of a podcast component VDI, podcast VDI or annotation VDI and involves the steps described in the example below.



1 Please provide the three componenents (slides, video, synchronization) for your lecture podcast.

Slides: Datei auswählen Keine ausgewählt
Video: Datei auswählen Keine ausgewählt
Synchronization File: Datei auswählen Keine ausgewählt

2 Please describe your podcast.

Title of Lecture: Multimedia in the Net
Author: Yamamoto Gucci
Keywords: MMN, Winter Term 2013 *(separate with comma. E.g: keyword1, keyword2)*

3 Choose an expiration date.

September 2012

Su	Mo	Tu	We	Th	Fr	Sa
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

□ I don't want to set an expiry date.

4 Publish your podcast.

Podcast Files
Slide: C:\fakepath\Einfuehrungsveranstaltung_2012.pdf
Video: C:\fakepath\video.mp4
Sync: C:\fakepath\sitzung.xml

Metadata
Title: Digital Media
Author: Tamara Jones
Keywords: HCI

Expiration Date
Date: 2012/09/29

Publish
previous
next
This application is developed in the framework of the EU-Project Convergence

Figure 6-2: Screenshots of the podcast creator application with the different steps for VDI creation and publication

In step 1, the software instantiates the RvdiCreator and creates a Resource-VDI using the resources selected by the user and adds the corresponding metadata. The creation of a Resource-VDI involves the following steps:

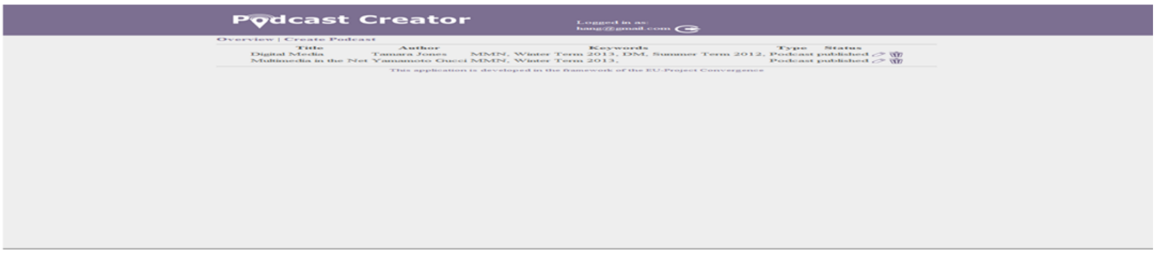
1. Instantiate the RvdiCreator using the VdiTool:
   RvdiCreatorrvdiTool = VdiTool.newRvdiCreator();

2. Create the resource and add it to the VDI:
   Resource resource = rvdiTool.createResource(path,“podcast”, null);
   rvdiTool.addAsset(resource, null, null);
3. Set an expiration date:
   rvdiTool.setExpiryDate(“26/09/2013”);
4. Add metadata to the resource:
   rvdiTool.addKeyword(“HCI”);
5. Generate the resource VDI:
   longvdiId = rvdiTool.generateRVDI();

In the second step, the Resource-VDI is published using a Publication-VDI. Publication involves the following steps:

1. Instantiate the PvdiCreator using the VdiTool:
   PvdiCreatorpubVdiTool = VdiTool.newPvdiCreator(vdiId);
2. Set an expiration date:
   pubVdiTool.setExpiryDate(“26/09/2013”);
3. Choose a fractal for the publication
   pubVdiTool.setFractal(“podcasts”);
4. Add metadata to the publication:
   pubVdiTool.addKeyword(“HCI”);
5. Create the publication VDI
   pubVdiTool.generatePVDI();
6. Publish the publication VDI:
   pubVdiTool.publish();

As soon as the podcast is published (i.e., the corresponding VDIs are successfully published), it is added to the overview of all published podcasts (see **Figure 6-3**). Lecturers can edit or delete podcasts by selecting the corresponding menu item. When a podcast is edited, the software creates a new VDI (as previously described) and deletes the old VDI (this feature has not yet been implemented). All this is transparent to the user. When the user deletes the podcast, the corresponding VDI is revoked and disappears from the CONVERGENCE network.



Podcast Creator

Logged in as:
hang@gmail.com

Overview | Create Podcast

Title	Author	Keywords	Type	Status
Digital Media	Tamara Jones	MMN, Winter Term 2013, DM, Summer Term 2012,	Podcast	published
Multimedia in the Net	Yamamoto Gucci	MMN, Winter Term 2013,	Podcast	published

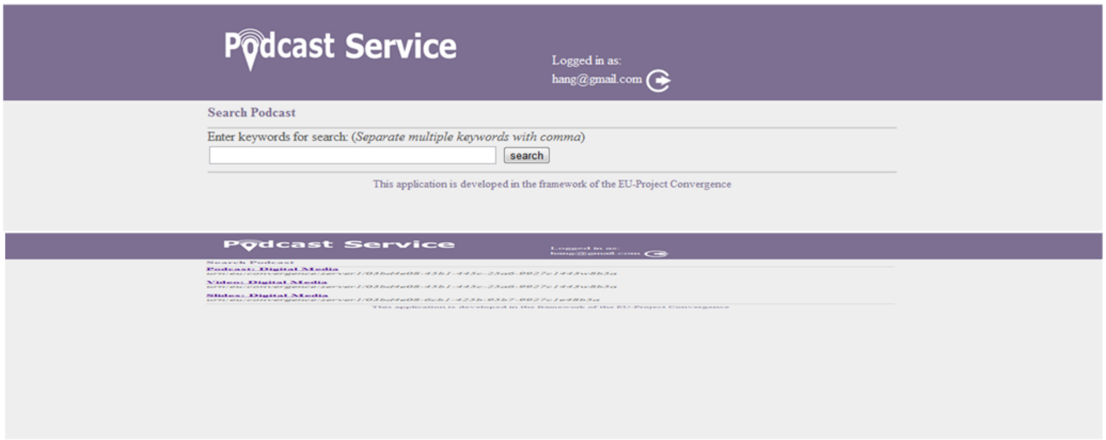
This application is developed in the framework of the EU-Project Convergence

Figure 6-3: Overview of all published podcast (with options to update and delete podcasts)

6.2.2 Create Subscription VDI

Once a podcast is published to the CONVERGENCE network, students can search for it by entering keywords in the web interface of the augmented lecture podcast service (see **Figure**

6-4). In the background, the software will use the entered keywords to create a subscription VDI, allowing CONVERGENCE to perform matches. All matches for the subscription will be displayed as results on the web interface.



Logged in as:
hang@gmail.com

Search Podcast

Enter keywords for search: (*Separate multiple keywords with comma*)

This application is developed in the framework of the EU-Project Convergence

Logged in as:
hang@gmail.com

Search Podcast

Podcast: Digital Media
urn/eu/convergence/server1/03bd4e08-43b1-443c-25a6-9927c1443w8b3a

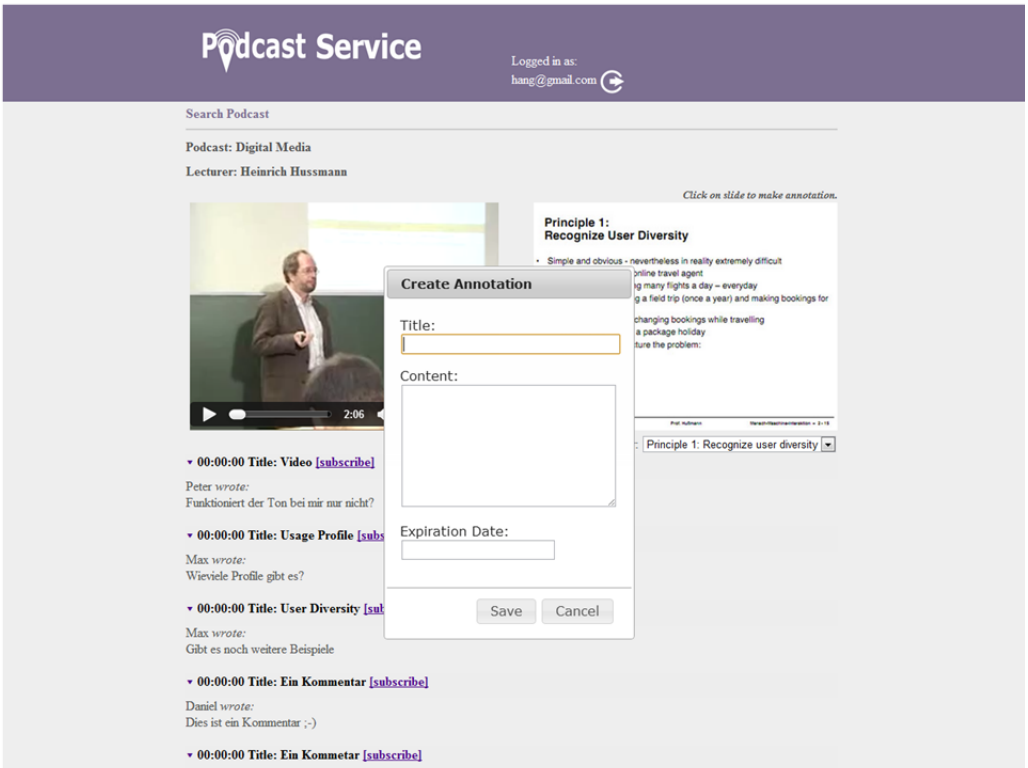
Video: Digital Media
urn/eu/convergence/server1/03bd4e08-43b1-443c-25a6-9927c1443w8b3a

Slides: Digital Media
urn/eu/convergence/server1/03bd4e08-6cb1-423b-93b7-9927c1a48b3a

This application is developed in the framework of the EU-Project Convergence

Figure 6-4: Screenshots of the augmented podcast service. The screenshots shows the search functionality with a list of results (example for subscription)

Selecting one of the results leads the user to **Figure *6-5*,Figure *6-5*:** Overview of an augmented lecture podcast. Popup is used to create and publish an annotation VDI. where she can annotate the podcast. Annotation is also based on a publish-subscribe mechanism.



Logged in as:
hang@gmail.com

Search Podcast

Podcast: Digital Media

Lecturer: Heinrich Hussmann

Click on slide to make annotation.

Principle 1:
Recognize User Diversity

- Simple and obvious - nevertheless in reality extremely difficult
- nline travel agent
- g many flights a day – everyday
- g a field trip (once a year) and making bookings for
- changing bookings while travelling
- a package holiday
- ture the problem:

Principle 1: Recognize user diversity

Create Annotation

Title:

Content:

Expiration Date:

▾ 00:00:00 Title: Video [subscribe]

Peter *wrote:*
Funktioniert der Ton bei mir nur nicht?

▾ 00:00:00 Title: Usage Profile [subs

Max *wrote:*
Wieviele Profile gibt es?

▾ 00:00:00 Title: User Diversity [sub

Max *wrote:*
Gibt es noch weitere Beispiele

▾ 00:00:00 Title: Ein Kommentar [subscribe]

Daniel *wrote:*
Dies ist ein Kommentar ;-)

▾ 00:00:00 Title: Ein Kommetar [subscribe]

Figure 6-5: Overview of an augmented lecture podcast. Popup is used to create and publish an annotation VDI.

Publication works as already described. The subscription process involves the following steps:

1. Instantiate Event Report tool:
   ERR e = EventReportTool.getInstance().createERR(
   EventTriggerType.MATCH);
2. Instantiate SvdiCreator using the VdiTool
   SvdiCreatorsvdiTool = VdiTool.newSvdiCreator();
3. Choose a fractal for the subscription
   svdiTool.setFractal("podcasts");
4. Set an expiration date:
   svdiTool.setExpiryDate("28/09/2013");
5. Add keywords to the subscription:
   svdiTool.addKeywordCondition(„HCI");
6. Add the event report to the subscription:
   svdiTool.addERR(e);
7. Generatet he subscription VDI and subscribe:
   svdiTool.generateSVDI();
   svdi_id = svdiTool.subscribe();

7 Technical Implementation of Smart Retailing (real world trial 4)

7.1 Brief description of use case

This scenario designates a “smart retailing” supply chain for consumer electronics products. The users and beneficiaries are Manufacturers, Retailers and Consumers.

1. Manufacturers
    a. Create, store, advertise and publish Product Type VDIs containing reliable information such as the product name, description, physical characteristics, technical features and resources
2. Retailers
    a. Subscribe to Product Type VDIs published by Manufacturers
    b. Receive notifications about updates on Product Type VDIs they have subscribed to
    c. Create and publish Retailers’ Product Type VDIs, augmenting Manufacturers’ Product Type VDIs with information about Retailer promotions, special offers, sales, etc.
    d. Create Product Instance VDIs when selling products to Consumers, adding new data such as consumer information, serial number and warranty details to Product Type VDIs.
3. Consumers
    a. Browse and subscribe to product VDIs
    b. Receive notifications about updates on product VDIs they have subscribed to
    c. Consult reliable information about products

7.2 Functional Description

The scenario involves the following functionalities:

1. Manufacturer
    i. Create and Store Product Type VDI
    ii. Create and inject Publication VDI for the Product Type VDI
    iii. Revoke Publication VDI from the CONVERGENCE Cloud
2. Retailer
    i. Create and inject Subscription VDI for Manufacturer’s Product Type VDI
    ii. Download (extract metadata and resources) Manufacturer’s Product Type VDI
    iii. Create and inject Publication VDI about promotions, special offers on Product Type VDIs
    iv. Process Product Type VDI to Create and Store Product Instance VDI (adding new information)

v. Revoke Publication VDI from the CONVERGENCE Cloud

3. Customer

   i. Create and inject Subscription VDI

   ii. Browse products VDIs

7.3 Creation of Application VDIs

7.3.1 Manufacturer Creates and Stores Product Type VDI (R-VDI)

In this method, the application is used to create and store a Product Type VDI containing product information (metadata) and resources, such as photos or user manuals. After the Product Type VDI is created, the application advertises it, interacting with the middleware. This involves the following steps:

1. Initialize a new Resource VDI instance:

   RvdiCreatorrvdiTool = VdiTool.newRvdiCreator();

2. Using the web application the user fill the “Create Product” form with the product information that will be parsed as metadata:

   rvdiTool.addFieldValue("PRODUCT_NAME",value);

   rvdiTool.addKeyword(keyword);

   rvdiTool.setStartDate(new Date());

3. Add files (url, mime type) to the Resource VDI:

   Resource fileRef = rvdiTool.createResource(url_resource, "image/jpeg", null);

4. Add asset to the files with its licenses (if they exist)

   rvdiTool.addAsset(fileRef, List<License>, null);

5. Generate the Product Type R-VDI

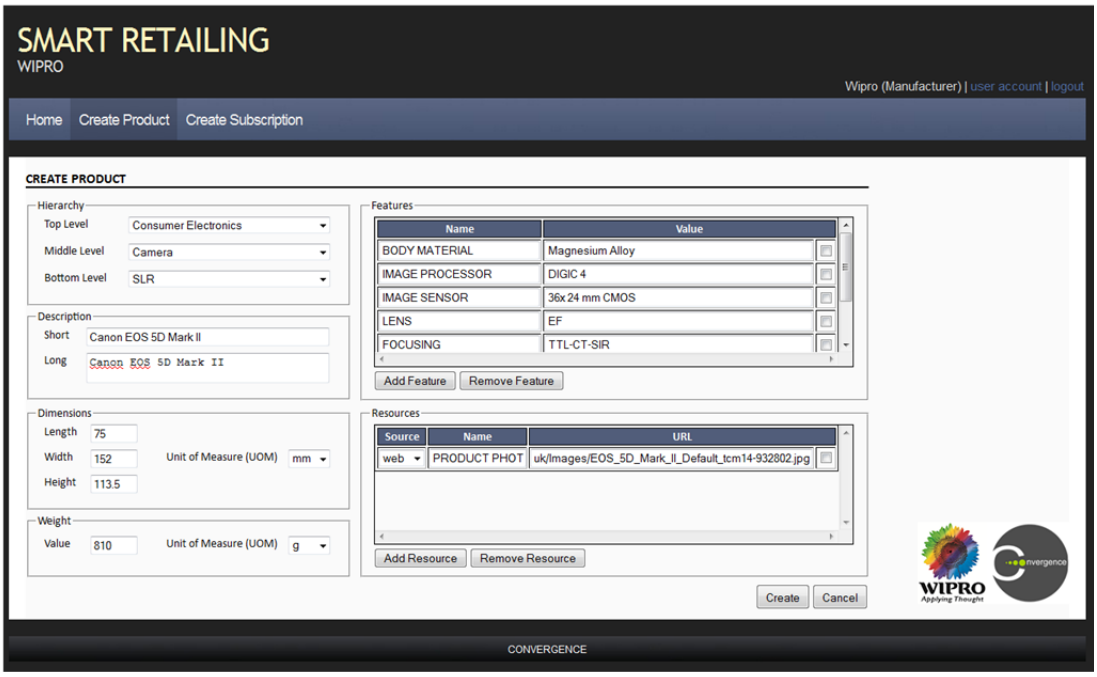
   rvdiTool.generateRVDI();

6. Store the Product Type R-VDI

   rvdiTool.store(filePath);

7. Advertise the Product Type R-VDI

   rvdiTool.advertise(filePath);



CREATE PRODUCT

Hierarchy

Top Level	Consumer Electronics
Middle Level	Camera
Bottom Level	SLR

Description

Short	Canon EOS 5D Mark II
Long	Canon EOS 5D Mark II

Dimensions

Length	75		
Width	152	Unit of Measure (UOM)	mm
Height	113.5		

Weight

Value	810	Unit of Measure (UOM)	g

Features

Name	Value	
BODY MATERIAL	Magnesium Alloy	□
IMAGE PROCESSOR	DIGIC 4	□
IMAGE SENSOR	36x 24 mm CMOS	□
LENS	EF	□
FOCUSING	TTL-CT-SIR	□

Resources

Source	Name	URL	
web	PRODUCT PHOT	uk/Images/EOS_5D_Mark_II_Default_tcm14-932802.jpg	□

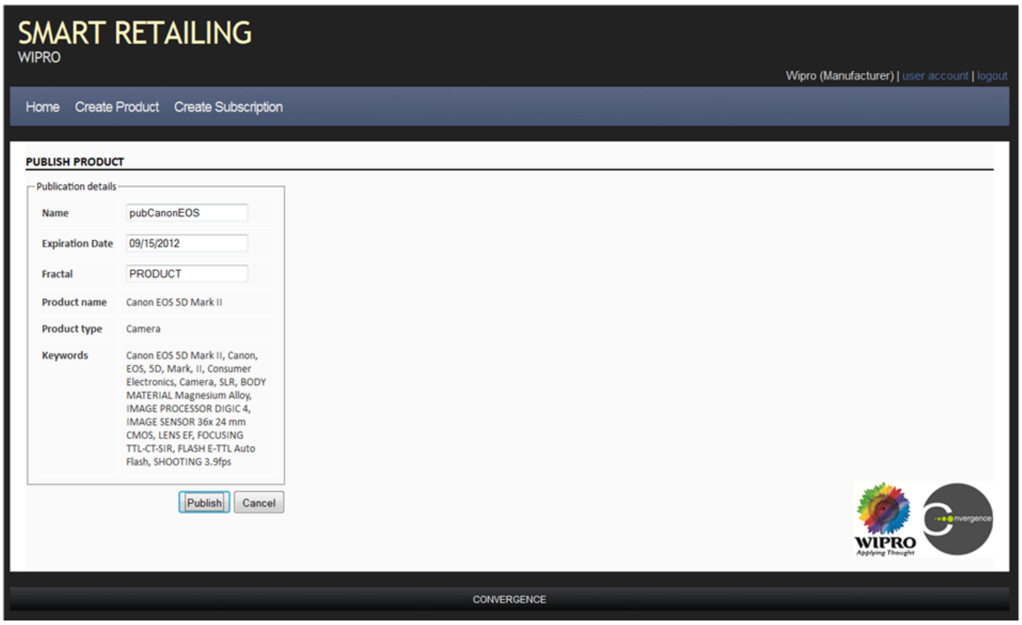
Figure 7-1: Web application "Create Product" form

Once the user clicks on the “Create Product” option of the web application (see **Figure *7-1***), the create product form appears. At the same time, a new instance of the R-VDI Tool is generated (step 1). The user can then add product information (e.g. description, dimensions, features) (step 2) and upload images and/or documents as resources (step 3). Once she clicks on the “Create” button, the actual R-VDI is created, stored and advertised in CoNet (steps 5, 6 and 7).

7.3.2 Manufacturer Publishes Product Type VDI (P-VDI)

In this method the application is used to publish a VDI, associating a fractal and an expiration date with the publication. This involves the following steps:

1. Initialize a new Publication VDI instance:
   PvdiCreatorpvdiTool = VdiTool.newPvdiCreator(product.getR_VDI_Id());
2. Using the web application the user fill the “Publish Product” form with the fractal and expiration date of the publication which will be added to the Publication VDI instance:
   pvdiTool.setFractal(fractal);
   pvdiTool.setExpiryDate(finalDate.getTime());
3. Generate the P-VDI:
   pvdiTool.generatePVDI();
4. Publish the P-VDI:
   pvdiTool.publish();



PUBLISH PRODUCT

Publication details

Name	pubCanonEOS
Expiration Date	09/15/2012
Fractal	PRODUCT
Product name	Canon EOS 5D Mark II
Product type	Camera
Keywords	Canon EOS 5D Mark II, Canon, EOS, 5D, Mark, II, Consumer Electronics, Camera, SLR, BODY MATERIAL Magnesium Alloy, IMAGE PROCESSOR DIGIC 4, IMAGE SENSOR 36x 24 mm CMOS, LENS EF, FOCUSING TTL-CT-SIR, FLASH E-TTL Auto Flash, SHOOTING 3.9fps

Figure 7-2: Web application "Publish Product" form

Once the user clicks on the "Publish" button (see **Figure** *7-2*), the publish product form appears. At the same time, a new instance of the P-VDI Tool is generated taking the R-VDI id as a parameter (step 1). The user can then add publication data (e.g. name, fractal and expiration date) (step 2). Once she clicks on the "Publish" button, the actual P-VDI is created and published in CoNet (steps 3 and 4).

7.3.3 Manufacturer Subscribes to Product Type VDI (S-VDI)

Here, the application is used to subscribe to VDIs, meeting specific search criteria. This involves the following steps:

1. Initialize a new Subscription VDI instance:

   SvdiCreatorsvdiTool = VdiTool.newSvdiCreator();
2. Using the web application the user fill the "Create Subscription" form with the fractal and expiration date of the subscription which will be added to the Subscription VDI instance:

   svdiTool.setFractal(fractal);

   svdiTool.setExpiryDate(finalDate.getTime());
3. Add condition(s) to the Subscription VDI:

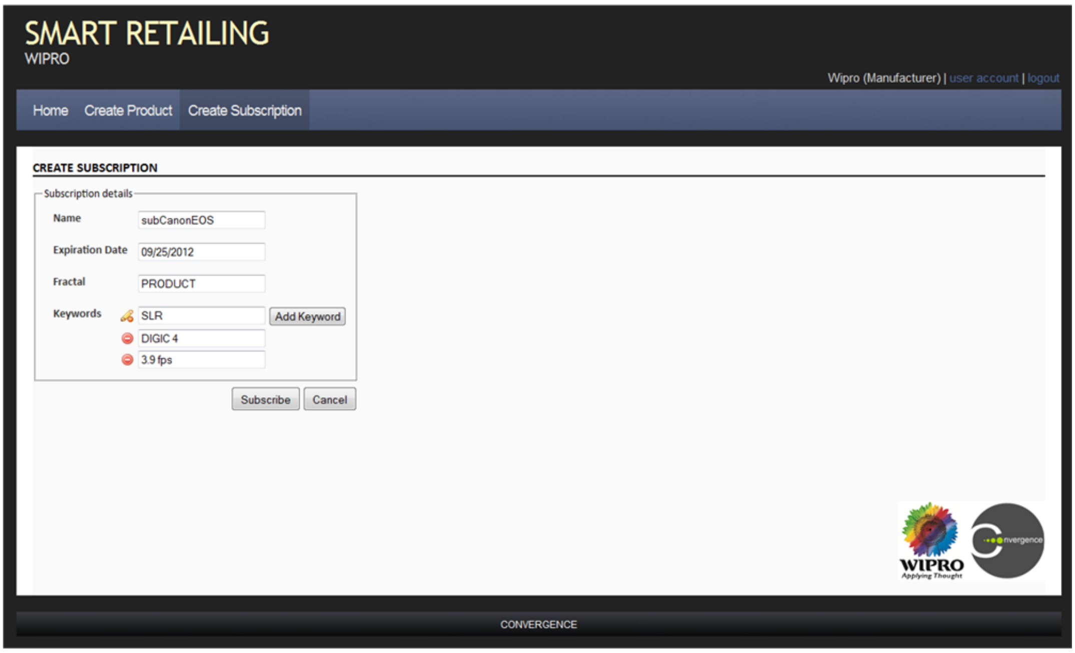
   svdiTool.addKeywordCondition(keywords);
4. Add event report to the Subscription VDI:

   ERR err = EventReportTool.getInstance().createERR(EventTriggerType.MATCH);

   svdiTool.addERR(err);
5. Generate the S-VDI:

   svdiTool.generateSVDI();
6. Publish the S-VDI:

   svdiTool.subscribe();



Name	subCanonEOS
Expiration Date	09/25/2012
Fractal	PRODUCT
Keywords	SLR DIGIC 4 3.9 fps

Figure 7-3: Web application "Create Subscription" form

Once the user clicks on the “Create Subscription” option of the web application (see **Figure 7-3**) the create subscription form appears. At the same time, a new instance of the S-VDI Tool is generated (step 1). The user can then add subscription data (e.g. name, fractal, expiration date (step 2). She can also add keywords (step 3). Once she clicks on the “Subscribe” button, the actual S-VDI is generated and published in CoNet (steps 4 and 5)

7.3.4 Retailer Subscribes to Product Type VDI (S-VDI)

A Retailer wants to search for product type VDIs with certain features or characteristics. To do so, she uses the application to subscribe to all VDIs satisfying certain search criteria. This involves the following steps:

1. Initialize a new Subscription VDI instance

   SvdiCreatorsvdiTool = VdiTool.newSvdiCreator();

2. Using the web application retailers fill the Subscribe to Product form with the fractal and expiration date of the publication which will be added to the Subscription VDI variable

   svdiTool.setFractal(fractal);

   svdiTool.setStartDate(startDate);

   svdiTool.setExpiryDate(expireDate);

3. Add condition to the Subscription VDI

   svdiTool.addFieldValueCondition("field", "=", "value");

   svdiTool.addKeywordCondition(keywords);

   svdiTool.addTagCondition(tags);

4. Add event report to the Subscription VDI

   svdiTool.addERR(e);

5. Publish the S-VDI

svdiTool.generateSVDI();
svdi_id = svdiTool.subscribe();

7.3.5 Retailer Creates Product Type VDI (S-VDI)

This scenario involves the POS machine used by the Retailer when selling a product to a Customer. The machine identifies the product that is being sold to the Customer and gets the product type VDI associated with the product. Then the Retailer inputs information about the product serial number and warranty detail. The Customer presents her personal information to the Retailer, either by inputting it manually into the POS or by presenting his/her loyalty card. This information now becomes metadata in the new product instance VDI, representing the product sold to the Customer. Finally, the system creates a link between the product type VDI and the product instance VDI that has just been created. The process involves the following steps.

1. Initialize a new Resource VDI Tool

   RvdiCreatorrvdiTool = VdiTool.newRvdiCreator();
2. Read the product type Resource VDI instance and get its metadata and then initialize a new Resource VDI instance

   RvdiParserrvdiParser = VdiTool.newRvdiParser(rvdi_id);

   Map<String, Object> metadata = rvdiParser.getFieldValueMap();
3. Add the metadata of step 2

   rvdiTool.addFieldValue(metadataKey, metadataValue);
4. Using the POS client application retailers input the product instance serial numberand the customer information that will be used as metadata (tags entry)

   rvdiTool.addTag(tagName);
5. Add the digital receipt file and the warranty details file (url, mimetype) to the Resource VDI and then add the link between the product type R-VDI and the product instance d

   Resource res = rvdiTool.createResource(productURL,MIME_TYPE,"productName");

   rvdiTool.addAsset(res, List<Licence>, null);
6. Store the product instance R-VDI

   rvdiTool.store(filePath);
7. Advertise the product instance R-VDI

   rvdiTool.advertise(filePath);

8 Integrating CoApps – The Alinari/WIPRO/UTI use case

8.1 Brief description of use case

This final use case integrates three CoApps: The Alinari Photographic Archive Management Application (PAM) and the two retail applications from UTI and WIPRO. The goal is, on the one hand, to demonstrate interoperability among applications, on the other hand to show how CONVERGENCE could support additional use cases, not described here.

The use case (see **Figure** ***8-1***) involves two user profiles, one for the content creator (a photographer) and one for the content consumer. The business stakeholders are Alinari (which sells image rights and fine art books and printed material), WIPRO and UTI (both technology providers for the retail industry). Once the photographer has created photographic images, including automatically generated camera information, and published her creations through the PAM interface, CONVERGENCE associates the photo with the Product VDIs for her camera and lenses, created with the WIPRO application) and with special offers to buy the equipment (created with the UTI application). When the content consumer retrieves images through the PAM, she receives similar information.

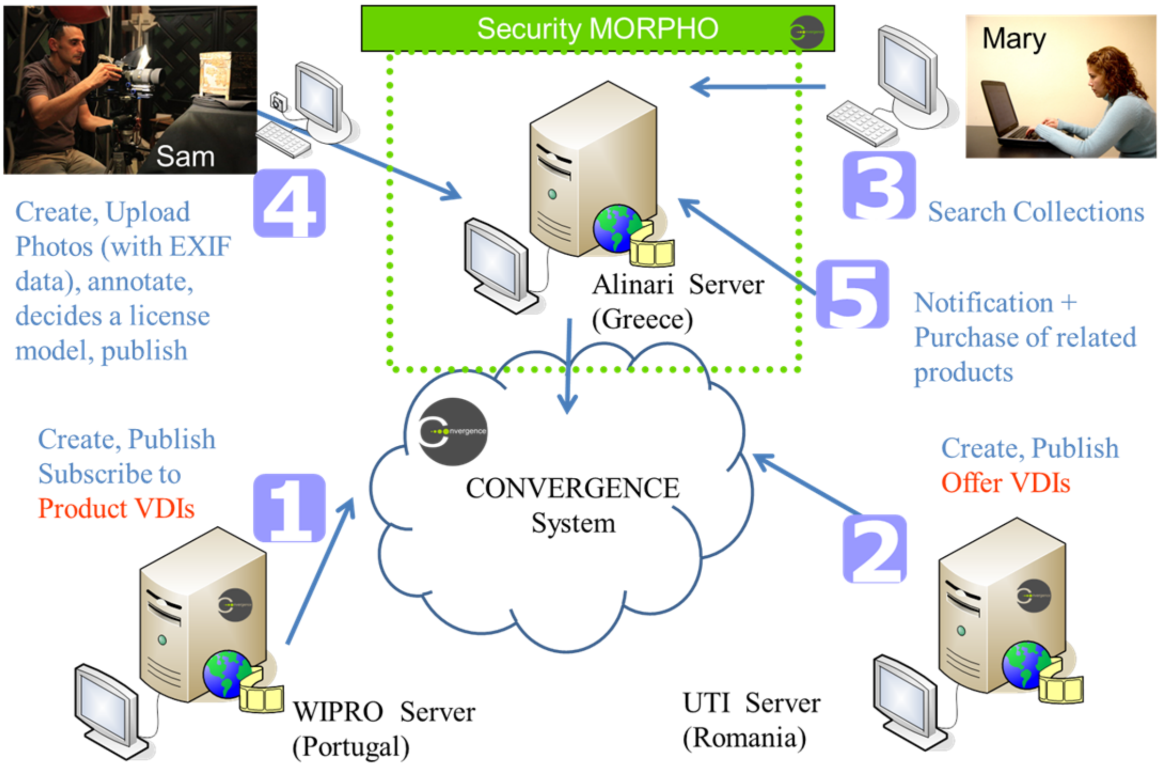

Security MORPHO
Sam
Mary
4
Create, Upload Photos (with EXIF data), annotate, decides a license model, publish
3
Search Collections
Alinari Server (Greece)
5
Notification + Purchase of related products
CONVERGENCE System
Create, Publish Subscribe to Product VDIs
1
Create, Publish Offer VDIs
2
WIPRO Server (Portugal)
UTI Server (Romania)

Figure 8-1: Integrated Alinari/WIPRO/UTI application

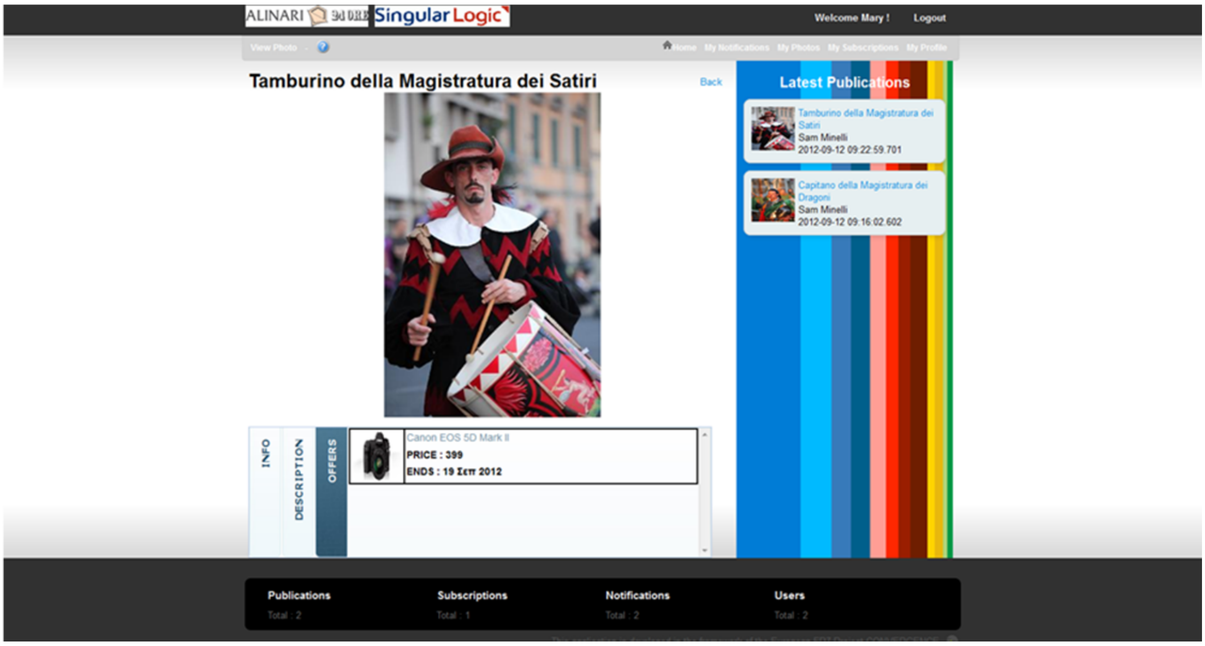
Users registered to the Alinari application are able to:

- Create, Annotate, License and Publish photos (photographers)

- Submit queries, personalizing the period to which the query refers (photographers and content consumers)
- Purchase products related to digital photos, such as cameras and lenses (photographers and content consumers).

To support this scenario, the application automatically matches EXIF information incorporated in the photo with product VDIs, for cameras, lenses etc. Other trusted subjects (e.g. UTI), can then use this information to generate offer VDIs, which can then be visualized through the PAM application.

In brief, the application provides the consumer with information that could lead her to make new purchases – in this case, cameras and lenses, in the future books from the photographer, Alinari books and posters, etc..



Tamburino della Magistratura dei Satiri

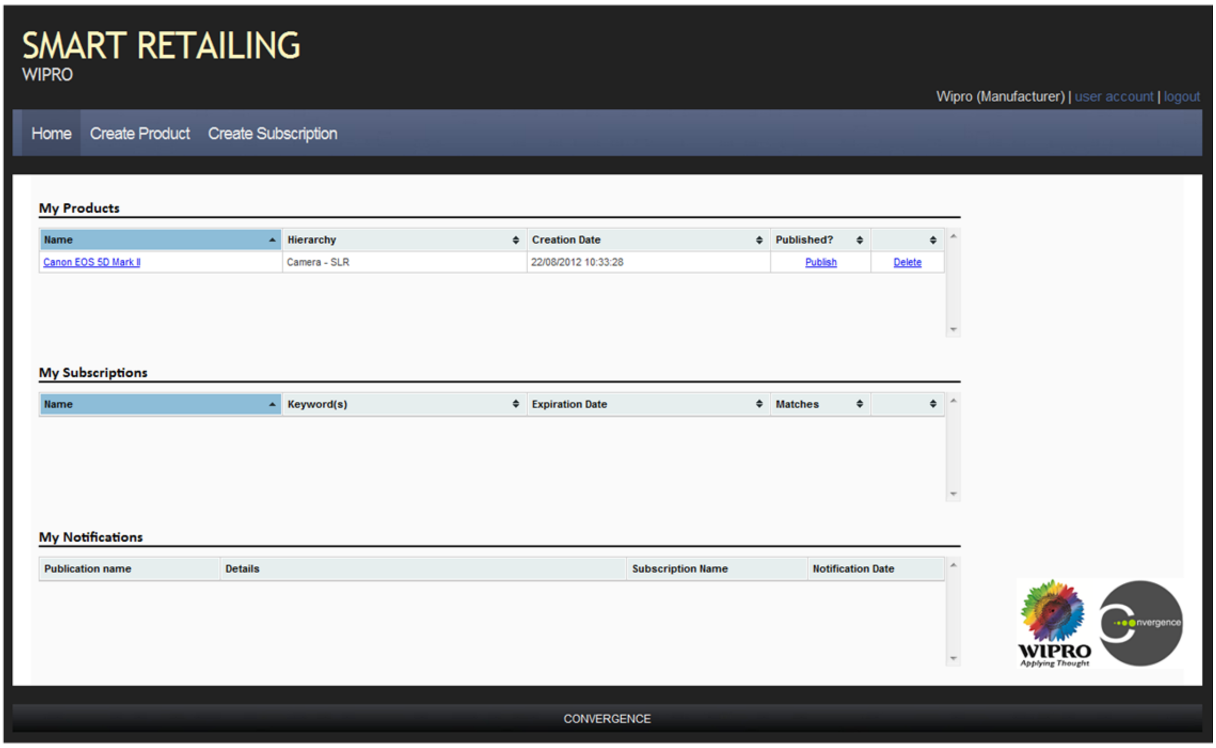
Canon EOS 5D Mark II
PRICE : 399
ENDS : 19 Σεπ 2012

Tamburino della Magistratura dei Satiri
Sam Minelli
2012-09-12 09:22:59.701

Capitano della Magistratura dei Dragoni
Sam Minelli
2012-09-12 09:16:02.602

Figure 8-2: Camera offers in the Alinari PAM application (see below the drummer).

For this system to work, a manufacturer has to publish a camera product VDI in the CONVERGENCE system, before bringing a new camera to market. To do this she accesses the application, checks her products and publishes a product Camera VDI.



My Products

Name	Hierarchy	Creation Date	Published?	
Canon EOS 5D Mark II	Camera - SLR	22/08/2012 10:33:28	Publish	Delete

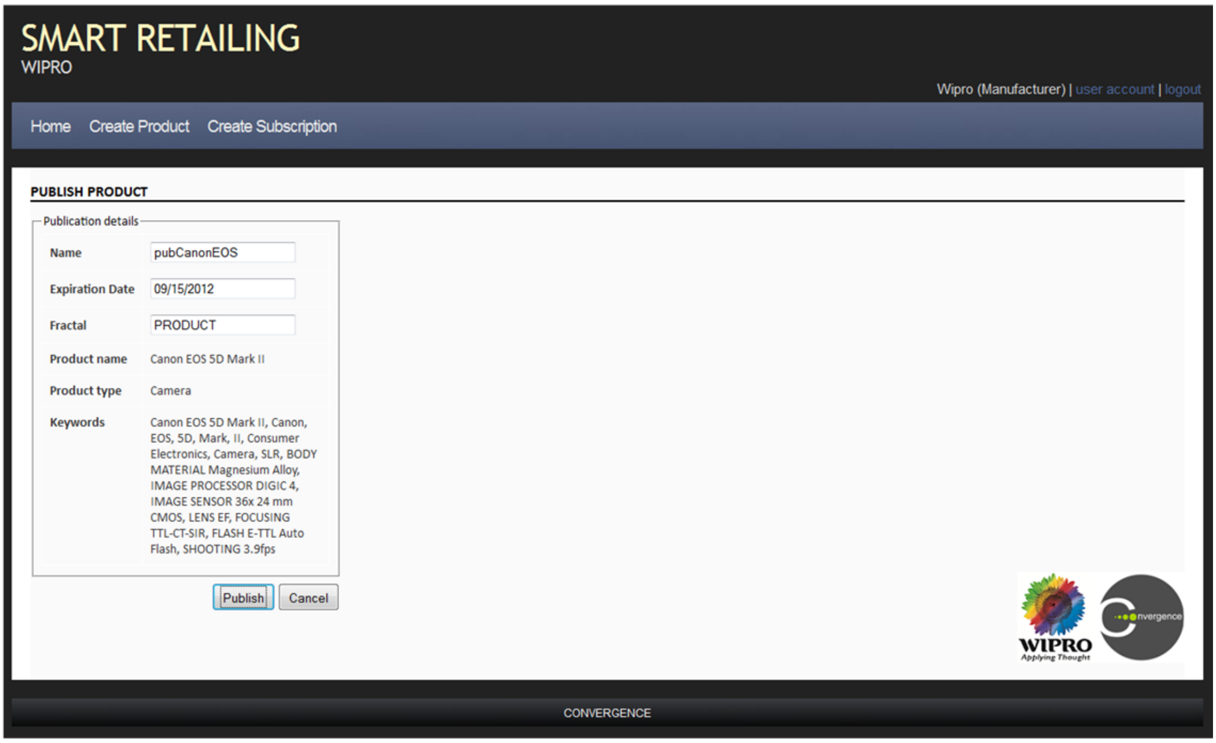
My Subscriptions

Name	Keyword(s)	Expiration Date	Matches	

My Notifications

Publication name	Details	Subscription Name	Notification Date

Figure 8-3: Home screen in Wipro application, with camera product VDI created



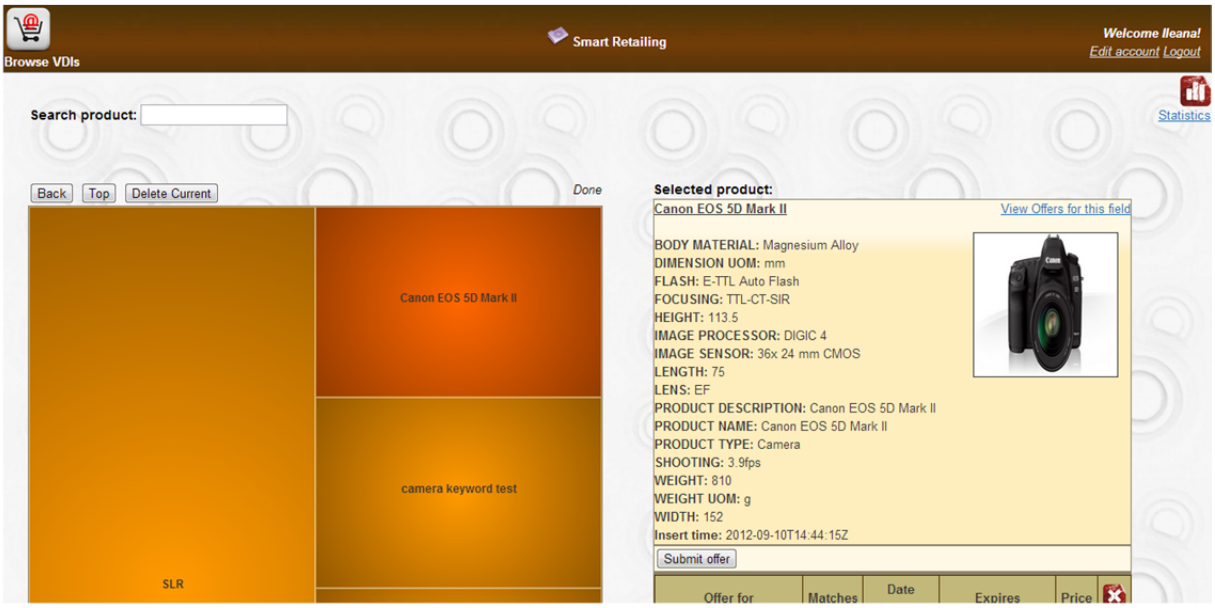
PUBLISH PRODUCT

Publication details

Name	pubCanonEOS
Expiration Date	09/15/2012
Fractal	PRODUCT
Product name	Canon EOS 5D Mark II
Product type	Camera
Keywords	Canon EOS 5D Mark II, Canon, EOS, 5D, Mark, II, Consumer Electronics, Camera, SLR, BODY MATERIAL Magnesium Alloy, IMAGE PROCESSOR DIGIC 4, IMAGE SENSOR 36x 24 mm CMOS, LENS EF, FOCUSING TTL-CT-SIR, FLASH E-TTL Auto Flash, SHOOTING 3.9fps

Figure 8-4: Publish product form in Wipro application

The camera product VDI is published in the fractal “PRODUCT” that the Manufacturer shares with her certified Retailers. Once the camera product VDI is published, all retailers that are allowed to subscribe to the fractal (all authorized retailers) will have the new product in their catalogue and will be able to make special offers to customers wishing to purchase the product.



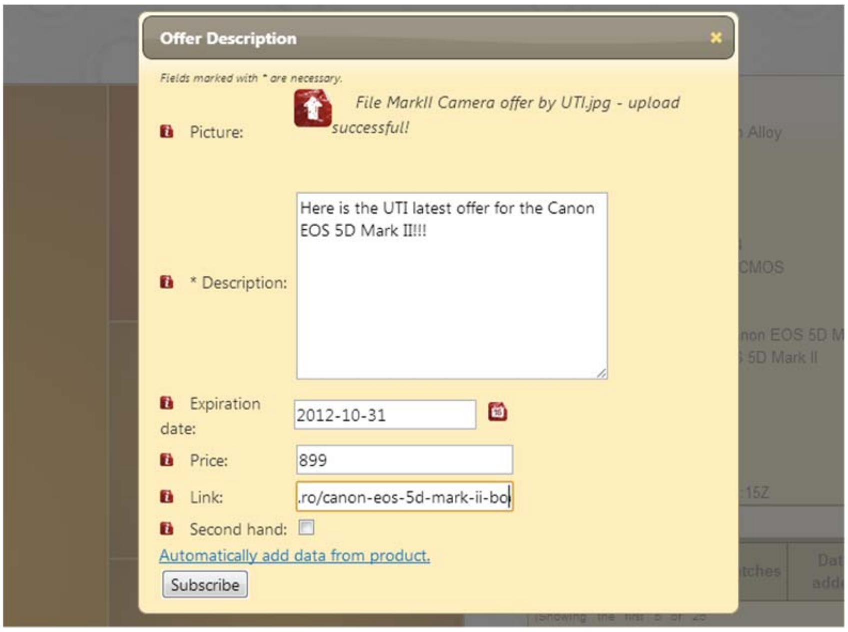
Search product:

Selected product:
Canon EOS 5D Mark II

BODY MATERIAL: Magnesium Alloy
DIMENSION UOM: mm
FLASH: E-TTL Auto Flash
FOCUSING: TTL-CT-SIR
HEIGHT: 113.5
IMAGE PROCESSOR: DIGIC 4
IMAGE SENSOR: 36x 24 mm CMOS
LENGTH: 75
LENS: EF
PRODUCT DESCRIPTION: Canon EOS 5D Mark II
PRODUCT NAME: Canon EOS 5D Mark II
PRODUCT TYPE: Camera
SHOOTING: 3.9fps
WEIGHT: 810
WEIGHT UOM: g
WIDTH: 152
Insert time: 2012-09-10T14:44:15Z

Figure 8-5: A new product is automatically added to the product catalogue of the UTI application

To create such an offer, a retailer defines an offer VDI linked to the Manufacturer’s product VDI. The offer VDI will contain specific fields for the offer (e.g. price, period of validity for the offer, picture of the promotional material). The VDI is published in the “OFFER” fractal.



Fields marked with * are necessary.

Picture: File MarkII Camera offer by UTI.jpg - upload successful!

* Description: Here is the UTI latest offer for the Canon EOS 5D Mark II!!!

Expiration date: 2012-10-31

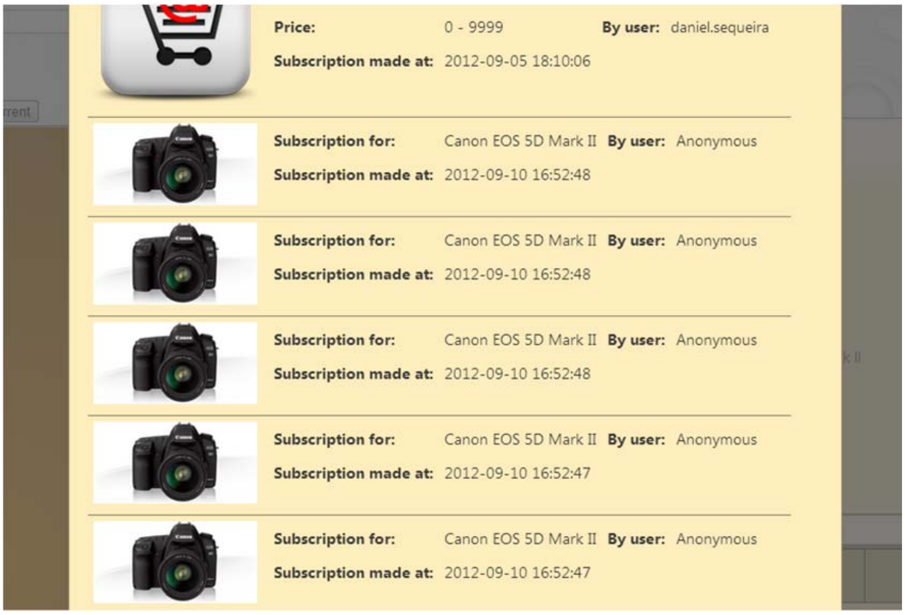
Price: 899

Link: .ro/canon-eos-5d-mark-ii-bo

Second hand: □

Figure 8-6: A new product offer is made by the retailer in the UTI application

After this step, every subscriber to the offers fractal will be notified about the new offer. The notification will allow the user to browse the details and go to a link where she can purchase the product. But, even before the user clicks on the link, the retailer will be able to see user subscriptions that match the publication.



Price: 0 - 9999 By user: daniel.sequeira
Subscription made at: 2012-09-05 18:10:06

Subscription for: Canon EOS 5D Mark II By user: Anonymous
Subscription made at: 2012-09-10 16:52:48

Subscription for: Canon EOS 5D Mark II By user: Anonymous
Subscription made at: 2012-09-10 16:52:48

Subscription for: Canon EOS 5D Mark II By user: Anonymous
Subscription made at: 2012-09-10 16:52:48

Subscription for: Canon EOS 5D Mark II By user: Anonymous
Subscription made at: 2012-09-10 16:52:47

Subscription for: Canon EOS 5D Mark II By user: Anonymous
Subscription made at: 2012-09-10 16:52:47

Figure 8-7*:* Retailers can consult subscriptions that match the offer

9 Conclusions and key findings

In this deliverable, we have presented the final version of the five Tools and Applications originally planned in the CONVERGENCE project, as well as an additional integrated application. The key findings of this deliverable can be summarized as follows:

- The five developed CONVERGENCE Tools and related APIs made possible the development of all applications for the considered use case scenarios.
- Going a step beyond, it was also shown that these Tools and APIs can be used in order to design and develop additional applications that correspond to more complex use cases (i.e. the considered integration between Alinari and Retail applications).

In the first case, with respect to Information Centric Networks (ICN) theory, the analysis of the applications as followed in D7.1 was decomposed in five fundamental operations: Subscription and registration to applications, creation of application VDIs, publish of VDIs, subscription to VDIs, revocation of VDIs. Following this decomposition, in D7.2 and D7.3 additional APIs required for these operations were also identified and implemented, such as creation of licenses, insertion of metadata as well as creation of ERRs.

Once the development of the five CONVERGENCE applications was finalized and validated through the CONVERGENCE trials, the consortium identified additional fields of exploitation of these applications. In particular, the first integrated application made possible the purchase of specific products such as cameras and lenses, related to photographic digital content. Since the structure of VDIs for all applications is similar, it is possible to extract information from a specific application's VDI in the environment of another application with the help of the APIs described in Description of the APIs for CONVERGENCE Tools. Therefore, professional photographers that make use of the Alinari application for example, can embed information related to the camera or the lenses of the photograph that is provided by the WIPRO application inside the Photo VDI.

If another user of the Alinari application is interested for example to buy the camera with which the photo was taken, then with the help of the UTI application an Offer VDI (i.e. VDI containing a specific offer for this camera) is created and provided to the user.

The outcomes of this integration are important for realistic ICN networks, where it is expected that different types of digital media content will be exchanged. Therefore, as search and retrieval of content is of primary importance, ICN applications should be in position to deal with different types of content and present it to the end users.

In this context, we believe that the interoperability demonstrated in the integrated application framework is relevant not just for business applications but also for education. Taking the FMSH and LMU scenarios as an example, it would be possible, for instance, for two institutions to share video material annotated by their respective users viewers, while using different software solutions and pursuing different educational goals (see **Figure *9-1***).

With respect to **Figure *9-2*** for example, students registered to the LMU application can watch Video VDIs created and published by the FMSH application provided that they have the appropriate rights and post an analysis in an FMSH channel.

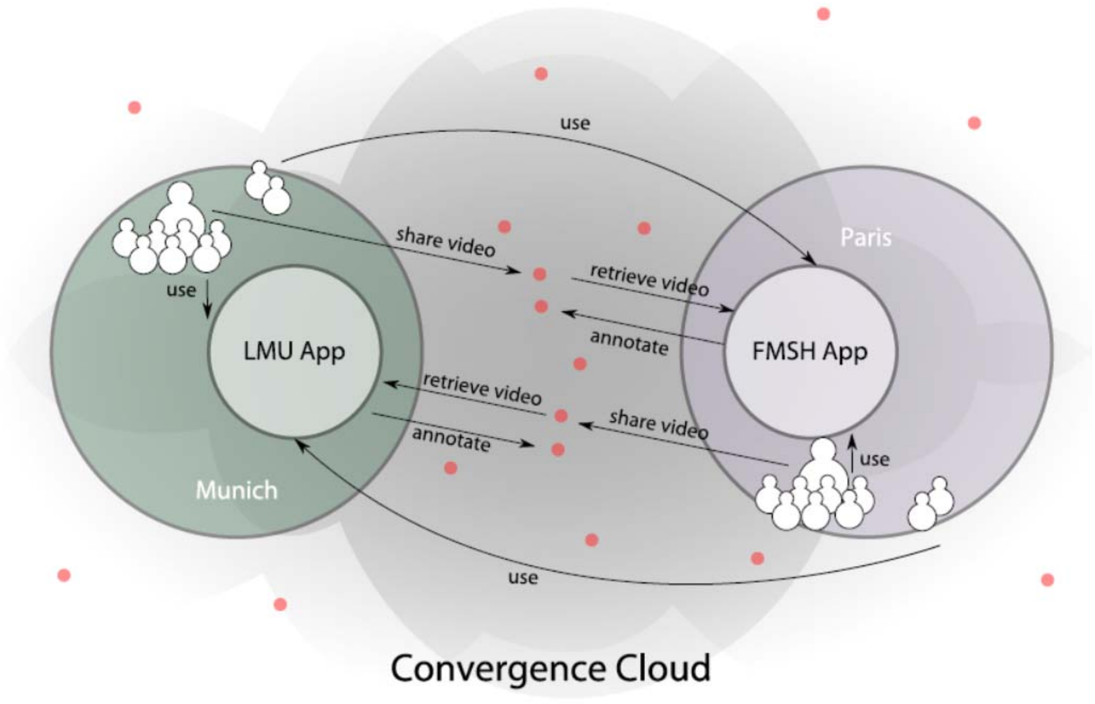


Figure 9-1: The LMU-FMSH integration vision

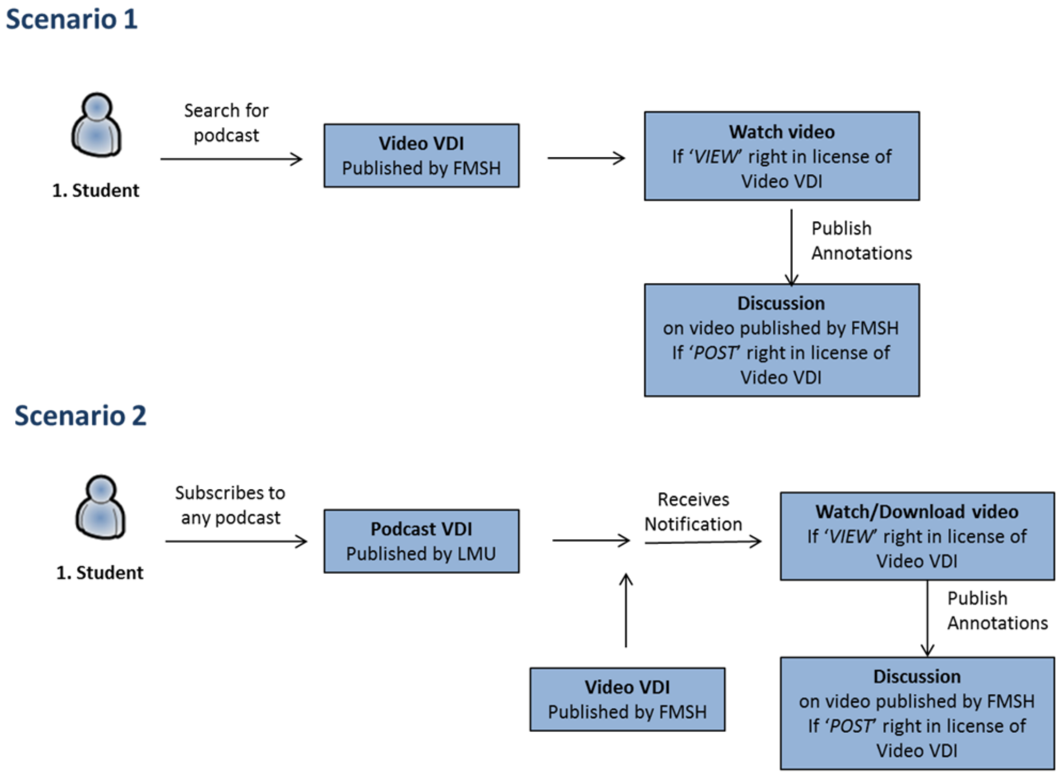


Figure 9-2: Scenarios of LMU-FMSH integrated scenario

10 References

[1] CONVERGENCE Project Deliverable D.7.1

[2] CONVERGENCE Project Deliverable D.7.2

[3] CONVERGENCE Project Deliverable D.6.2

[4] CONVERGENCE Project Deliverable D.6.3

[5] CONVERGENCE Project Deliverable D.4.2

11 ANNEX A

11.1 ESCoM-FMSH OWL Ontology & Video Analysis Metadata

11.1.1 *Introduction*

The ESCoM-FMSH ontology contains:

- The definition of video analysis structure
- 2 thesauri
- ▪ A thesaurus of terms specific to video analysis
- ▪ A thesaurus of Real World Objects
- A common ontology of description patterns, called ESCOM.
- A set of description templates for domain-specific ontologies. Each template defines the way each thematic of a domain can be analysed, and use description patterns of the ESCOM ontology.
    - **ADA**: Archaeology
    - **ALIA**: Literature
    - **ARC**: Intangible Diversity
    - **FMSH-AAR**: Social and Human Sciences
    - **PCIA**: Andean Intangible Cultural Heritage

The preferred domains for CONVERGENCE will be ARC and PCIA.

Video analyses are stored in directory */rdf_metaDescriptions*. An example of an analysis where all possible fields are filled in can be found in the file *full_description.rdf* (using ALIA ontology). The ontologies use the namespace http://escom.msh-paris.fr/. They are divided into several files stored in directory */owl_ontologies*,:

File	Description	Content/Example
Videodescription.owl	- Classes used for video analysis - Object & Data Properties used for video analysis and every ontology as well	
Static_vocabulary.owl	Thesaurus of Real World Objects	
Escom_vocabulary.owl	Thesaurus of terms specific to video analysis	

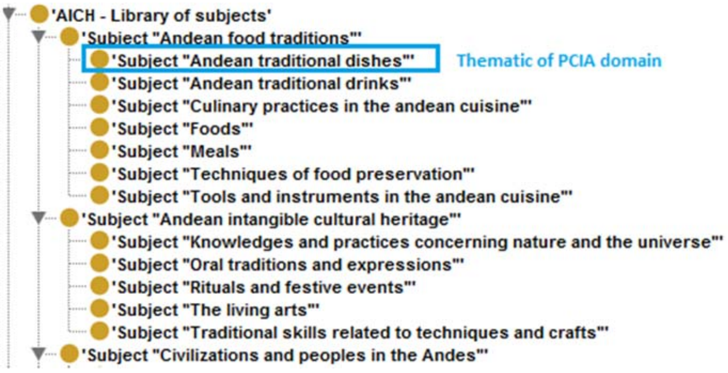
Static_type.owl	Ontology of data types, used by description patterns (escom ontology)	
Escom_*.owl	ESCOM ontology	*Escom_root.owl* *Escom_patterns.owl* *Escom_signs.owl*
{domain}_*.owl	Domain ontology. Example, for PCIA domain ->	*Pcia_root.owl* *Pcia_sequences.owl* *Pcia_patterns.owl*

To facilitate browsing, all owl files are stored in the root directory,

File	**Description**	**Content (imports)**
escom.owl	Shared ontology	*Videodescription.owl* *Static_vocabulary.owl* *Static_types.owl* *Escom_vocabulary.owl* *Escom_root.owl* *Escom_patterns.owl* *Escom_signs.owl*
{domain}.owl	Domain ontology. It is an import of the shared ontology (escom.owl) and files specific to the domain. Example, for PCIA domain ->	*Escom.owl* *Owl_ontologies/pcia_root.owl* *Owl_ontologies/pcia_sequences.owl* *Owl_ontologies/pcia_patterns.owl*

11.1.2 Description templates

Most of the files listed above define classes used in description templates. Description templates are used during the video analysis process, so their content may not be useful for search – except as a way of defining themes for individual domains. These are defined in the file *{domain}_root.owl* (example: *pcia_root.owl*).



'AICH - Library of subjects'

'Subject "Andean food traditions"'

'Subject "Andean traditional dishes"'

'Subject "Andean traditional drinks"'

'Subject "Culinary practices in the andean cuisine"'

'Subject "Foods"'

'Subject "Meals"'

'Subject "Techniques of food preservation"'

'Subject "Tools and instruments in the andean cuisine"'

'Subject "Andean intangible cultural heritage"'

'Subject "Knowledges and practices concerning nature and the universe"'

'Subject "Oral traditions and expressions"'

'Subject "Rituals and festive events"'

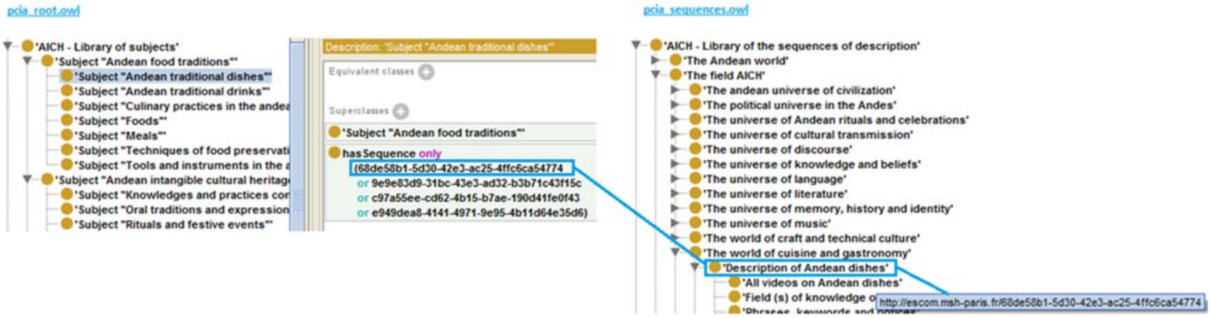
'Subject "The living arts"'

'Subject "Traditional skills related to techniques and crafts"'

'Subject "Civilizations and peoples in the Andes"'

Figure 11-1: Description Templates of *PCIA* Domain

Note: each class is named (*rdf:about*) by an URI in the form: *http://escom.msh-paris.fr/{guid}* (*example: http://escom.msh-paris.fr/589f6f72-1998-40b2-ad58-8f2c3d2c68c1*), and labelled (*rdfs:label*) in several languages (at least French & English). Each class also includes a non-unique *dc:identifier* in full-text. Some classes are also associated with a comment (*rdfs:comment*) and/or a description (*dc:description*). The remaining files describe a series of classes, each containing references to other classes. The example below shows the relationship between classes *pcia_root* and *pcia_sequences*:



pcia_root.owl

'AICH - Library of subjects'

'Subject "Andean food traditions"'

'Subject "Andean traditional dishes"'

'Subject "Andean traditional drinks"'

'Subject "Culinary practices in the andea

'Subject "Foods"'

'Subject "Meals"'

'Subject "Techniques of food preservati

'Subject "Tools and instruments in the a

'Subject "Andean intangible cultural heritag

'Subject "Knowledges and practices cor

'Subject "Oral traditions and expression

'Subject "Rituals and festive events"'

Description: 'Subject "Andean traditional dishes"'

Equivalent classes

Superclasses

'Subject "Andean food traditions"'

hasSequence only
(68de58b1-5d30-42e3-ac25-4ffc6ca54774
or 9e9e83d9-31bc-43e3-ad32-b3b71c43f15c
or c97a55ee-cd62-4b15-b7ae-190d41fe0f43
or e949dea8-4141-4971-9e95-4b11d64e35d6)

pcia_sequences.owl

'AICH - Library of the sequences of description'

'The Andean world'

'The field AICH'

'The andean universe of civilization'

'The political universe in the Andes'

'The universe of Andean rituals and celebrations'

'The universe of cultural transmission'

'The universe of discourse'

'The universe of knowledge and beliefs'

'The universe of language'

'The universe of literature'

'The universe of memory, history and identity'

'The universe of music'

'The world of craft and technical culture'

'The world of cuisine and gastronomy'

'Description of Andean dishes'

'All videos on Andean dishes'

'Field (s) of knowledge o

http://escom.msh-paris.fr/68de58b1-5d30-42e3-ac25-4ffc6ca54774

Figure 11-2: Description Sequences of *PCIA* Domain

The table below summarizes these relationships:

File	Object Property	Source for Object Property
{domain}_root.owl	**hasSequence** only (list of classes of)	{domain}_sequences.owl
{domain}_sequences.owl	**hasPattern** only (list of classes of)	{domain}_pattern.owl
{domain}_pattern.owl	**hasObject** only (list of classes of)	escom_root.owl
escom_root.owl	**hasScheme** only (list of classes of) **hasDataType** only (list of classes	escom_patterns.owl

	of)	static_types.owl
escom_patterns.owl	**hasSign** only (list of classes of) **hasDataType** only (list of classes of)	escom_signs.owl static_types.owl
escom_signs.owl	**hasDataType** only (list of classes of)	static_types.owl
static_types.owl	**hasFacet** only (list of classes of)	Static_vocabulary.owl

11.1.3 Thesaurii

Individuals are stored in two different files:

- *escom_vocabulary.owl*: Thesaurus of terms specific to video analysis (kinds of analysis, audiovisual techniques, kinds of video material, etc.)

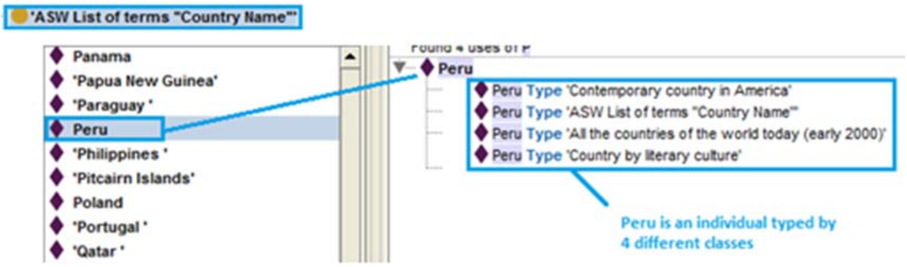
- *static_vocabulary.owl*: Thesaurus of Real World Objects. This is the most important file of the ontology, with more than 28 500 triples.

As classes, each individual is named (*rdf:about*) by an URI in the form of: *http://escom.msh-paris.fr/{guid}*, and labelled (*rdfs:label*) in several languages (at least French & English). It also includes a non-unique *dc:identifier* in full-text,. Some individuals are also described with a comment (*rdfs:comment*) and/or description (*dc:description*).

The thesaurus of RWOs is composed of:

- Classes for lists of terms that reference the individuals in their largest category (person, country, instrument, etc.)
- Classes for specific list of individuals, called "facets" (French writers of the 19th Century, cities of Peru, etc.)

Thus, an individual can be typed with several classes. In the example below, we see that Peru is referenced with 4 different classes:



'ASW List of terms "Country Name"'

Panama
'Papua New Guinea'
'Paraguay '
Peru
'Philippines '
'Pitcairn Islands'
Poland
'Portugal '
'Qatar '

Found 4 uses of P

Peru
Peru Type 'Contemporary country in America'
Peru Type 'ASW List of terms "Country Name"'
Peru Type 'All the countries of the world today (early 2000)'
Peru Type 'Country by literary culture'

Peru is an individual typed by 4 different classes

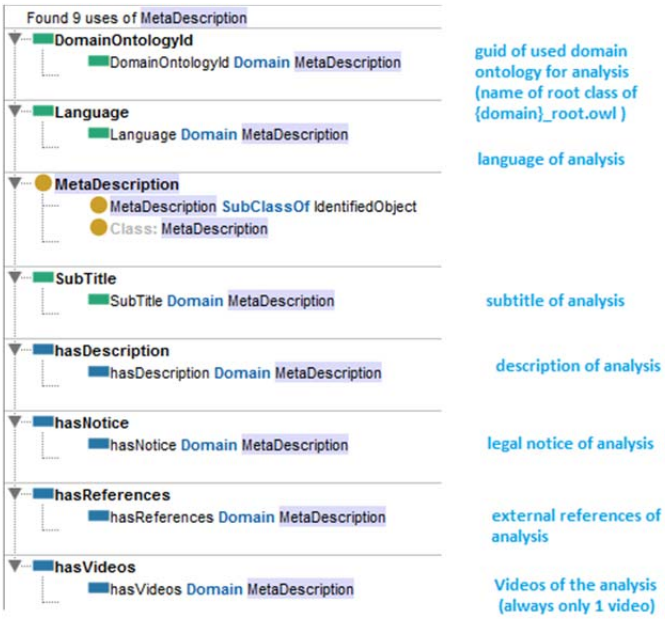
Figure 11-3: '*Peru*' Individual

11.1.4 Video Analysis Metadata

The Metadata for an analysis is divided in 3 levels:

- Analysis of the analysis (Class "*MetaDescription*");
- Analysis of the whole video (Class "*Video*");
- Analysis of parts of the video (Class "*VideoShot*").

Each class has its own structure, though the structures for*Video* and *VideoShot* are very similar. *MetaDescription* is used as internal information – it may not be relevant for search. The structure of these 3 classes is shown in following figures below.



Found 9 uses of MetaDescription

DomainOntologyId
DomainOntologyId Domain MetaDescription

guid of used domain ontology for analysis (name of root class of {domain}_root.owl)

Language
Language Domain MetaDescription

language of analysis

MetaDescription
MetaDescription SubClassOf IdentifiedObject
Class: MetaDescription

SubTitle
SubTitle Domain MetaDescription

subtitle of analysis

hasDescription
hasDescription Domain MetaDescription

description of analysis

hasNotice
hasNotice Domain MetaDescription

legal notice of analysis

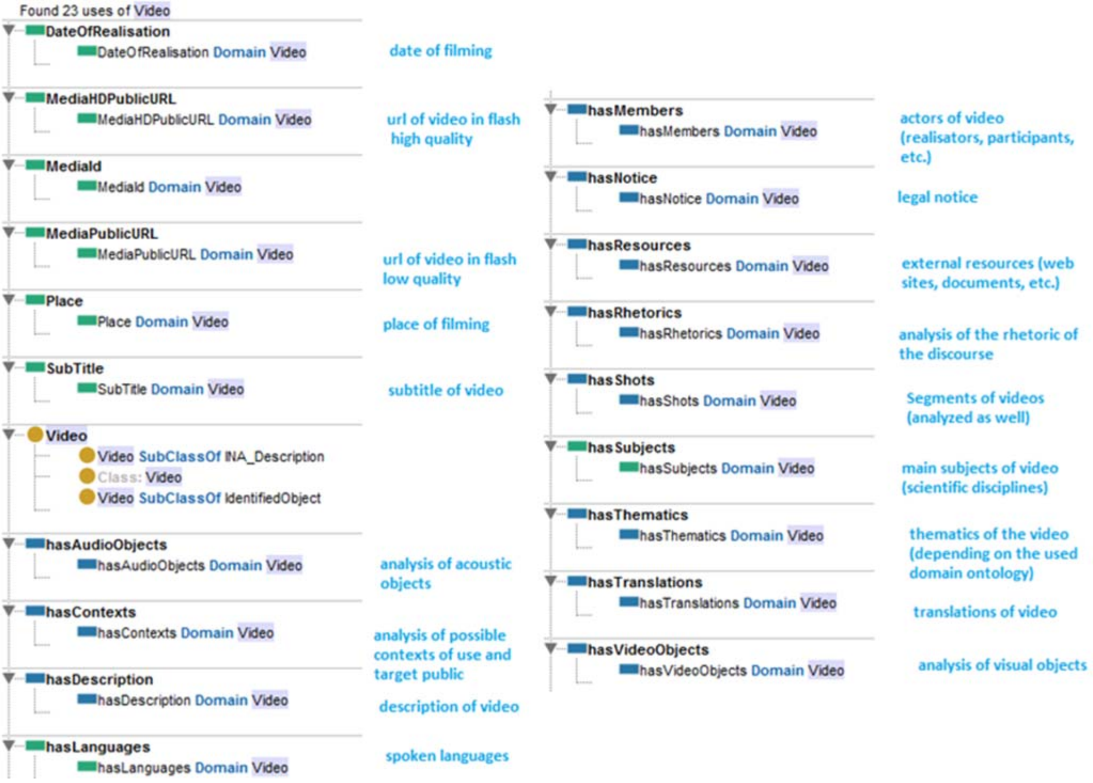
hasReferences
hasReferences Domain MetaDescription

external references of analysis

hasVideos
hasVideos Domain MetaDescription

Videos of the analysis (always only 1 video)

Figure 11-4: *MetaDescription* Class



Found 23 uses of Video

DateOfRealisation
DateOfRealisation Domain Video — date of filming

MediaHDPublicURL
MediaHDPublicURL Domain Video — url of video in flash high quality

MediaId
MediaId Domain Video

MediaPublicURL
MediaPublicURL Domain Video — url of video in flash low quality

Place
Place Domain Video — place of filming

SubTitle
SubTitle Domain Video — subtitle of video

Video
Video SubClassOf INA_Description
Class: Video
Video SubClassOf IdentifiedObject

hasAudioObjects
hasAudioObjects Domain Video — analysis of acoustic objects

hasContexts
hasContexts Domain Video — analysis of possible contexts of use and target public

hasDescription
hasDescription Domain Video — description of video

hasLanguages
hasLanguages Domain Video — spoken languages

hasMembers
hasMembers Domain Video — actors of video (realisators, participants, etc.)

hasNotice
hasNotice Domain Video — legal notice

hasResources
hasResources Domain Video — external resources (web sites, documents, etc.)

hasRhetorics
hasRhetorics Domain Video — analysis of the rhetoric of the discourse

hasShots
hasShots Domain Video — Segments of videos (analyzed as well)

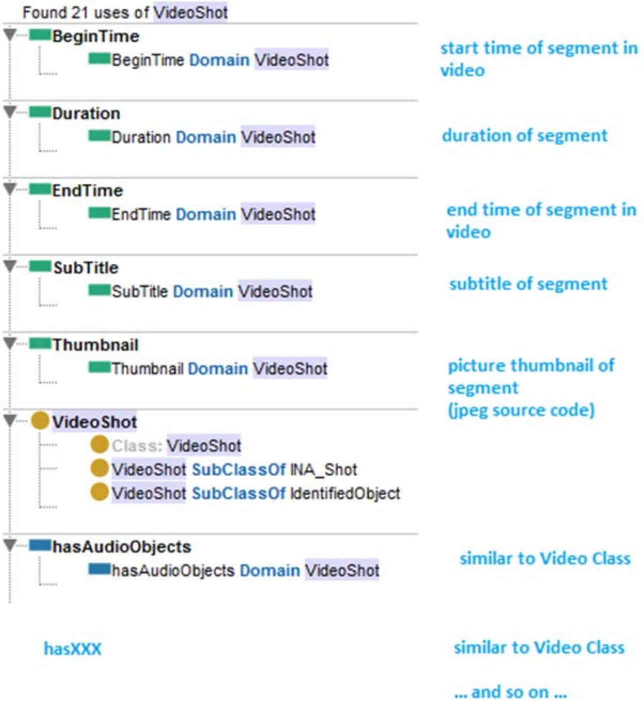
hasSubjects
hasSubjects Domain Video — main subjects of video (scientific disciplines)

hasThematics
hasThematics Domain Video — thematics of the video (depending on the used domain ontology)

hasTranslations
hasTranslations Domain Video — translations of video

hasVideoObjects
hasVideoObjects Domain Video — analysis of visual objects

Figure 11-5: *Video* Class



Found 21 uses of VideoShot

BeginTime
BeginTime Domain VideoShot — start time of segment in video

Duration
Duration Domain VideoShot — duration of segment

EndTime
EndTime Domain VideoShot — end time of segment in video

SubTitle
SubTitle Domain VideoShot — subtitle of segment

Thumbnail
Thumbnail Domain VideoShot — picture thumbnail of segment (jpeg source code)

VideoShot
Class: VideoShot
VideoShot SubClassOf INA_Shot
VideoShot SubClassOf IdentifiedObject

hasAudioObjects
hasAudioObjects Domain VideoShot — similar to Video Class

hasXXX — similar to Video Class

... and so on ...

Figure 11-6: *VideoShot* Class

Almost every class uses the data property "***Kind***", which describes the kind of the corresponding data (kind of analysis, kind of video, kind of resource, kind of actor, etc.). This property references the URI for an individual in the thesaurus (escom_vocabulary.owl). Other data properties are literals (string, int, datetime). Each class is named (*rdf:about*) by an URI in the form: *http://escom.msh-paris.fr/{guid}*, and labelled (*rdfs:label*) in one language, corresponding to the title of the corresponding object (analysis, video, actor, resource, etc.). Some classes are also described with a comment (*rdfs:comment*) and/or description (*dc:description*).

11.1.5 Themes

When analysing a video, the analysis takes account of the relevant domain:

- Themes relevant to the domain are identified. The class of the thematic is referenced in the data property "*Kind*".
- For each theme, analysts can fill in several concepts (tags or keywords), in the description template for the theme. Each concept is stored in the object property "***hasConcepts***" for the theme, and contains the following information:
    - *DescriptionPattern*: indicates which part of the description template the concept refers to. This information is useful for the analysis application but may not be useful for searching.
    - *Kind*: data type of the concept. It references a class of static_types.owl. This is also useful for analysis application; it may not be used for search.
    - *Value*: references the individual. This individual can be:
        - An individual of the thesaurus
        - An individual created inside the analysis, when the analyst filled the concept with free text. In that case, the only information we have is the label of this individual, in the language of the analysis.

11.1.6 Analysis Metadata Queries

Users can search for analyses using the following criteria:

Property	Class of Individual	Data type	Facet	Comments
Language	MetaDescription	Language (en, fr, etc.)		Language of analysis
hasLanguages	Video	Language		Spoken language in video
Label, SubTitle	Video, VideoShot	Text		
Kind	Video	Individual URI	'Audiovisual document'	Kind of video (TV program, interview, etc.)

hasSubjects	Video, VideoShot	Individual URI	‘Basic classification’	Scientific disciplines (Anthropology, Mathematics, etc.)
Name, Forname	Member of Video, VideoShot	Text		
Kind	Member of Video, VideoShot	Individual URI	‘Speaking role in any recording type’	Role of an actor (realization, interviewee, etc.)
Company	Member of Video, VideoShot	Individual URI	‘ASW list of terms "Institution Name"’	Company or institution of an actor
Kind	Thematic of Video, VideoShot	Class URI	Root class (example: ‘AICH - Library of subjects’)	Thematic (or subject) of a domain (Andean Artifacts, Andean Food, etc.)
Value	Concept of Video, VideoShot	Individual URI Or text (label of individual)	See below	Tag. Each tag is saved as a concept of a thematic.

Queries on Concepts depend on the selected domain. Here is a sample of the relevant facets for the *PCIA* domain (the domain used in the CONVERGENCE trial).

Facet label@en	**Facet URI**
All the countries of the world today (early 2000)	http://escom.msh-paris.fr/589f6f72-1998-40b2-ad58-8f2c3d2c68c1
AICH facet for the specialized CT "Department of Peru"	http://escom.msh-paris.fr/a5c6e92d-af07-4ce5-ac31-5cecfeaf4ce3
AICH facet for the specialized CT "Department of Bolivia"	http://escom.msh-paris.fr/f6376fd3-4377-448c-a428-3ee4dd483dfb

ASW facet for the CT "Natural language (world)"	http://escom.msh-paris.fr/09899f94-8b2b-4be4-89a4-77664941c72b
ASW facet for the CT "Quechua Languages”	http://escom.msh-paris.fr/0e2d17d9-04e4-4ad4-a15d-bb149fc30b94
ASW facet for the CT "Quechua II South Dialects”	http://escom.msh-paris.fr/a683e0ae-f39f-4d20-b670-9d8ec69d7c41
Oral literature	http://escom.msh-paris.fr/db8ca54d-97b5-4c52-9c1e-197be7701269
Living Arts	http://escom.msh-paris.fr/fcf0f229-ba66-43ea-8e30-ac30b4ffaea3
AICH facet for the specialized CT "Andean crafts"	http://escom.msh-paris.fr/d6bb46a0-c136-466b-8bd1-20364eb59b2e
AICH facet for the specialized CT "Kind of andean music"	http://escom.msh-paris.fr/3d7d32e9-1c77-46e3-8ab3-ddc3e98d47bd
AICH facet for the specialized CT "Andean musical instruments"	http://escom.msh-paris.fr/8ffefa8a-578a-4aea-86ca-b0e33ea2fec9
AICH facet for the specialized CT "Andean dances"	http://escom.msh-paris.fr/b44422f2-5c8d-4804-a56e-14a91ed9103b
Scientific communication event	http://escom.msh-paris.fr/0a3e0e74-c2ef-49a6-8158-c54a43976344
Pedagogical communications sector	http://escom.msh-paris.fr/d7d65256-01fc-4b02-809e-bba6ae42a4fc
ASW facet for the for CT "Name of scientific discipline"	http://escom.msh-paris.fr/2dd90615-7d83-460f-99ff-727022b87e6c

12 ANNEX B - Test Datasets

12.1 VDI Tool tests

12.1.1 RVDI creation and parsing

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<mpegm-didl:DIDL xmlns:didmodel="urn:mpeg:mpeg21:2002:02-DIDMODEL-NS"
xmlns:didl_mp21="urn:mpeg:mpeg21:2002:02-DIDL-NS"
xmlns:ipmpinfo="urn:mpeg:mpeg21:2004:01-IPMPINFO-NS"
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
xmlns:mpegmipmpdidl="urn:mpeg:mpegM:schema:08-IPMPDIDL-NS:2012" xmlns:mpegm-
didl="urn:mpeg:mpegM:schema:06-didl-NS:2012"
xmlns:didl_msx="urn:mpeg:maf:schema:mediastreaming:DIDLextensions"
xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS" xmlns:er="urn:mpeg:mpeg21:2005:01-ERL-
NS" xmlns:rel-r="urn:mpeg:mpeg21:2003:01-REL-R-NS" xmlns:rel-
m1x="urn:mpeg:mpeg21:2005:01-REL-M1X-NS" xmlns:rel-sx="urn:mpeg:mpeg21:2003:01-
REL-SX-NS" xmlns:ipmpmsg="urn:mpeg:mpegB:schema:IPMP-XML-MESSAGES:2007"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" 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:ipmpinfo_msx="urn:mpeg:maf:Schema:mediastreaming:IPMPINFOextensions:2007"
xmlns:mpeg4ipmp="urn:mpeg:mpeg4:IPMPSchema:2002"
xmlns:mpeg7_v3="urn:mpeg:mpeg7:schema:2004"
xmlns:ns21="urn:conv:metadata:schema:2011" xmlns:ns22="urn:mpeg:mpqf:schema:2008"
xmlns:ns23="urn:conv:relation:schema:2011">
    <mpegm-didl:Item id="A0">
        <mpegm-didl:Descriptor>
            <mpegm-didl:Statement>
                <dii:Identifier>urn:eu:CONVERGENCE:server1\ad9d8160-4a6d-4b67-
850f-d4f03869ea28</dii:Identifier>
            </mpegm-didl:Statement>
        </mpegm-didl:Descriptor>
        <mpegm-didl:Descriptor>
            <mpegm-didl:Statement>
                <didl_msx:Metadata>
                    <didl_msx:StructuredData ref="urn:conv:metadata:schema:2011">
                        <ns21:CONVERGENCEMetadata>

<ns21:SequenceIdentifier>urn:sequence_identifier:temp:85bd850d-60af-4b33-b296-
9eace8ab1b0a</ns21:SequenceIdentifier>
                            <ns21:VDIkind>RVDI</ns21:VDIkind>
                            <ns21:ExpiryDate>2012-10-
01T10:43:19.104+03:00</ns21:ExpiryDate>
                            <ns21:ResourceMetadata>
<ns21:Keyword>ORIGINAL</ns21:Keyword>
<ns21:FieldValue>
    <ns21:Field>TEAM</ns21:Field>
    <ns21:Value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns25="http://www.w3.org/2001/XMLSchema"
xsi:type="ns25:string">AEK</ns21:Value>
</ns21:FieldValue>
<ns21:StructuredData ref="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

```
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:videodescription="http://escom.msh-paris.fr/videodescription.owl#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
  <rdf:Description rdf:about="http://escom.msh-paris.fr/9257b284-7bf1-4272-9542-
732e566b00b9">
    <rdfs:label xml:lang="en">quechua translation</rdfs:label>
    <rdf:type rdf:resource="http://escom.msh-
paris.fr/videodescription.owl#RichDescription"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://escom.msh-paris.fr/92bec06d-f033-4ea6-836b-
bd3c218826d4_Individual">
    <rdfs:label xml:lang="en">English literature is the literature written in the
English language, including literature composed in English by writers not
necessarily from England; for example, Robert Burns was Scottish, James Joyce was
Irish, Joseph Conrad was Polish, Dylan Thomas was Welsh, Edgar Allan Poe was
American, J. R. R. Tolkien was born in the Orange Free State, V.S. Naipaul was
born in Trinidad, and Vladimir Nabokov was Russian, but all are considered
important writers in the history of English literature. In other words, English
literature is as diverse as the varieties and dialects of English spoken around
the world. In academia, the term often labels departments and programmes
practising English studies in secondary and tertiary educational systems. Despite
the variety of authors of English literature, the works of William Shakespeare
remain paramount throughout the English-speaking world.</rdfs:label>
  </rdf:Description>
  <rdf:Description rdf:about="http://escom.msh-paris.fr/df76b56b-77ca-4053-b0dd-
1878f8f01dd1_DescriptionPattern">
    <videodescription:Sign rdf:resource="http://escom.msh-paris.fr/d3651e01-5076-
4969-bbb4-948d8dac8cb4"/>
    <videodescription:Sequence rdf:resource="http://escom.msh-paris.fr/8890602d-
f1bb-402a-ae26-e63389524651"/>
    <videodescription:Scheme rdf:resource="http://escom.msh-paris.fr/57327ae3-
a1e8-461d-a5f1-d465d9b19b6a"/>
    <videodescription:Pattern rdf:resource="http://escom.msh-paris.fr/bb974842-
adf6-41f6-a44c-c30740b36653"/>
    <videodescription:Object rdf:resource="http://escom.msh-paris.fr/f9803178-
d235-4325-988c-0351ea2bdde4"/>
    <videodescription:DataType rdf:resource="http://escom.msh-paris.fr/5b7f6e0c-
49ab-4dd0-b9b9-2d086f1ec2cd"/>
    <rdf:type rdf:resource="http://escom.msh-
paris.fr/videodescription.owl#DescriptionPattern"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://escom.msh-paris.fr/b9312a22-4c0f-4282-910d-
f18c00313e35">
    <videodescription:hasDescription rdf:resource="http://escom.msh-
paris.fr/fd7dcdea-75ae-4d79-a582-cd411dcc91ab"/>
    <videodescription:Kind rdf:resource="http://escom.msh-paris.fr/7217ab7b-5a2a-
4eb6-aa64-a77b38a37b3b"/>
    <rdf:type rdf:resource="http://escom.msh-
paris.fr/videodescription.owl#Rhetoric"/>
  </rdf:Description>
</rdf:RDF>
</ns21:StructuredData>
                            </ns21:ResourceMetadata>
                        </ns21:CONVERGENCEMetadata>
                    </didl_msx:StructuredData>
                </didl_msx:Metadata>
            </mpegm-didl:Statement>
```

```
        </mpegm-didl:Descriptor>
    </mpegm-didl:Item>
</mpegm-didl:DIDL>
```

12.1.2 PVDI creation

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<mpegm-didl:DIDL              xmlns:didmodel="urn:mpeg:mpeg21:2002:02-DIDMODEL-NS"
xmlns:didl_mp21="urn:mpeg:mpeg21:2002:02-DIDL-NS"
xmlns:ipmpinfo="urn:mpeg:mpeg21:2004:01-IPMPINFO-NS"
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
xmlns:mpegmipmpdidl="urn:mpeg:mpegM:schema:08-IPMPDIDL-NS:2012"       xmlns:mpegm-
didl="urn:mpeg:mpegM:schema:06-didl-NS:2012"
xmlns:didl_msx="urn:mpeg:maf:schema:mediastreaming:DIDLextensions"
xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS"   xmlns:er="urn:mpeg:mpeg21:2005:01-ERL-
NS"            xmlns:rel-r="urn:mpeg:mpeg21:2003:01-REL-R-NS"            xmlns:rel-
m1x="urn:mpeg:mpeg21:2005:01-REL-M1X-NS"     xmlns:rel-sx="urn:mpeg:mpeg21:2003:01-
REL-SX-NS"            xmlns:ipmpmsg="urn:mpeg:mpegB:schema:IPMP-XML-MESSAGES:2007"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"                           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:ipmpinfo_msx="urn:mpeg:maf:Schema:mediastreaming:IPMPINFOextensions:2007"
xmlns:mpeg4ipmp="urn:mpeg:mpeg4:IPMPSchema:2002"
xmlns:mpeg7_v3="urn:mpeg:mpeg7:schema:2004"
xmlns:ns21="urn:conv:metadata:schema:2011"   xmlns:ns22="urn:mpeg:mpqf:schema:2008"
xmlns:ns23="urn:conv:relation:schema:2011">
    <mpegm-didl:Item id="A0">
        <mpegm-didl:Descriptor>
            <mpegm-didl:Statement>
                <dii:Identifier>urn:eu:CONVERGENCE:server1\99fd1738-c4b5-49a4-
a53e-3712a277a1e8</dii:Identifier>
            </mpegm-didl:Statement>
        </mpegm-didl:Descriptor>
        <mpegm-didl:Descriptor>
            <mpegm-didl:Statement>
                <didl_msx:Metadata>
                    <didl_msx:StructuredData ref="urn:conv:metadata:schema:2011">
                        <ns21:CONVERGENCEMetadata>

<ns21:SequenceIdentifier>urn:sequence_identifier:temp:59087dbf-4448-4da7-9f27-
dcf4f7319d74</ns21:SequenceIdentifier>
                            <ns21:VDIkind>PVDI</ns21:VDIkind>
                            <ns21:ExpiryDate>2012-10-
01T10:47:58.957+03:00</ns21:ExpiryDate>
                            <ns21:PublicationMetadata>
<ns21:Dimension>
    <ns21:FractalAlgebra>
        <ns21:Fractal>analysis</ns21:Fractal>
    </ns21:FractalAlgebra>
</ns21:Dimension>
<ns21:RVDIid>urn:eu:CONVERGENCE:server1\ad9d8160-4a6d-4b67-850f-
d4f03869ea28</ns21:RVDIid>
<ns21:Keyword>ORIGINAL</ns21:Keyword>
<ns21:FieldValue>
    <ns21:Field>TEAM</ns21:Field>
    <ns21:Value          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xmlns:ns25="http://www.w3.org/2001/XMLSchema"
xsi:type="ns25:string">AEK</ns21:Value>
</ns21:FieldValue>
<ns21:StructuredData ref="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:RDF                xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:videodescription="http://escom.msh-paris.fr/videodescription.owl#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
        <rdf:Description  rdf:about="http://escom.msh-paris.fr/9257b284-7bf1-4272-
9542-732e566b00b9">
            <rdfs:label xml:lang="en">quechua translation</rdfs:label>
            <rdf:type                           rdf:resource="http://escom.msh-
paris.fr/videodescription.owl#RichDescription"/>
  </rdf:Description>
        <rdf:Description  rdf:about="http://escom.msh-paris.fr/92bec06d-f033-4ea6-
836b-bd3c218826d4_Individual">
            <rdfs:label xml:lang="en">English literature is the literature written
in the English language, including literature composed in English by writers not
necessarily from England; for example, Robert Burns was Scottish, James Joyce was
Irish, Joseph Conrad was Polish, Dylan Thomas was Welsh, Edgar Allan Poe was
American, J. R. R. Tolkien was born in the Orange Free State, V.S. Naipaul was
born in Trinidad, and Vladimir Nabokov was Russian, but all are considered
important writers in the history of English literature. In other words, English
literature is as diverse as the varieties and dialects of English spoken around
the world. In academia, the term often labels departments and programmes
practising English studies in secondary and tertiary educational systems. Despite
the variety of authors of English literature, the works of William Shakespeare
remain paramount throughout the English-speaking world.</rdfs:label>
  </rdf:Description>
        <rdf:Description  rdf:about="http://escom.msh-paris.fr/9c33aaa5-f16c-4f27-
be6f-c7c60bfe157e_DescriptionPattern">
            <videodescription:Sign                 rdf:resource="http://escom.msh-
paris.fr/5c694344-c2e1-4efc-ba95-3746908229f1"/>
            <videodescription:Sequence             rdf:resource="http://escom.msh-
paris.fr/8890602d-f1bb-402a-ae26-e63389524651"/>
            <videodescription:Scheme               rdf:resource="http://escom.msh-
paris.fr/8dbcf418-07f4-474b-8c2a-071c302a93a8"/>
            <videodescription:Pattern              rdf:resource="http://escom.msh-
paris.fr/3f5f0686-b013-47c7-ac28-c713a51f34f3"/>
            <videodescription:Object               rdf:resource="http://escom.msh-
paris.fr/5d2276d5-f625-4a7d-9703-d7e67b7bcb31"/>
            <videodescription:DataType             rdf:resource="http://escom.msh-
paris.fr/0665bf9a-cca1-44cb-8b6f-45af1a01703a"/>
            <rdf:type                              rdf:resource="http://escom.msh-
paris.fr/videodescription.owl#DescriptionPattern"/>
  </rdf:Description>


</rdf:RDF>
</ns21:StructuredData>
                            </ns21:PublicationMetadata>
                        </ns21:CONVERGENCEMetadata>
                    </didl_msx:StructuredData>
                </didl_msx:Metadata>
            </mpegm-didl:Statement>
        </mpegm-didl:Descriptor>
    </mpegm-didl:Item>
```

```
</mpegm-didl:DIDL>
```

12.1.3 SVDI creation and parsing

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<mpegm-didl:DIDL              xmlns:didmodel="urn:mpeg:mpeg21:2002:02-DIDMODEL-NS"
xmlns:didl_mp21="urn:mpeg:mpeg21:2002:02-DIDL-NS"
xmlns:ipmpinfo="urn:mpeg:mpeg21:2004:01-IPMPINFO-NS"
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
xmlns:mpegmipmpdidl="urn:mpeg:mpegM:schema:08-IPMPDIDL-NS:2012"       xmlns:mpegm-
didl="urn:mpeg:mpegM:schema:06-didl-NS:2012"
xmlns:didl_msx="urn:mpeg:maf:schema:mediastreaming:DIDLextensions"
xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS"   xmlns:er="urn:mpeg:mpeg21:2005:01-ERL-
NS"            xmlns:rel-r="urn:mpeg:mpeg21:2003:01-REL-R-NS"            xmlns:rel-
m1x="urn:mpeg:mpeg21:2005:01-REL-M1X-NS"     xmlns:rel-sx="urn:mpeg:mpeg21:2003:01-
REL-SX-NS"            xmlns:ipmpmsg="urn:mpeg:mpegB:schema:IPMP-XML-MESSAGES:2007"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"                           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:ipmpinfo_msx="urn:mpeg:maf:Schema:mediastreaming:IPMPINFOextensions:2007"
xmlns:mpeg4ipmp="urn:mpeg:mpeg4:IPMPSchema:2002"
xmlns:mpeg7_v3="urn:mpeg:mpeg7:schema:2004"
xmlns:ns21="urn:conv:metadata:schema:2011"   xmlns:ns22="urn:mpeg:mpqf:schema:2008"
xmlns:ns23="urn:conv:relation:schema:2011">
    <mpegm-didl:Item id="A0">
        <mpegm-didl:Descriptor>
            <mpegm-didl:Statement>
                <dii:Identifier>urn:eu:CONVERGENCE:server1\53c42d17-3a5b-481b-
b71b-d3bb1608e954</dii:Identifier>
            </mpegm-didl:Statement>
        </mpegm-didl:Descriptor>
        <mpegm-didl:Descriptor>
            <mpegm-didl:Statement>
                <didl_msx:Metadata>
                    <didl_msx:StructuredData ref="urn:conv:metadata:schema:2011">
                        <ns21:CONVERGENCEMetadata>

<ns21:SequenceIdentifier>urn:sequence_identifier:temp:9f8910f5-50dd-44c8-afec-
09a53bef08e2</ns21:SequenceIdentifier>
                            <ns21:VDIkind>SVDI</ns21:VDIkind>
                            <ns21:ExpiryDate>2012-10-
01T10:47:56.965+03:00</ns21:ExpiryDate>
                            <ns21:SubscriptionMetadata>
<ns21:Dimension>
    <ns21:FractalAlgebra>
        <ns21:Fractal>analysis</ns21:Fractal>
    </ns21:FractalAlgebra>
</ns21:Dimension>
<ns21:Query>
    <ns22:QueryCondition>
        <ns22:Condition      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns22:OR">
            <ns22:Condition xsi:type="ns22:Equal">
                <ns22:StringField>FIELD</ns22:StringField>
                <ns22:StringValue>VALUE</ns22:StringValue>
            </ns22:Condition>
            <ns22:Condition xsi:type="ns22:QueryBySPARQL">
```

```
                <ns22:SPARQL>SELECT  ?x
WHERE
  {   ?x      &lt;http://escom.msh-paris.fr/videodescription.owl#hasReferences&gt;
&lt;http://escom.msh-paris.fr/531c1582-5950-4a2a-aab2-f428c2172212&gt; . }
</ns22:SPARQL>
            </ns22:Condition>
        </ns22:Condition>
    </ns22:QueryCondition>
</ns21:Query>
                            </ns21:SubscriptionMetadata>
                        </ns21:CONVERGENCEMetadata>
                    </didl_msx:StructuredData>
                </didl_msx:Metadata>
            </mpegm-didl:Statement>
        </mpegm-didl:Descriptor>
        <mpegm-didl:Item id="B0">
            <mpegm-didl:Descriptor>
                <mpegm-didl:Statement>
                    <dii:Identifier>identifier:contentelement:temp:79684509-9ada-
4b20-a013-d63532b33d40</dii:Identifier>
                </mpegm-didl:Statement>
            </mpegm-didl:Descriptor>
            <mpegm-didl:Descriptor>
                <mpegm-didl:Statement>
                    <er:ERR>
                        <er:ERSpecification>
                            <er:ERDeliverySpecification>
<er:DITransportService>
    <rel-r:serviceReference>
        <rel-m1x:serviceLocation>
            <rel-m1x:url>peer6.peers.ict-CONVERGENCE.eu/er</rel-m1x:url>
        </rel-m1x:serviceLocation>
    </rel-r:serviceReference>
</er:DITransportService>
                            </er:ERDeliverySpecification>
                            <er:EmbeddedERR>
<er:ERRReference>urn:err_identifier:temp:03a18886-e39d-4bce-9eec-
c1d5b9391148</er:ERRReference>
                            </er:EmbeddedERR>
                        </er:ERSpecification>
                        <er:EventConditionDescriptor>
                            <er:DIOperationCondition>
<er:DIOperationEvent>
    <er:Operation>match</er:Operation>
</er:DIOperationEvent>
                            </er:DIOperationCondition>
                        </er:EventConditionDescriptor>
                    </er:ERR>
                </mpegm-didl:Statement>
            </mpegm-didl:Descriptor>
        </mpegm-didl:Item>
    </mpegm-didl:Item>
</mpegm-didl:DIDL>
```

12.2 License Tool tests

12.2.1 License creation and parsing

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rel-r:license     xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"     xmlns:mpegm-
didl="urn:mpeg:mpegM:schema:06-didl-NS:2012"
xmlns:didmodel="urn:mpeg:mpeg21:2002:02-DIDMODEL-NS"
xmlns:didl_mp21="urn:mpeg:mpeg21:2002:02-DIDL-NS"
xmlns:ipmpinfo="urn:mpeg:mpeg21:2004:01-IPMPINFO-NS"
xmlns:mpegmipmpdidl="urn:mpeg:mpegM:schema:08-IPMPDIDL-NS:2012"
xmlns:didl_msx="urn:mpeg:maf:schema:mediastreaming:DIDLextensions"
xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS"   xmlns:er="urn:mpeg:mpeg21:2005:01-ERL-
NS"              xmlns:rel-r="urn:mpeg:mpeg21:2003:01-REL-R-NS"            xmlns:rel-
m1x="urn:mpeg:mpeg21:2005:01-REL-M1X-NS"     xmlns:rel-sx="urn:mpeg:mpeg21:2003:01-
REL-SX-NS"                               xmlns:mpeg7_v3="urn:mpeg:mpeg7:schema:2004"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"                             xmlns:rel-
m2x="urn:mpeg:mpeg21:2006:01-REL-M2X-NS"     xmlns:rel-mx="urn:mpeg:mpeg21:2003:01-
REL-MX-NS"                        xmlns:rel-m3x="urn:mpeg:mpeg21:2006:01-REL-M3X-NS"
xmlns:ipmpmsg="urn:mpeg:mpegB:schema:IPMP-XML-MESSAGES:2007"
xmlns:ipmpinfo_msx="urn:mpeg:maf:Schema:mediastreaming:IPMPINFOextensions:2007"
xmlns:mpeg4ipmp="urn:mpeg:mpeg4:IPMPSchema:2002"
xmlns:fru="urn:mpeg:mpegB:schema:FragmentRequestUnits:2007"
xmlns:ns22="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:ns23="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:mpegmipmpinfo="urn:mpeg:mpegM:schema:07-IPMPINFO-NS:2012">
    <rel-r:inventory>
        <rel-r:keyHolder licensePartId="endUserJohn">
            <rel-r:info>
                <dsig:KeyValue>
                    <dsig:RSAKeyValue>

<dsig:Modulus>YTFoa2QwVmFOMDV1Y1RaNVlqTXlkRkZsUWtsUVIyZHVlV1lyVW1KR1ZXUnVkSEJWWXps
UWNqbFNVMGx4U0d3MlRXOW1iR1V5YlRobWQwd3dWa3BhZDBGeVRVdDRURWxWVlhNNWN3b3pNV1pFYkhOU2
JraDNQVDA9</dsig:Modulus>
                        <dsig:Exponent>UVZGQ1FnPT0=</dsig:Exponent>
                    </dsig:RSAKeyValue>
                </dsig:KeyValue>
            </rel-r:info>
        </rel-r:keyHolder>
        <rel-r:keyHolder licensePartId="endUserJack">
            <rel-r:info>
                <dsig:KeyValue>
                    <dsig:RSAKeyValue>

<dsig:Modulus>YTFoa2QwVmFOMDV1Y1RaNVlqTXlkRkZsUWtsUVIyZHVlV1lyVW1KR1ZXUnVkSEJWWXps
UWNqbFNVMGx4U0d3MlRXOW1iR1V5YlRobWQwd3dWa3BhZDBGeVRVdDRURWxWVlhNNWN3b3pNV1pFYkhOU2
JraDNQVDA9</dsig:Modulus>
                        <dsig:Exponent>UVZGQ1FnPT0=</dsig:Exponent>
                    </dsig:RSAKeyValue>
                </dsig:KeyValue>
            </rel-r:info>
        </rel-r:keyHolder>
        <rel-mx:diReference licensePartId="diReference">
            <rel-mx:identifier>http://www.onlinemusic.com/mySong.mp3</rel-
mx:identifier>
        </rel-mx:diReference>
        <rel-r:validityInterval>
            <rel-r:notBefore>2006-01-01T00:00:00</rel-r:notBefore>
            <rel-r:notAfter>2006-12-31T23:59:59</rel-r:notAfter>
        </rel-r:validityInterval>
    </rel-r:inventory>
    <rel-r:grant>
        <rel-r:forAll varName="anyEndUser"/>
        <rel-r:forAll varName="anyProducer"/>
        <rel-r:forAll varName="anyAggregator"/>
        <rel-r:principal varRef="anyProducer"/>
        <rel-r:obtain/>
        <rel-r:validityInterval licensePartIdRef="oneYear">
            <rel-r:notBefore>2006-01-01T00:00:00</rel-r:notBefore>
            <rel-r:notAfter>2006-12-31T23:59:59</rel-r:notAfter>
        </rel-r:validityInterval>
    </rel-r:grant>
    <rel-r:grantGroup>
        <rel-r:grant>
            <rel-r:principal varRef="anyProducer"/>
            <rel-mx:play/>
            <rel-mx:diReference licensePartIdRef="diReference"/>
            <rel-r:validityInterval licensePartIdRef="oneYear">
                <rel-r:notBefore>2006-01-01T00:00:00</rel-r:notBefore>
                <rel-r:notAfter>2006-12-31T23:59:59</rel-r:notAfter>
            </rel-r:validityInterval>
        </rel-r:grant>
        <rel-r:grant>
            <rel-r:principal varRef="anyProducer"/>
            <rel-m1x:governedCopy/>
            <rel-mx:diReference licensePartIdRef="diReference"/>
            <rel-r:validityInterval licensePartIdRef="oneYear">
                <rel-r:notBefore>2006-01-01T00:00:00</rel-r:notBefore>
                <rel-r:notAfter>2006-12-31T23:59:59</rel-r:notAfter>
            </rel-r:validityInterval>
        </rel-r:grant>
        <rel-r:grant>
            <rel-r:principal varRef="anyProducer"/>
            <rel-m3x:governedAggregate/>
            <rel-mx:diReference licensePartIdRef="diReference"/>
            <rel-r:validityInterval licensePartIdRef="oneYear">
                <rel-r:notBefore>2006-01-01T00:00:00</rel-r:notBefore>
                <rel-r:notAfter>2006-12-31T23:59:59</rel-r:notAfter>
            </rel-r:validityInterval>
        </rel-r:grant>
        <rel-r:grant>
            <rel-r:principal varRef="anyEndUser"/>
            <rel-mx:play/>
            <rel-mx:diReference licensePartIdRef="diReference"/>
        </rel-r:grant>
    </rel-r:grantGroup>
</rel-r:license>
```

12.3 Metadata Tool tests

12.3.1 Metadata creation

```
<ns21:StructuredData ref="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:j.0="http://uti.com/" xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
<rdf:Description rdf:about="http://uti.com/offerA">
<j.0:expiration_date
rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2012-10-
21T21:49:50.353Z</j.0:expiration_date>
<j.0:price rdf:datatype="http://www.w3.org/2001/XMLSchema#int">300</j.0:price>
<j.0:interest_category rdf:resource="http://uti.com/product_categoryA"/>
<j.0:interest_product rdf:resource="http://uti.com/productA"/>
</rdf:Description>
</rdf:RDF>
</ns21:StructuredData>
```

12.4 Revoke Tool tests

12.4.1 VDI input file

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<mpegm-didl:DIDL xmlns:didmodel="urn:mpeg:mpeg21:2002:02-DIDMODEL-NS">
<mpegm-didl:Item id="A0">
        <mpegm-didl:Descriptor>
            <mpegm-didl:Statement>
                <dii:Identifier>urn:eu:CONVERGENCE:server1/66096a69-3b80-
45e8-b340-9b7b7b7d3f63</dii:Identifier>
            </mpegm-didl:Statement>
        </mpegm-didl:Descriptor>
        <mpegm-didl:Descriptor>
            <mpegm-didl:Statement>
                <didl_msx:Metadata>
                    <didl_msx:StructuredData
ref="urn:conv:metadata:schema:2011">
                        <ns21:CONVERGENCEMetadata>

<ns21:SequenceIdentifier>urn:sequence_identifier:temp:bc2b6a09-ba8b-43a4-
bb27-646256576932</ns21:SequenceIdentifier>
                            <ns21:VDIkind>RVDI</ns21:VDIkind>
                            <ns21:ExpiryDate>2012-06-
12T16:08:04.191+03:00</ns21:ExpiryDate>
                            <ns21:ResourceMetadata>
<ns21:Keyword>Test</ns21:Keyword>
                            </ns21:ResourceMetadata>
                        </ns21:CONVERGENCEMetadata>
                    </didl_msx:StructuredData>
                </didl_msx:Metadata>
            </mpegm-didl:Statement>
        </mpegm-didl:Descriptor>
        <mpegm-didl:Item id="B0">
            <mpegm-didl:Descriptor>
                <mpegm-didl:Statement>
```

```
                    <dii:Identifier>temp_inner_level_id</dii:Identifier>
                </mpegm-didl:Statement>
            </mpegm-didl:Descriptor>
            <mpegm-didl:Component>
                <mpegm-didl:Resource
ref="file:/var/folders/jq/jqBGHUw4G0C1IlAPh04Nrk+++TI/-Tmp-/upload-
00fe53b0-b561-4143-aee1-2a9a0b628610-video-3994725604289010107.mp4"
mimeType="video/mp4"/>
            </mpegm-didl:Component>
        </mpegm-didl:Item>
    </mpegm-didl:Item>
</mpegm-didl:DIDL>
```

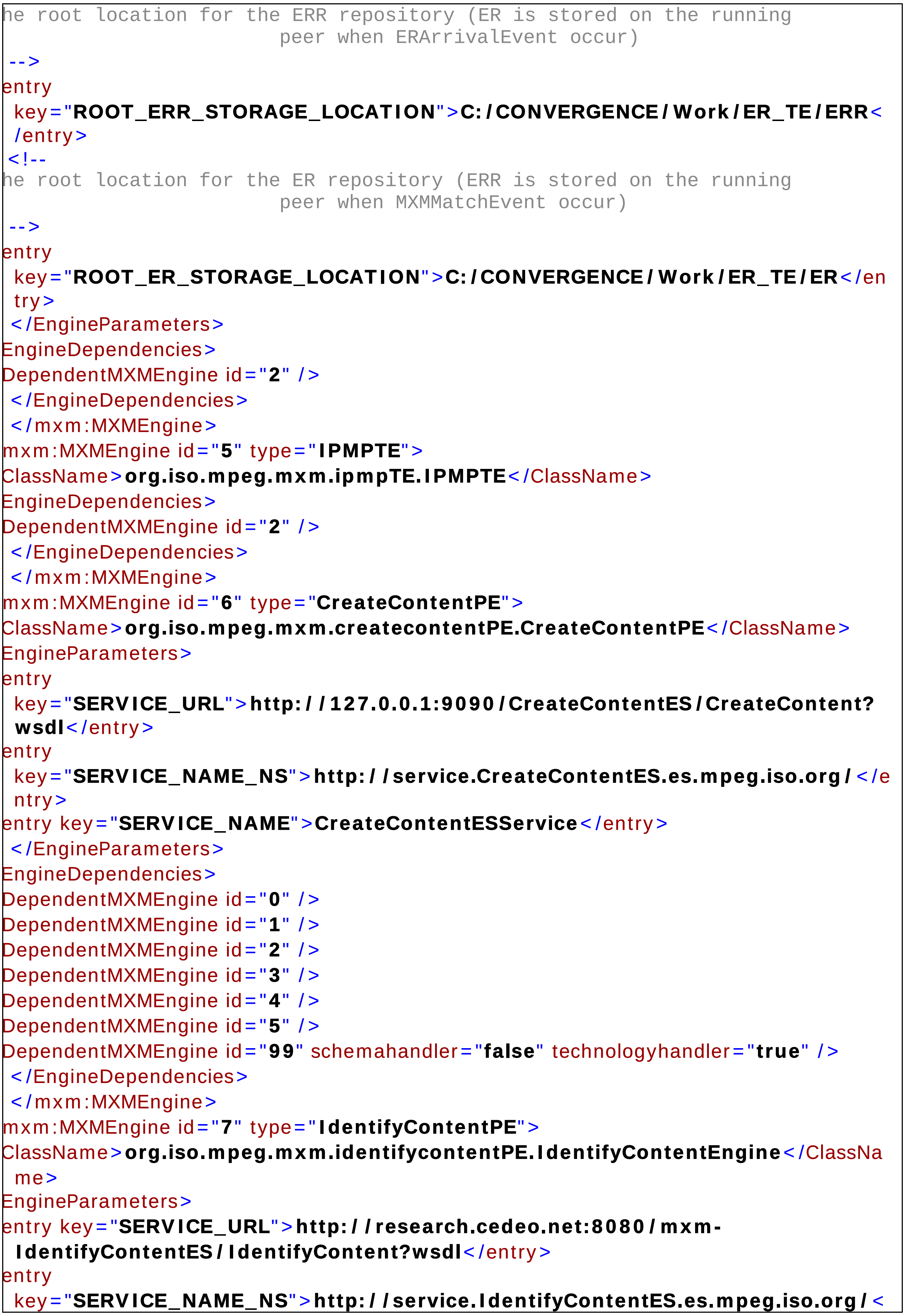
12.4.2 VDI output file

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<mpegm-didl:DIDL xmlns:didmodel="urn:mpeg:mpeg21:2002:02-DIDMODEL-NS" >
<mpegm-didl:Item id="A0">
        <mpegm-didl:Descriptor>
            <mpegm-didl:Statement>
                <dii:Identifier>urn:eu:CONVERGENCE:server1/66096a69-3b80-
45e8-b340-9b7b7b7d3f63</dii:Identifier>
            </mpegm-didl:Statement>
        </mpegm-didl:Descriptor>
        <mpegm-didl:Descriptor>
            <mpegm-didl:Statement>
                <didl_msx:Metadata>
                    <didl_msx:StructuredData
ref="urn:conv:metadata:schema:2011">
                        <ns21:CONVERGENCEMetadata>

<ns21:SequenceIdentifier>urn:sequence_identifier:temp:bc2b6a09-ba8b-43a4-
bb27-646256576932</ns21:SequenceIdentifier>
                            <ns21:VDIkind>RVDI</ns21:VDIkind>
                            <ns21:ExpiryDate>2012-06-
12T16:08:04.191+03:00</ns21:ExpiryDate>
                            <ns21:ResourceMetadata>
<ns21:Keyword>Test</ns21:Keyword>
                            </ns21:ResourceMetadata>
                        </ns21:CONVERGENCEMetadata>
                    </didl_msx:StructuredData>
                </didl_msx:Metadata>
            </mpegm-didl:Statement>
        </mpegm-didl:Descriptor>
        <mpegm-didl:Item id="B0">
            <mpegm-didl:Descriptor>
                <mpegm-didl:Statement>
                    <dii:Identifier>temp_inner_level_id</dii:Identifier>
                </mpegm-didl:Statement>
            </mpegm-didl:Descriptor>
            <mpegm-didl:Component>
            </mpegm-didl:Component>
        </mpegm-didl:Item>
    </mpegm-didl:Item>
</mpegm-didl:DIDL>
```

13 ANNEX C – Sample MXM Configuration file

```
 <?xml version="1.0" encoding="UTF-8" ?>
mxm:MXMConfiguration xmlns:mxm="org:iso:mpeg:mxm:configuration:schema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="org:iso:mpeg:mxm:configuration:schema
 mxmconfiguration.xsd">
mxm:MXMEngine id="0" type="MetadataTE">
ClassName>org.iso.mpeg.mxm.metadataTE.MetadataTE</ClassName>
 </mxm:MXMEngine>
mxm:MXMEngine id="1" type="DigitalItemTE">
ClassName>eu.CONVERGENCE.middleware.vdiTE.VDITE</ClassName>
EngineDependencies>
DependentMXMEngine id="0" />
DependentMXMEngine id="5" />
DependentMXMEngine id="3" />
DependentMXMEngine id="4" />
 </EngineDependencies>
 </mxm:MXMEngine>
mxm:MXMEngine id="2" type="RELTE">
ClassName>org.iso.mpeg.mxm.relTE.RELEngine</ClassName>
EngineDependencies>
DependentMXMEngine id="3" />
 </EngineDependencies>
 </mxm:MXMEngine>
mxm:MXMEngine id="3" type="SecurityTE">
ClassName>org.iso.mpeg.mxm.securityTE.SecurityTE</ClassName>
EngineParameters>
entry key="enableSmartCard">false</entry>
entry key="cardProtocol">1</entry>
entry key="cardTerminalName">OMNIKEY CardMan</entry>
 <!--
engineFocus is one of: Application, ServiceProvider or IdentityProvider
 -->
 <!--
if you don't know what to put in here, choose 'Application'
 -->
entry key="engineFocus">Application</entry>
entry key="mySqlUrl">localhost:3306</entry>
entry key="mySqlUsername">root</entry>
entry key="mySqlPassword" />
entry key="dataBaseName">app_sec_db</entry>
 </EngineParameters>
 </mxm:MXMEngine>
mxm:MXMEngine id="4" type="EventReportTE">
ClassName>eu.CONVERGENCE.middleware.eventreportTE.kernel.conet.EventRepo
 rtTE</ClassName>
EngineParameters>
entry key="PEER_ID">AZ</entry>
entry key="ER_SAP">peerAZ.peers.ict-CONVERGENCE.eu/er</entry>
 <!--
```



```
he root location for the ERR repository (ER is stored on the running
                         peer when ERArrivalEvent occur)
-->
entry
 key="ROOT_ERR_STORAGE_LOCATION">C:/CONVERGENCE/Work/ER_TE/ERR<
 /entry>
<!--
he root location for the ER repository (ERR is stored on the running
                         peer when MXMMatchEvent occur)
-->
entry
 key="ROOT_ER_STORAGE_LOCATION">C:/CONVERGENCE/Work/ER_TE/ER</en
 try>
</EngineParameters>
EngineDependencies>
DependentMXMEngine id="2" />
</EngineDependencies>
</mxm:MXMEngine>
mxm:MXMEngine id="5" type="IPMPTE">
ClassName>org.iso.mpeg.mxm.ipmpTE.IPMPTE</ClassName>
EngineDependencies>
DependentMXMEngine id="2" />
</EngineDependencies>
</mxm:MXMEngine>
mxm:MXMEngine id="6" type="CreateContentPE">
ClassName>org.iso.mpeg.mxm.createcontentPE.CreateContentPE</ClassName>
EngineParameters>
entry
 key="SERVICE_URL">http://127.0.0.1:9090/CreateContentES/CreateContent?
 wsdl</entry>
entry
 key="SERVICE_NAME_NS">http://service.CreateContentES.es.mpeg.iso.org/</e
 ntry>
entry key="SERVICE_NAME">CreateContentESService</entry>
</EngineParameters>
EngineDependencies>
DependentMXMEngine id="0" />
DependentMXMEngine id="1" />
DependentMXMEngine id="2" />
DependentMXMEngine id="3" />
DependentMXMEngine id="4" />
DependentMXMEngine id="5" />
DependentMXMEngine id="99" schemahandler="false" technologyhandler="true" />
</EngineDependencies>
</mxm:MXMEngine>
mxm:MXMEngine id="7" type="IdentifyContentPE">
ClassName>org.iso.mpeg.mxm.identifycontentPE.IdentifyContentEngine</ClassNa
 me>
EngineParameters>
entry key="SERVICE_URL">http://research.cedeo.net:8080/mxm-
 IdentifyContentES/IdentifyContent?wsdl</entry>
entry
 key="SERVICE_NAME_NS">http://service.IdentifyContentES.es.mpeg.iso.org/<
 /entry>
entry key="SERVICE_NAME">IdentifyContentESService</entry>
 </EngineParameters>
EngineDependencies>
DependentMXMEngine id="0" />
DependentMXMEngine id="1" />
DependentMXMEngine id="2" />
DependentMXMEngine id="3" />
DependentMXMEngine id="4" />
DependentMXMEngine id="5" />
DependentMXMEngine id="99" schemahandler="false" technologyhandler="true" />
 </EngineDependencies>
 </mxm:MXMEngine>
mxm:MXMEngine id="8" type="CreateLicensePE">
ClassName>org.iso.mpeg.mxm.createlicensePE.CreateLicensePE</ClassName>
EngineParameters>
entry
 key="SERVICE_URL">http://147.102.9.117:9090/CreateLicenseES/CreateLicen
 se?wsdl</entry>
entry
 key="SERVICE_NAME_NS">http://service.CreateLicenseES.es.mpeg.iso.org/</e
 ntry>
entry key="SERVICE_NAME">CreateLicenseESService</entry>
 </EngineParameters>
EngineDependencies>
DependentMXMEngine id="0" />
DependentMXMEngine id="1" />
DependentMXMEngine id="2" />
DependentMXMEngine id="3" />
DependentMXMEngine id="4" />
DependentMXMEngine id="5" />
DependentMXMEngine id="99" schemahandler="false" technologyhandler="true" />
 </EngineDependencies>
 </mxm:MXMEngine>
mxm:MXMEngine id="9" type="CONVERGENCEMetadataTE">
ClassName>eu.CONVERGENCE.middleware.metadataTE.CONVERGENCEMetadataTE
 </ClassName>
EngineDependencies>
DependentMXMEngine id="1" />
DependentMXMEngine id="10" />
 </EngineDependencies>
 </mxm:MXMEngine>
mxm:MXMEngine id="10" type="MpqfTE">
ClassName>eu.CONVERGENCE.middleware.mpqfTE.MpqfTE</ClassName>
 </mxm:MXMEngine>
mxm:MXMEngine id="11" type="MatchTE">
ClassName>eu.CONVERGENCE.middleware.matchTE.MatchTE</ClassName>
 </mxm:MXMEngine>
mxm:MXMEngine id="12" type="CDSTE">
ClassName>eu.CONVERGENCE.middleware.cdsTE.CdsTE</ClassName>
EngineParameters>
entry
```

```
key="LOCAL_REPOSITORY_PATH">C:/CONVERGENCE/Work/CDS_TE</entry>
</EngineParameters>
</mxm:MXMEngine>
mxm:MXMEngine id="13" type="ConetTE">
ClassName>eu.CONVERGENCE.middleware.conetTe.ConetTE</ClassName>
EngineParameters>
entry key="CONET_SERVLET_BASE_URL">http://147.102.9.7/conet/</entry>
entry
key="SAP_ADVERTISE_CFG">C:\CONVERGENCE\Work\CONET_TE\cfg\advertiser-sip.cfg</entry>
entry key="SAP_SEND_CFG">C:\CONVERGENCE\Work\CONET_TE\cfg\sender-sip.cfg</entry>
</EngineParameters>
</mxm:MXMEngine>
mxm:MXMEngine id="14" type="OverlayTE">
ClassName>eu.CONVERGENCE.middleware.overlayTE.kernel.conet.OverlayTE</ClassName>
EngineParameters>
entry
key="PUBLICATION_MESSAGES_LOCATION">C:/CONVERGENCE/Work/OVERLAY_TE/publicationMessages/</entry>
entry
key="READY_PUBLICATION_MESSAGES_LOCATION">C:/CONVERGENCE/Work/OVERLAY_TE/publicationMessages/ready/</entry>
entry
key="SUBSCRIPTION_MESSAGES_LOCATION">C:/CONVERGENCE/Work/OVERLAY_TE/subscriptionMessages/</entry>
entry
key="READY_SUBSCRIPTION_MESSAGES_LOCATION">C:/CONVERGENCE/Work/OVERLAY_TE/subscriptionMessages/ready/</entry>
entry
key="REVOCATION_MESSAGES_LOCATION">C:/CONVERGENCE/Work/OVERLAY_TE/revocationMessages/</entry>
entry
key="READY_REVOCATION_MESSAGES_LOCATION">C:/CONVERGENCE/Work/OVERLAY_TE/revocationMessages/ready/</entry>
entry
key="REGISTRY_LOCATION">C:/CONVERGENCE/Work/OVERLAY_TE/registry/</entry>
entry
key="READY_REGISTRY_LOCATION">C:/CONVERGENCE/Work/OVERLAY_TE/registry/ready/</entry>
entry
key="PEER_PROPERITES_LOCATION">C:/CONVERGENCE/Work/OVERLAY_TE/peer_configuration.xml</entry>
entry key="PEER_ID">peerAZ</entry>
entry key="OVERLAY_SAP">peerAZ.peers.ict-CONVERGENCE.eu/overlay</entry>
</EngineParameters>
</mxm:MXMEngine>
mxm:MXMEngine id="99" type="Orchestrator">
ClassName>eu.CONVERGENCE.middleware.orchestratorTE.CONVERGENCEOrchestratorTE</ClassName>
EngineDependencies>
```

```
DependentMXMEngine id="0" />
DependentMXMEngine id="1" />
DependentMXMEngine id="2" />
DependentMXMEngine id="3" />
DependentMXMEngine id="4" />
DependentMXMEngine id="5" />
DependentMXMEngine id="6" />
DependentMXMEngine id="7" />
DependentMXMEngine id="8" />
DependentMXMEngine id="9" />
DependentMXMEngine id="10" />
DependentMXMEngine id="11" />
DependentMXMEngine id="12" />
DependentMXMEngine id="13" />
DependentMXMEngine id="14" />
 </EngineDependencies>
 </mxm:MXMEngine>
 </mxm:MXMConfiguration>
```