



CORES

Contract: IST-2001-35273

A forum on shared metadata vocabularies

Web Interface for Self Registration

D22

Document number:

CORES-UKOLN-WP2-D22-Final-20021104

General Information

Title	Web interface for self registration
Creator	Rachel Heery, Andras Micsik, Csaba Fulop
Subject-Keywords	schema creation tool, schema registries, annotation, interoperability
Description	This report provides documentation for a schema creation and registration tool to be used within the CORES project. This development work takes forward previous work within the SCHEMAS and MEG projects, and in particular enhances the MEG software with annotation and authentication functionality.
Publisher	PricewaterhouseCoopers
Date	4 November 2002
Type	Text Manuscript
Format	application/MSWord 2000
Identifier	
Document Number	CORES-UKOLN-WP2-D22-Final-20021104
Language	PricewaterhouseCoopers
Rights	European Commission; External distribution via CORES Web site

Dublin Core metadata for this document

```
<?xml version="1.0"?>
<rdf:RDF xml:lang="en"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
<rdf:RDF rdf:resource = " ">
<dc:title> web interface for self registration </dc:title>
<dc:creator> Rachel Heery </dc:creator>
<dc:creator> Andras Micsik, </dc:creator>
<dc:creator> Csaba Fulop </dc:creator>
<dc:subject> Deliverable D22 </dc:subject>
<dc:subject> WP2 </dc:subject>
<dc:description> This report provides documentation for a schema creation
and registration tool to be used within the CORES project. This development
work takes forward previous work within the SCHEMAS and MEG projects, and in
particular enhances the MEG software with annotation and authentication
functionality.</dc:description>
<dc:publisher> PricewaterhouseCoopers </dc:publisher>
<dc:date> 2002-11-04 </dc:date>
<dc:type> Text </dc:type>
<dc:format> application/MSword </dc:format>
<dc:identifier>CORES-UKOLN-WP2-D22-Final-20021104</dc:identifier>
<dc:language> en </dc:language>
<dc:rights> European Commission; External distribution via CORES Web site
</dc:rights>
</rdf:RDF>
```

Table of Contents

1	FUNCTIONAL REQUIREMENTS.....	5
1.1	BACKGROUND.....	5
1.2	SUMMARY OF REQUIREMENTS	5
1.3	PURPOSE OF THE CORES REGISTRY SYSTEM	6
1.4	SCOPE OF THE CORES REGISTRY SYSTEM	8
1.5	STRUCTURE AND USE OF THE CORES SCHEMA CREATION TOOL.....	9
1.5.1	<i>The registry.....</i>	9
1.5.2	<i>The schema creation and registration tool.....</i>	10
1.6	THE CORES SCHEMA CREATION TOOL DATA MODEL	14
1.6.2	<i>Creating and editing schemas using the CORES schema creation tool</i> <i>20</i>	
1.6.3	<i>6. Creating and editing annotations</i>	23
1.7	NOTES AND REFERENCES	23
2	INSTALLATION GUIDE.....	25
2.1	INSTALL GUIDE FOR THE CLIENT	25
2.2	INSTALL GUIDE FOR THE SERVER.....	25
2.3	LOADING THE INITIAL REGISTRY	27
2.4	INSTALL AND CONFIGURATION OF THE WEB INTERFACE.....	27
3	DEVELOPER'S MANUAL	29
3.1	THE CLIENT	29
3.1.1	<i>Architecture.....</i>	29
3.1.2	<i>Packages</i>	30
3.1.3	<i>Classes</i>	30
3.2	THE SERVER.....	31
3.2.1	<i>Architecture.....</i>	31
3.2.2	<i>Modules.....</i>	32
3.2.3	<i>Functions and scripts.....</i>	33
3.2.4	<i>References</i>	34
3.3	WORK LIST: BUG FIXES, CHANGES AND DEVELOPMENTS.....	35
3.3.1	<i>Server</i>	35
3.3.2	<i>Client.....</i>	36
4	USER'S MANUAL.....	38
4.1	THE SERVER (REGISTRY).....	38
4.1.1	<i>Browse Schemas.....</i>	38
4.1.2	<i>Search Schemas</i>	40
4.1.3	<i>Browse Annotations</i>	40
4.1.4	<i>Add/Edit/Delete Annotations</i>	41
4.2	THE CLIENT	42
4.2.1	<i>Create Schemas.....</i>	42
4.3	SEARCH THE REGISTRY	45
4.4	STORE SCHEMAS (SAVE/UPLOAD)	47
4.5	ADD ANNOTATIONS.....	47

1 Functional Requirements

1.1 Background

The ambition of the CORES project is to build consensus on a shared data model for the declaration of element sets and application profiles. Such a shared data model would facilitate the re-use of existing schemas by other services, projects and initiatives. In order to further this ambition, CORES is implementing a schema creation tool which will work interactively with a complementary schema registry. Both will be based on the same data model to enable implementors to create schemas, register the schemas, then make them available for re-use by other implementors.

Development of the CORES schema creation tool and registry build on work carried out within the UK MEG Registry project [\[1\]](#). CORES builds on the data model and functional requirements [\[2\]](#) developed in that project, and enhances the existing software in order to provide an appropriate tool for use by a range of schema implementors. In addition CORES will explore the use of annotations within the Registry to convey information about the deployment and use of element sets and application profiles. Previously, within the SCHEMAS [\[3\]](#) prototype registry, such information had been structured in a more formal manner as activity reports submitted by experts to a central administrator for input to the Registry.

The CORES project will implement a de-centralised, distributed mode whereby more informal comments can be added by authenticated users as 'annotations' to the Registry.

CORES is collaborating closely with the MEG Registry work in the UK and the DCMI Registry work [\[4\]](#) which is being undertaken as part of the international Dublin Core Metadata Initiative. In addition CORES partners have been involved in discussion with the W3C Semantic Web activity [\[5\]](#) regarding the use of RDF Vocabulary Description Language [\[6\]](#) to express schemas. Both the CORES and the MEG Registries build on previous activity within the DESIRE and SCHEMAS projects which established data models for declaring schemas and local usage within a schema registry, and implemented prototype registries [\[7\]](#) [\[8\]](#) [\[9\]](#).

1.2 Summary of Requirements

The primary purpose of the schema creation tool is to provide the means for non-experts to declare their schemas according to the RDF schema specification. The schema creation tool must enable a structured RDF schema to be produced identifying and defining:

- Element Sets;

- the Elements in those Element Sets;
- Application Profiles;
- Element Usages which make up those Application Profiles;
- Encoding Schemes;
- Values within those Encoding Schemes;
- the Agencies who own/create/maintain these resources
- Commentaries (contextual annotations) outlining deployment of the Element Sets, Application Profiles, and Schemes
- links to User Guidelines for the Element Sets, Application profiles or Schemes

The schema creation tool must be suitable for use by non-experts to enter information about these entities. The schema creation tool should allow the owners of this information to create and maintain schemas in a convenient way. These tools should allow the user to focus on the structure and semantics of the descriptions they create and manage and should insulate them from, for example, details of the XML syntax of the schemas. A generic "node and arcs" representation may not be the most useful "view" for the users of this tool, and consideration should be given to presenting table-/text- based views which may be more familiar to this constituency.

The information must be output in a form suitable for registration in the CORES registry, though it may be necessary to supplement the information input by the schema creator in order to record attributes of these resources required for the registry.

The information about these resources must also be stored locally to the schema creation tool in machine-processable form as RDF schemas, which may be distributed on the Web. The owners of schemas may wish to make them available to other applications independently of the SCHEMAS registry, and the tool should support this. The W3C Resource Description Framework (RDF) recommendation provides a general data model for expressing this information, and conventions for representing those descriptions in XML. RDF Schema defines a basic vocabulary which can be used to describe these types of resource.

Devolving the management of this information introduces questions of how to establish "trust" in the information content presented by the registry. Sufficient "administrative metadata" should be recorded and made available so that a registry might ensure it can record by whom assertions were made, and when.

1.3 Purpose of the CORES Registry system

The variety of new and emerging Web based services and initiatives have adopted or developed a number of different metadata **Element Sets** or vocabularies for the description of the resources on which their services are based. CORES intends to encourage sharing of metadata semantics, and will provide a metadata schema registry

[10] to facilitate the 'publication' of metadata schemas. Such a registry is intended to help implementors of information projects and services find out about metadata terms in use (both official definitions and local variations) to encourage harmonisation of metadata usage within particular fields and applications.

In order to facilitate building such a registry the descriptions and definitions of these Element Sets need to be made available in machine-processable form as **Schemas**. The purpose of the schema creation tool is to enable implementors to 'publish' their schemas so that they can be registered, either in the CORES registry or elsewhere. Additionally the schemas may be used in other applications, either locally or remotely. The registry provides a publication environment in which the developers and implementors of education-related metadata schemas can disclose information on schemas and on their use.

The CORES registry is more than a "dictionary" of the Elements described by schemas: fundamental to the registry is the recognition that implementers deploy and adapt "standard" Element Sets in a pragmatic way. While "standard" schemas are widely available, these use-oriented adaptations, which are often localised and service-specific, tend to be less visible. Researchers on schema usage have introduced the idea of the "Application Profile" as a means of capturing this information on adaptations and constraints of Element Set usage. [11]

Such differentiation in the use of schemas brings a cost in terms of interoperability between systems that deploy them, and unnecessary variation should be avoided where possible. Making such variation visible by means of a schema registry

- contributes to the goal of enhancing interoperability between systems which use these schemas,
- reduces confusion for implementers, and
- may help avoid costly duplication of effort by promoting the reuse of existing solutions.

Users of the registry might include:

- Standards creators publishing "standard" Element Sets
- Implementers publishing Application Profiles
- Implementers/developers seeking appropriate Element Sets/Application Profiles
- Metadata instance creators seeking guidance on use of Element Sets and Application Profiles
- Researchers studying schema usage
- Developers studying tools to support schema usage
- Commentators on deployment of Element Sets and Application Profiles

1.4 Scope of the CORES Registry system

The registry will provide access to information on [\[12\]](#)

- Element Sets (i.e. on the Element Sets as units, rather than on their constituent Elements), including information on their intended scope/area of use and their relationship to other Element Sets;
- Elements which make up those Element Sets, including information on the semantics of the Elements and their recommended usage, and any semantic relationships to other Elements in this or other vocabularies (e.g. the relationship described by the DCMI concept of "element refinement" or by RDF Schema as a "sub-property" relation)
- Application Profiles, including information on their intended scope/area of use and their relationship to other Element Sets and Application Profiles;
- Usages of Elements which make up those Application Profiles, including the Element used, any prescription of Encoding Schemes, and other constraints on element use;
- Encoding Schemes, which constrain the value space of Elements, including information on their intended scope/area of use;
- values prescribed by that Encoding Scheme (in practice this is likely to be recorded only where that list consists of a small number (<20?) of values);
- Agencies who own/create/maintain Element Sets, Application Profiles, and Encoding Schemes
- Commentaries allowing for evaluative information to be added about Element Sets, Application Profiles, Elements and Encoding Schemes
- Guidelines for linking to usage guidelines relating to Element Sets, Application Profiles, and Encoding Schemes Application Profiles

This information is stored and made available in machine-processable form as **schemas**. As far as possible, the information above should be expressed using "standard" (or at least widely understood) semantic vocabularies, such as RDF Schema and the Element Sets of the DCMI. It is recognised, however, that some information used by the registry will require the use of additional vocabularies.

One of the limitations of the current SCHEMAS registry is the absence of an interface to allow the owners/maintainers of schemas to manage the relevant entries within the registry. It is not scalable for schemas to be entered into the registry centrally, by hand. Implementors need to create schemas for their Element Sets and Application Profiles in order to submit them to the registry. It should be possible for the schema creators to **store** a copy of that schema, serialised in a form compatible with Web metadata standards, which they can make available - independently of the CORES registry - for reuse by other applications, which may include other metadata schema registries.

1.5 Structure and use of the CORES Schema Creation tool

The primary user community for the CORES schema creation tool are implementors of Web based systems working within EC projects, particularly in areas related to the Semantic Web activity. This document has been drafted with the interests of this community in mind. However, it is hoped that this tool will be of interest to the broader community working with metadata schemas for the description of Web resources, in particular schema researchers, developers and implementers. It would be valuable if, during development, consideration could be given to the potential for reuse of existing tools and future re-purposing of the tool by this broader community.

The CORES registry system can be separated into components:

- The **registry** proper, which will read schemas describing the resources listed above, store that aggregated data, and present an interface to that data which allows a human reader to browse and search it. From this point in this document, references to "the registry" are to this "central" component of the system. The registry will be hosted on a server at UKOLN. The existing SCHEMAS registry software solution, will be migrated to an RDF based Registry solution now in alpha test within the MEG project.
- The **commentary tool** for creating annotations within the Registry. This will be associated with an **authentication mechanism** to validate commentator by email address which will enable the commentator to update their annotations.
- The **schema creation and registration tool**, which will allow a human user to create and edit schemas, and to submit/re-submit them to the registry. It is expected that the tool will be used on PCs, probably mainly in a Windows environment, but it would be highly desirable to ensure that this tool is portable to other environments.

The schema creation/registration tool will need to exchange data with the registry to perform these functions. If the interfaces to the registry can be created in standardised forms and supporting documentation provided, that would enhance the possibility for the reuse of these tools and facilitate interoperability with other tools developed independently.

The commentary tool will be integrated within the Registry.

1.5.1 The registry

Users of the CORES **registry** will include

- Standards creators publishing "standard" Element Sets
- Implementers publishing Application Profiles
- Implementers/developers seeking appropriate Element Sets/Application Profiles

- Metadata instance creators seeking guidance on use of Element Sets and Application Profiles
- Researchers studying schema usage
- Commentators creating annotations associated with Element Sets, Application Profiles, Elements and Schemes

The registry should:

- read/gather schemas - machine-processable forms of descriptions of the resources listed above;
- store and index that aggregated data; present an interface to that data which allows a human reader to browse and search it and navigate the relationships between resources in a convenient fashion;
- provide interfaces which permit the schema creation and registration tool to request information from the registry and to submit new information to the registry enable creation of annotations provide means to browse annotations

The "source" schemas may be distributed at various locations on the Web, and the registry must provide support for re-reading and re-indexing any of those descriptions as they are updated by their owners. Those owners will update their schemas using the schema creation and registration tool.

This devolved model may require some form of basic **authentication/authorisation** mechanism to ensure that the maintainers of this information can edit only those units of information in the registry which they "own". Sufficient "**administrative metadata**" should be recorded and made available so that a user browsing this information can see by whom assertions were made, and when, regarding element sets, application profiles, and annotations.

The schemas contain descriptions of instances of various classes of resource (Element Sets, Elements, Element Usages etc). Specific types of **relationship** exist between instances of these different classes of resource. The interface through which human readers of the registry access this information should make these relationships clear to the reader and should allow them to navigate those relationships, for example, via hypertext links. In displaying information about resources, the registry should apply appropriate "labels" in preference to displaying URIs of properties or resource classes.

1.5.2 The schema creation and registration tool

Users of the **schema creation/registration tool** will include:

- Standards creators publishing "standard" Element Sets
- Implementers publishing Application Profiles
- Developers studying tools to support schema usage
- Standards creators, implementers adding links to user guidelines

- Standards creators, implementers and the registry 'editor' adding commentary

The **schema creation/registration tool** should allow a human user

- to create and edit schemas (descriptions of the resources listed above), and
- to submit/re-submit those schemas to the registry.

The user interface to this tool should allow the user to focus on the structure and semantics of the descriptions they are managing; as far as possible, it should insulate them from the syntax of the machine-readable forms of those descriptions. As in the case of the registry interface, the "labels" of properties and classes should be displayed in preference to their URIs wherever possible. Further, many schema creators within CORES have a limited familiarity with the concepts and terminology used in association with the RDF model, and it will be important to provide interfaces which take that limitation into account. A "view" of an "Element Set" as a table of "Elementss" will be more intuitive than a graphical representation based on nodes and arcs, for example. Ideally the tool would provide the flexibility to render different views for different groups of user.

In constructing an "Application Profile", a core part of the task is to establish relationships to existing descriptions of resources within the registry e.g. a profile "uses" Elements from existing Element Sets and may associate them with existing Encoding Schemes. In such cases, the interface should allow the user to do so in a manner which is intuitive and simple to use, but which maintains the integrity of relationships between resources. e.g. the tool might present views of the appropriate resources available within the registry, from which the user can select the relevant items.

As noted above, the registry is not a closed system, and the schema creation and registration tool should permit a user to store their descriptions as RDF/XML documents so that they might be used for purposes other than submission to the registry.

Use Case 1: Publishing a description of a Element Set

An EC project provides a resource discovery service for Web-based educational materials. That service utilises a simple metadata schema developed specifically for that purpose. The organisation wishes to publish this information to the wider community via the CORES registry.

To publish requires the following steps. The Element Set publisher:

1. Uses the Schema Creation and Registration Tool (SCART) to add Agency description (if not already present)
2. Submits new Agency description to registry
3. Uses SCART to add Namespace/Element Set description

4. Uses SCART to add Element/Term descriptions for Element/Terms in Namespace/Element Set, including association of Element/Term with Encoding Scheme(s) where appropriate
5. Submits new Namespace/Element Set and Element/Term descriptions to registry
6. May check results by browsing Namespace/Element Set descriptions via registry Web interface

Use Case 2: Publishing a description of an Application Profile

An EC project provides a resource discovery service for Web-based educational materials. That service utilises a simple metadata schema developed specifically for that purpose. The schema uses a number of Elements drawn from the cross-domain Element Sets of the Dublin Core Metadata Initiative; a domain-specific Element that was created by another portal service for their own schema; and a number of new Elements specific to this service. The organisation has developed a number of controlled vocabularies for several of the Elements in this schema; the service also specifies the use of some standardised forms for dates and identifiers within metadata instances. The organisation wishes to publish this information to the MEG community via the registry.

In the terms of the registry data model, this organisation's schema is an Application Profile. To publish requires the following steps. The Application Profile publisher:

7. Uses the the CORES Schema Creation Tool to add Agency description (if not already present)
8. Submits new Agency description to registry
9. Uses SCART to add Namespace/Element Set description
10. Uses SCART to add Element/Term descriptions for Element/Terms in Namespace/Element Set, including association of Element/Term with Encoding Scheme(s) where appropriate
11. Submits new Namespace/Element Set and Element/Term descriptions to registry
12. May check results by browsing Namespace/Element Set descriptions via registry Web interface.

Use Case 3: Indexing a standard schema for a Element Set

An international standards body makes schema for their cross-domain Element Set available in RDF/XML on their Web server. European implementors wish to "use" Elements from the Element Set in their Application Profiles.

Either the representative of standards body or the registry administrator:

13. Uses the CORES Schema Creation Tool to add Agency description (if not already present).

14. Submits new Agency description to registry
15. Uses the CORES Schema Creation Tool to add Element Set description (assumes external schema on Web does not contain required data)
16. Requests registry to read Element descriptions from URL
17. May uses the CORES Schema Creation Tool to enhance Element descriptions for registry-specific data (or may leave incomplete)
18. Submits updated Element descriptions to registry
19. May check results by browsing Element Set descriptions via registry Web interface .

Use Case 4: Exploring Element Usage

A schema developer wishes to survey the usage of the DCMI "audience" element, and particularly the use of any controlled vocabularies to control values of this element.

The developer

20. Browses Elements via registry Web interface
21. Displays description of Element "dcterms:audience", which includes pointers to the Encoding Schemes associated with the Element, and pointers to its usage by various Application Profiles
22. Follows references to Element Usage descriptions, which included descriptions of how implementers have constrained the use of the Element, including the prescription of other Encoding Schemes

Use Case 5: Creating annotations

5a. Schema Creator creates a schema using CORES schema creation tool and registers it in CORES Registry. They then want to annotate the schema with information about the number of implementations, domains in which this schema is deployed, and pointers to user guidelines.

The schema creator

23. Browses element sets via Registry web interface
24. Displays details of the chosen element set
25. Adds an annotation

5b. A schema creator searches the registry and displays an element set, then is interested in whether this schema is currently being maintained. He browses each annotation associated with the schema, looking in particular at the names of the annotator and associated organisation.

26. Display details of chosen element sets

27. Display annotations

5c. A commentator wants to annotate an element with usage notes and comments regarding a scheme outlining experience gained in using that scheme.

1.6 The CORES Schema Creation Tool data model

A graphical outline of the classes of resource described and the relationships between instances is provided below. This section outlines the attributes/properties of instances of these classes. It is based on the data model for the MEG registry [2] and also draws on the work of the SCHEMAS project [13]. In particular, it adopts the recommendation of Baker et al that "Element Usages" should be modeled as resources in their own right. [14]

An **Application Profile** defines a set of **Element Usages** of **Elements** drawn from one or more **Element Sets**. Such a **Element Usage** may:

- introduce constraints on the value of an Element by associating the it with one or more **Encoding Schemes**,
- introduce constraints on the **obligation** to use an Element (e.g. make its use mandatory) or the **occurrence** of an Element (e.g. whether it is repeatable),
- **refine** the semantic definition of an Element to make it narrower or more specific to the application domain.

In the tables below, the prefixes used in property names are associated with XML Namespaces as follows:

- **rdf:** <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
- **rdfs:** <http://www.w3.org/2000/01/rdf-schema#>
- **dc:** <http://purl.org/dc/elements/1.1/>
- **dcterms:** <http://purl.org/dc/terms/>
- **dcmitype:** <http://purl.org/dc/dcmitype/>
- **reg:** <http://www.ukoln.ac.uk/metadata/education/regproj/reg/>

Figure 1 CORES data model

1.6.1.1 Agency attributes/properties

Label	Property name	Comments	
Identifier	(rdf:about)		

Agency Name	reg:agencyName	The name or title of the Agency	
Agency Homepage	reg:agencyHomepage	The URL of a document which provides more information about the Agency, typically an organisational "home page".	

1.6.1.2

1.6.1.3 Element Set attributes/properties

Label	Property name	Comments	
Identifier	(rdf:about)		
Name of Element Set	dc:title	The name or title of the Element Set	
Version	reg:version	The version of the Element Set.	
Creation date	dcterms:created	Date this version created	
Status	reg:status	Status of Element Set.	
Description	dc:description	Description of Element Set, including any notes of scope/purpose.	
Classification	reg:classification	Classification of use of this Element Set	
Responsible Agency	reg:responsibleAgency	Publisher of Element Set	Pointer to resource of type Agency
XML Namespace Name	reg:xmlNamespace	XML Namespace name/URI associated with this Element Set	
XML Namespace prefix	reg:xmlNamespacePrefix	??Prefix to be used in instances using this Element Set	
Specification	reg:specification	URL of prose description of/guidelines for use of Element Set	

1.6.1.4

1.6.1.5

1.6.1.6 Element attributes/properties

Label	Property name	Comments	
Identifier	(rdf:about)		
Name of Element	rdfs:label	A human-readable version of the property name	
Definition	rdfs:comment	A statement that clearly represents the concept and essential nature of the Element	
Comment	reg:useComment	A remark concerning the application/use of the data element	
Data type	reg:datatype	Indicates the type of data that can be represented in the value of the data element	
Obligation	reg:obligation	Indicates whether the Element is always or sometimes required to be present	
Maximum occurrence	reg:maximumOccurrence	Indicates any limit to the repeatability of the Element	
Associated Encoding Scheme	reg:associatedEncodingScheme	An Encoding Scheme which specifies constraints on the value of this Element	Repeatable. Pointer to resource of type Encoding Scheme
Refines	rdfs:subPropertyOf	An Element of which this Element is a "refinement" [DC]	Pointer to resource of type rdf:Property
Element Set	reg:isElementOf	An Element Set of which this Element is part	Pointer to resource of type Element Set

1.6.1.7

1.6.1.8 Encoding Scheme attributes/properties

Label	Property name	Comments	
Identifier	(rdf:about)		
Name of Encoding Scheme	rdfs:label	The name or title of the Encoding Scheme	

Scheme		Scheme	
Version	reg:version	The version of the Encoding Scheme.	
Creation date	determs:created	Date this version created	
Status	reg:status	Status of Encoding Scheme.	
Description	rdfs:comment	Description of Encoding Scheme, including any notes of scope/purpose.	
Classification	reg:classification	Classification of use of this Encoding Scheme	
Responsible Agency	reg:responsibleAgency	Publisher of Encoding Scheme	Pointer to resource of type Agency
Specification	reg:specification	URL of prose description of/guidelines for use of Encoding Scheme	

1.6.1.9

1.6.1.10 Value attributes/properties

An Encoding Scheme is a `rdfs:Class`. Values are instances of resources of that Class.

Label	Property name	Comments	
Identifier			
Value	rdf:value	The value of a term in an Encoding Scheme	
Label	rdfs:label	The human-readable form of the value of a term in an Encoding Scheme	
Description	rdfs:comment	Description	
Encoding Scheme	rdf:type	An Encoding Scheme of which this value is part.	Pointer to resource of type Encoding Scheme.

1.6.1.11

1.6.1.12 Application Profile attributes/properties

Label	Property name	Comments	
Identifier	(rdf:about)		
Name of Application Profile	dc:title	The name or title of the Application Profile	
Version	reg:version	The version of the Application Profile.	
Creation date	dcterms:created	Date this version created	
Status	reg:status	Status of Application Profile.	
Description	dc:description	Description of Application Profile, including any notes of scope/purpose.	
Classification	reg:classification	Classification of use of this Application Profile	
Responsible Agency	reg:responsibleAgency	Publisher of Application Profile	Pointer to resource of type Agency
Associated XML Schema	reg:associatedXMLSchema	XML Schema associated with this Application Profile	

Specification	reg:specification	URL of prose description of/guidelines for use of Application Profile	
---------------	--------------------------	---	--

4.7 Element Usage attributes/properties

Label	Property name	Comments	
Identifier	(rdf:about)		
Uses	reg:uses	An Element which is used in the context of this Application Profile.	Pointer to resource of type rdf:Property
Name of Element Usage	rdfs:label	A human-readable version of the property name, when used in the context of this Application Profile.	
Definition	rdfs:comment	A statement that clearly represents the concept and essential nature of the Element, as used in this Application Profile.	
Comment	reg:useComment	A remark concerning the application/use of the Element, when used in the context of this Application Profile.	
Data type	reg:datatype	Indicates the type of data that can be represented in the value of the Element, when used in the context of this Application Profile.	
Obligation	reg:obligation	Indicates whether the Element is always or sometimes required to be present, when used in the context of this Application Profile.	
Maximum occurrence	reg:maximumOccurrence	Indicates any limit to the repeatability of the Element, when used in the context of this Application Profile.	
Associated Encoding Scheme	reg:associatedEncodingScheme	An Encoding Scheme which specifies constraints on the value of this Element	Repeatable. Pointer to resource of type Encoding Scheme
Application Profile	reg:isUsageIn	An Application Profile of which this Element Usage is part	Pointer to resource of type Application Profile

1.6.1.13

1.6.1.14 Annotation attributes/properties

Label	Property name	Comments
Identifier	(rdf:about)	
Annotates	reg:annotates	The name or title of the resource that is topic of commentary
Body	reg:body	The body of this annotation
Type	reg:type	The type of this annotation
Access	reg:access	Who can access this annotation
Creator	reg:creator	The creator of this annotation
Date	reg:date	The date of creation of this annotation

1.6.1.15

1.6.1.16 Commentator attributes/properties

Label	Property name	Comments
Identifier	(rdf:about)	
User	reg:user	Name of person creating commentary
User's Organisation	reg:userOrg	User's organisation
Password	reg:password	This user's password
User's Name	reg:userName	This user's name
User's E-mail	reg:userEmail	This user's e-mail address

1.6.2

1.6.3 Creating and editing schemas using the CORES schema creation tool

The tool should allow a user to create, maintain and remove resource descriptions which they own within the CORES registry.

1.6.3.1 Add Agency description

- Create description of Agency

- Save description of Agency
- Submit description of Agency to registry

1.6.3.2 Edit/Update Agency description

- Open existing description of Agency
- Amend description of Agency
- Save description of Agency
- Re-submit description of Agency to registry

1.6.3.3 Remove Agency description

- Open existing description of Agency
- Remove existing description of Agency (Do not permit removal if used in relationships?)

1.6.3.4 Add Element Set description

- Create description of Element Set
- For each Element in Element Set, create description of Element
- Save description of Element Set
- Submit description of Element Set to registry

1.6.3.5 Edit/Update Element Set description

- Open existing description of Element Set
- Amend description of Element Set
- For each Element to be amended, open existing description of Element, amend description of Element
- For each Element to be added, create description of Element
- For each Element to be removed, remove description of Element (Do not permit removal if Element used in relationships?)
- Save description of Element Set
- Re-submit description of Element Set to registry

1.6.3.6 Remove Element Set description

- Open existing description of Element Set
- Remove existing description of Elements in Element Set, remove existing description of Element Set (Do not permit removal if any Element used in relationships, or if Element Set used in relationships?)

1.6.3.7 Add Encoding Scheme description

- Create description of Encoding Scheme
- For each Value in Encoding Scheme, create description of Value
- Save description of Encoding Scheme
- Submit description of Encoding Scheme to registry

1.6.3.8 Edit/Update Encoding Scheme description

- Open existing description of Encoding Scheme
- Amend description of Encoding Scheme
- For each Value to be amended, open existing description of Value, amend description of Value
- For each Value to be added, create description of Value
- For each Value to be removed, remove description of Value
- Save description of Encoding Scheme
- Re-submit description of Encoding Scheme to registry

1.6.3.9 Remove Encoding Scheme description

- Open existing description of Encoding Scheme
- Remove existing description of Values in Encoding Scheme (OK because no references), remove existing description of Encoding Scheme (Do not permit removal if Encoding Scheme used in relationships)

1.6.3.10 Add Application Profile description

- Create description of Application Profile
- For each Element Usage in Application Profile, create description of Element Usage. Note: a Element Usage does not automatically inherit the Encoding Schemes associated with the used Element. Any relevant Encoding Schemes must be explicitly associated with the Element Usage.
- Save description of Application Profile
- Submit description of Application Profile to registry

1.6.3.11 Edit/Update Application Profile description

- Open existing description of Application Profile
- Amend description of Application Profile
- For each Element Usage to be amended, open existing description of Element Usage, amend description of Element Usage

- For each Element Usage to be added, create description of Element Usage
- For each Element Usage to be removed, remove description of Element Usage (OK because no references)
- Save description of Application Profile
- Re-submit description of application to registry

1.6.3.12 Remove Application Profile description

- Open existing description of Application Profile
- Remove existing description of Element Usages in Application Profile, remove existing description of Application Profile (OK because no references)

1.6.4 6. Creating and editing annotations

1.6.4.1 Add commentator description

- Prompt for details on first attempt to create annotation

1.6.4.2 Edit commentator description

- Amend description

1.6.4.3 Add annotation description

- Click on annotations button
- Enter details of annotation
- Specify whether public or private
- Specify type of annotation
- Specify whether it is public or private annotation

1.6.4.4 Edit annotation description

- Open existing annotation
- Amend text

1.6.4.5 Remove annotation description

Delete annotation

1.7 Notes and references

[1] *MEG Registry project.*

HTML: <http://www.ukoln.ac.uk/metadata/education/regproj/>

[2] **Johnston, Pete.** *Functional Requirements for MEG Registry*

HTML: <http://www.ukoln.ac.uk/metadata/education/regproj/funcreq/>

[3] *Schemas project.*

HTML: <http://www.schemas-forum.org>

[4] *DCMI Registry.*

HTML: <http://dublincore.org/dcregistry/>

[5] *W3C Semantic Web Activity Group.*

HTML: <http://www.w3.org/2001/sw/>

[6] *RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft 30 April 2002 (Work in Progress).*

HTML: <http://www.w3.org/TR/rdf-schema>

[7] *DESIRE Registry*

HTML: <http://www.desire.ukoln.ac.uk>

[8] **Heery, Rachel, Tracy Gardner, Michael Day & Manjula Patel,** *DESIRE metadata registry framework*

HTML: <http://www.desire.org/html/research/deliverables/D3.5/>

[9] *The SCHEMAS Forum Registry*

HTML: <http://www.schemas-forum.org/registry/>

[10] For a contextual definition of 'Registry' see *The SCHEMAS Forum : A Retrospective Glossary*

HTML: <http://www.schemas-forum.org/info-services/d74.htm>

[11] **Heery, Rachel & Manjula Patel,** *Application Profiles: mixing and matching metadata schemas Ariadne 25. Sept 2000.*

HTML: <http://www.ariadne.ac.uk/issue25/app-profiles/>

[12] *This is a subset of the information managed within the current prototype SCHEMAS registry, which uses the model developed by the DESIRE project. The model suggested here excludes support for the capacity to group Element Sets by "Namespace Concept" and for the capacity to describe "Value Components" for an Encoding Scheme. Activity reports are modelled as Annotations.*

[13] *The SCHEMAS Forum Registry : sample RDF encodings of Application Profiles*

HTML: <http://www.schemas-forum.org/registry/schemas/>

[14] **Baker, Thomas, Makx Dekkers, Rachel Heery, Manjula Patel and Gauri Salokhe,** What terms does your metadata use? Application profiles as machine-understandable narratives, *Journal of Digital Information* Volume 2 Issue 2 (Nov 2001)

HTML: <http://jodi.ecs.soton.ac.uk/Articles/v02/i02/Baker/>

2 Installation Guide

2.1 Install guide for the client

1. The client is written in Java
 - if you have not got Java installed the go to <http://java.sun.com>
 - download and install the latest version of J2SE JRE for your platform and OS
 - for install and configuration instructions please look at the documents of Java
2. Download the latest client from the home page of the registry (CORESClient.jar)
 - go to <http://cores.dsd.sztaki.hu>
 - the link to the client is in the header ('the latest client')
 - the way of download is depend on your browser (usually click on the link)
3. Start the client
 - enter 'java -jar CORESClient.jar' to the console in the directory where you have downloaded the client
 - OR just double-click on the downloaded file to start the client (this is depend on your OS)
 - for any further information about use of the client see the user's manual

2.2 Install guide for the server

(based on [Dave Beckett's](#) Meg Registry Install guide)

Getting the source

- The code will be available in CVS. Instructions follow later.

Building Registry (Server)

```
cd cores/src/server
```

which will be called the <SERVER-DIR> below. This directory contains the registry, based on Redland RDF system.

You will need to create some directories at the top level:

```
mkdir data
mkdir data/Agency data/AppProfile data/Element
mkdir data/ElementSet data/ElementUsage
mkdir data/EncodingScheme data/Schema
mkdir db logs redland
```

and make data, db writable by the apache web user/group

(you will probably have to do the following as root)

```
chown -R apacheuser.apachegroup data db
chmod -R 755 data db
```

or the more dangerous:

```
chmod -R a+w data db
```

You will need to get Redland from <http://www.redland.opensource.ac.uk/> and install it. The registry uses the stable 0.9.11 version and it is strongly recommended to use that release.

Once you have downloaded the 0.9.11 tar.gz, extract and built it:

```
./configure --prefix=<SERVER_DIR>/redland
make
```

the following fixes a bug in the Redland 0.9.11 configuration:

```
ln -s .libs/librdf.a
```

and the perl interface:

```
cd perl
make test-perl
cd ..
```

and install the library and the perl interface:

```
make install
```

```
cd perl
make install-perl
```

2.3 Loading the initial registry

The loading is made about data/contents.tab file. It contains the location of the initial schemas.

use bin/get-schemas.pl to fetch the above. Then run bin/registry.pl to force an index of the files.

2.4 Install and configuration of the web interface

Each of the cgi-bin programs or command line versions requires some environment specific variables. They can be set in the config/cores.conf file. They are:

- ROOT_DIR: above has called <SERVER-DIR>
- WEB_ROOT: the URL where you have put the registry
- DOMAIN: the domain part of the above URL
- CORES_ADMIN_EMAIL: the administrator's e-mail address
- REDLAND: where you have installed Redland. Default is <SERVER-DIR>/redland

You have to set the following Apache configuration:

```
SetEnv LD_LIBRARY_PATH <SERVER
DIR>/redland/lib:/usr/local/lib
```

to get the redland shared library visible to apache.

The cgi-bin programs have to be mapped to handle a set of URI space. Use the following aliases:

```
ScriptAlias /cgi-bin <SERVER-DIR>/cgi-bin
```

```
ScriptAlias /api <SERVER-DIR>/cgi-bin/api.pl
```

```
ScriptAliasMatch ^/$ <SERVER-DIR>/cgi-bin/browse.pl
```

(order is important)

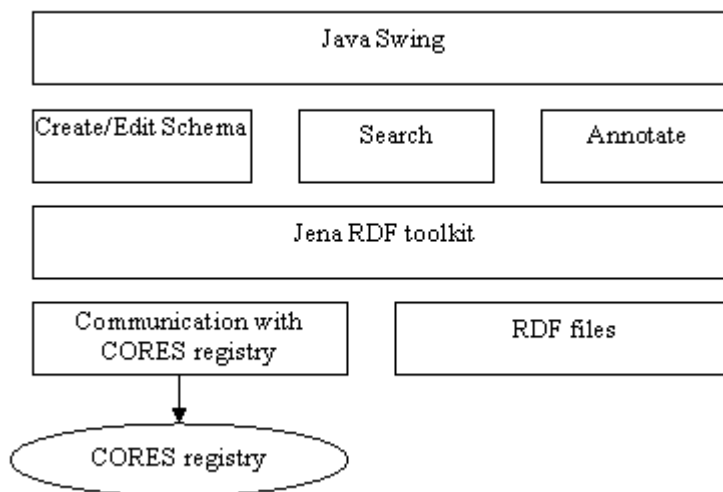
3 Developer's manual

The software consist of a client and a server. The client is a Swing application written in Java, the server is a set of CGI scripts written in Perl.

3.1 The Client

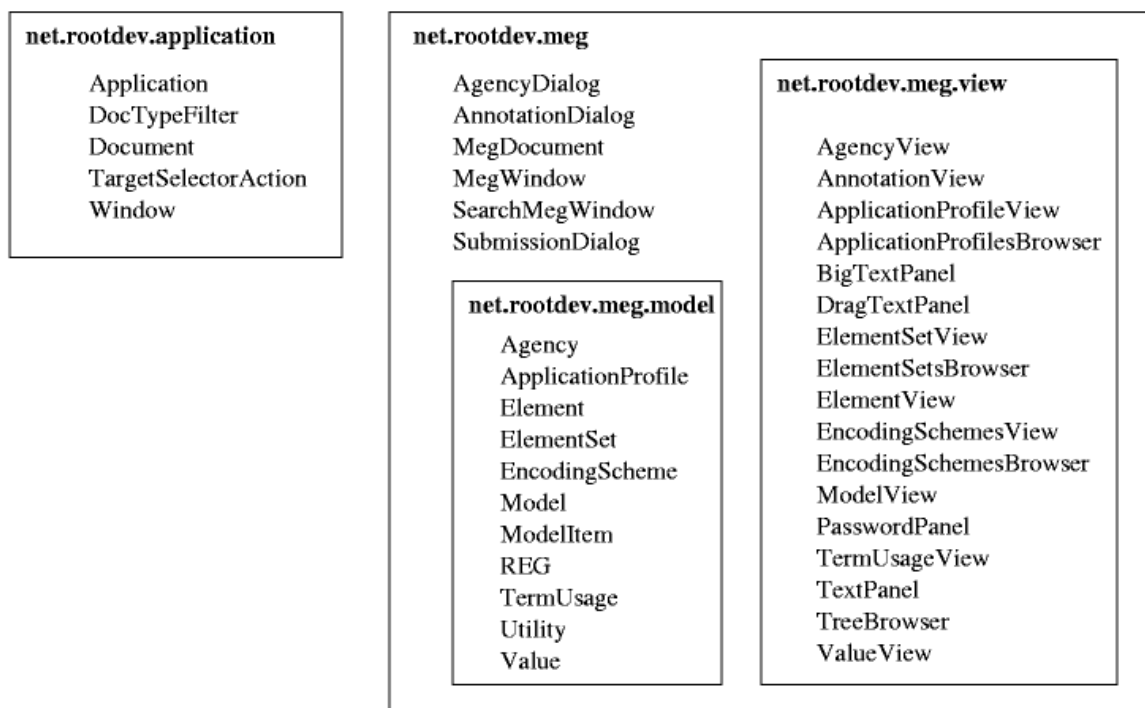
3.1.1 Architecture

The client is built on the [Jena RDF toolkit](#) [1]. The architecture of the client is shown on the figure. Java Swing provides the graphical presentation toolkit of the client. 'Drag and drop' metaphor is intensively used, mostly for the reuse of existing schema elements. All RDF (collected from the registry or created locally) are stored in memory according to the memory storage model of Jena. On request portions of this RDF can be saved in local files. The client communicates with the registry using the HTTP protocol, exchanging data in RDF format. As most of the transferred data is in RDF originally, this solution eliminates the need of data conversion during network transfer.



3.1.2 Packages

The client consist of the following packages and classes:



- net.rootdev.application package: it defines a document-window architecture to separate the document and application logic.
- net.rootdev.meg package (later a name change will be necessary): the CORES client built on the previous application model.
- net.rootdev.meg.model package: classes to build up a model. A document is meant here as a model of an RDF schema, and it is modelled in Java using these classes. The ModelItem class is the ancestor of all these classes.
- net.rootdev.meg.view package: classes to build up a view, a window consist of this classes. The ModelView class is the ancestor of all these classes.

3.1.3 Classes

The major classes are:

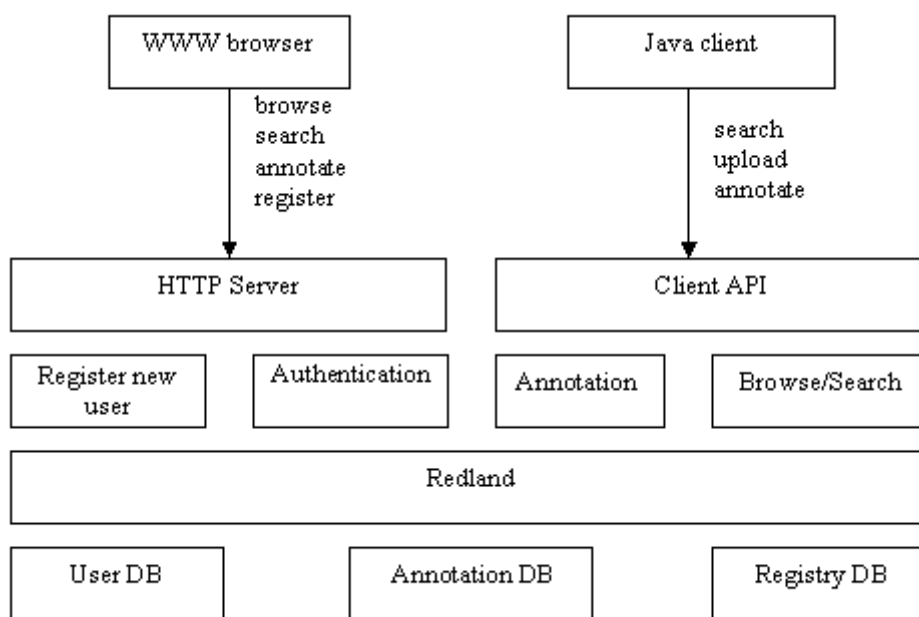
- MegDocument: the model of a schema.
- MegWindow: the view of a schema.
- AgencyDialog: input window for the details of an agency.

- SearchMegWindow: search interface for the registry.
- SubmissionDialog: interface for upload to the registry.
- AnnotationDialog: annotate facility for the client.
- REG: specific constants (namespaces, Jena variables).
- Model: a class to store models
- ModelItem: elements of a model. It's descendants are Agency, ApplicationProfile, Element, ElementSet, EncodingScheme, TermUsage and Value.
- ModelView: a common interface for the elements of a view of a model. It's implementing classes are AgencyView, ApplicationProfileView, ElementView, ElementSetView, EncodingSchemeView, TermUsageView and ValueView.
- TreeBrowser: A class to browse schemas. It's descendants are ApplicationProfilesBrowser, ElementSetsBrowser and EncodingSchemesBroser.

3.2 The Server

3.2.1 Architecture

The server is built on the [Redland](#) [2] RDF toolkit. The architecture of the server is shown on the figure. The server provides two interfaces: one for the user to browse its contents with a web browser, and a machine API for the client to upload schema definitions. Annotations can be created using both interfaces, but annotation display is available only through the web interface. The server is implemented as a set of Perl scripts, using the Redland Perl API to manipulate RDF databases. There are 3 separate databases for user registration data, annotations and the schema registry itself. The storage model used for these databases is Berkeley DB. Some other information are stored in files (e.g. session data, data for confirmation of registration), and the original RDF schemas are also available as plain files.



3.2.2 Modules

The server consist of the following modules and scripts:

Registry.pm	api.pl browse.pl
Annotation.pm	list.pl add.pl
Session.pm	new_user.pl login.pl/logout.pl
Users.pm	confirm.pl

The important functions are grouped in modules:

- Registry.pm: functions responsible for display and handle the registry.
- Annotation.pm: functions responsible for annotations.

- Session.pm: session handling.
- Users.pm: user management.

3.2.3 Functions and scripts

The major functions are:

in Annotation.pm:

- `get_annotations_for`: list the annotation database for annotations to the given resource.
- `get_annotation`: get the details of a given annotation from the annotation database.
- `count_annotations`: count the annotations for a given resource.
- `delete_annotation`: delete a given annotation from the annotation database.
- `upload_annotation`: upload a given annotation to the annotation database

in Registry.pm:

- `l`: write debug information to the web servers `error_log` file.
- `start_html_page`, `start_annotation_page`: headers for the web registry.
- `end_html_page`, `end_annotation_page`: footers for the web registry.
- `class_label`, `class_label_from_uri`: determine the class of a resource.
- `display_fields_for_class`: the properties belong to the given resource class.
- `summary_fields_for_class`: the properties to list when browsing.
- `concept_is_internal`: it indicates if a property is internal and do not have to display.
- `uses_details_for_class`, `related_properties_for_class`: links to associated resources for the given class.
- `nodes_sorted_by_name`: sort the given resources by they name.
- `update_contents_index`: update the index file (`lindex.tab`). `lindex.tab` is used for searching.
- `build_indexes`: rebuild the registry database from the stored files.
- `name_for_resource`: find the correct name property for a given resource and list it from the registry database.

in Session.pm:

- getSession: get session data from the stored file or create it.
- saveSession: save session data to file.

in Users.pm:

- get_users_data: get the details of a given user for the annotation listing.
- check_login: validate an attempt for login.

The scripts are:

- add.pl: add a new annotation.
- api.pl: api for the client. It can search, upload and annotate through this script.
- browse.pl: browse the content of the registry.
- confirm.pl: finish the new user registration process with confirmation.
- list.pl: list the annotations for a given resource.
- login.pl: handle and check the logins.
- logout.pl: logout the current user.
- new_user.pl: start the registration process for a new user and send her a confirmation code.

3.2.4 References

[1] *Jena RDF toolkit:*

HTML: <http://www.hpl.hp.com/semweb/index.html>

[2] *Redland RDF toolkit:*

HTML: <http://www.redland.opensource.ac.uk/>

3.3 Work list: bug fixes, changes and developments

3.3.1 Server

- graphical layout changes (logo replaced, new header/footer etc.)
- provide download of the latest client
- the command='publish', class='element' parameter combination was not implemented
- evaluation of Java and Perl based solutions: functionality and performance tests
- installation and evaluation of W3C Annotea
- write simple perl client for annotea
- store annotations in Redland instead of Annotea (it was very slow)
- creation of annotation display window
- integration of annotation window with the registry browser
- support for deleting annotations
- editing annotations
- planning of user registration
- implementation of user registration in CORES registry
- planning and development of authentication
- login/logout page
- separating rdf databases for the registry, annotations and users
- change from the MEG namespace to CORES
- update the CORES RDFS (with annotations and users)
- making server installation easier
- switch to the CORES RDFS in the code
- display names instead of ids in registry browser

- numerous bug fixes: avoid drawing empty tables, correcting HTML output, etc.

3.3.2 Client

- implementation of annotation functionality in the client, new class AnnotationDialog to send annotations to the registry
- implementation of authentication for sending annotations
- modify the GridBagConstraints
- the MegDocument calls MegWindow.validate() for the necessary resizing after changing of the viewPane, to this the MegDocument needs a reference to MegWindow: MegDocument.mv
- modify the source of Document after 'save' or 'save as', so it remembers the (new) file
- there were some bug about the Element identifier
- change the MEG picture to CORES
- there were a bug in the upload: the client had used different namespace to identify the type of the Elements
- the program did not exit after closing of the last window, for this it needs a new method: MegWindow.windowClosing()
- closed windows were not removed from the window menu and they documents from the application, for this the client needs new method: Application.removeDocument()
- make checking for file opening, if the client can't open any files from the parameters then it opens an empty document
- the close menu did not do anything, to fix it the client needs a new method MegWindow.close() and MegWindow.closeWindow();
- correct removing and drawing of EncodingSchemes (there were a bug with JTree)
- add the missing schema elements (ElementSet-xmlNamespace, Element-datatype, Value-label, AppProfile-classification, ElementUsage-datatype)
- when removing a Value then remove all instance (if the EncodingScheme is associated with elements and element usages), this needs a new method: Value.removeItem:
- correct the drag&drop over <drag here to set> label in TermUsage
- correct the SearchMegWindow: it had crashed after several searches (it had called bad function of the listeners of JTree with bad parameters)

- change in MegDocument.valueChanged() : the focus of the browser trees can change only by clicking on the title. (changing the focus by clicking in the tree is disturbing when drag&drop)
- change in MegDocument.remove() : maybe there isn't a selected element
- change method: ModelItem.removeItem: locate the removable elements by browser and PATH (if an EncodingScheme was associated with two element then both were removed)
- when removing from EncodingSchemesBrowser then remove all instance: new method was needed: EncodingScheme.removeItem()
- change in Document.saveAs: now it can overwrite existing files

4 User's Manual

4.1 The Server (Registry)

The home page of the CORES registry is <http://cores.dsd.sztaki.hu>. The user can do the following things with the registry:

4.1.1 Browse Schemas

There are links on the main page and in the footer to the content of the registry.



The screenshot shows the homepage of the CORES Registry. At the top left, the text "CORES Registry" is displayed in blue. To the right is the CORES logo, which consists of a circular arrangement of stars and the text "CORES" in the center. Below the logo, the text "the latest client" is on the left and "not logged in" is on the right. A dark blue horizontal bar with the word "Index" in white is centered below the navigation area. Underneath this bar, there are several lines of text, each starting with a category name followed by "Browse - Search": "Agencies: Browse - Search", "Element Sets: Browse - Search", "Elements: Browse - Search", "Encoding Schemes: Browse - Search", "Application Profiles: Browse - Search", and "Element Usages: Browse - Search". At the bottom of the page, there is a navigation menu with links: "Index - Agencies - Element Sets - Elements - Encoding Schemes - Application Profiles - Element Usages - login". Below the navigation menu, there is a copyright notice: "© 2002 MEG Registry Project, ILRT and UKOLN If you have any problems please contact the administrator 2002, CORES Project, MTA SZTAKI DSD".

The content can be listed in two views:

- list a group of resources (Application Profiles, Element Usages, etc.)
- detailed view of a resource

The user can list a selected group of resources from the main page or from the footer. There are links in the list to the details.

Elements

Name	Element Set	
Abstract	The Dublin Core Terms namespace	Detail
Alternative	The Dublin Core Terms namespace	Detail
Audience	The Dublin Core Terms namespace	Detail
Available	The Dublin Core Terms namespace	Detail
Conforms To	The Dublin Core Terms namespace	Detail

From the detailed view she can reach the associated resources (Encoding Schemes, Responsible Agency, etc.).

Element: Abstract

ID	http://purl.org/dc/terms/abstract
Name	Abstract
Definition	A summary of the content of the resource.
Comment	
Data type	
Obligation	
Maximum Occurrence	
Associated Encoding Scheme	
Refines	Description
Element Set	The Dublin Core Terms namespace

Annotations

There is no annotations for this resource. [Add new annotation.](#)

Element Usages

Name	Application Profile	
Abstract	The Qualified Dublin Core Application Profile	Detail

Refines

Name	Element Set	
Description	The Dublin Core Element Set v1.1	Detail

4.1.2 Search Schemas

The search page can be reached from the main page. The user can select the kind of resource she would like to search and gives the search string. The search is not executed on the names only, but on all the descriptions of resources.

Search for Elements matching 'abstract'

Search for an matching

Results

Name	Element Set	
Abstract	The Dublin Core Terms namespace	Detail

4.1.3 Browse Annotations

List the annotations for a resource by the link at the detailed page.

Annotations

There are 2 annotations for this resource. [List annotations.](#)

The annotation list opens in a new window. The annotations is ordered by they creation date. The user can see the creator, the creation date, the annotation type and the access fields for annotations.

Annotations for Test Element 1

Csaba Fulop(MTA SZTAKI)	Thu Oct 24 21:32:24 2002	Comment	Public
Test annotation 2. delete edit			
Csaba Fulop(MTA SZTAKI)	Thu Oct 24 21:30:49 2002	Example	Public
This is a test annotation from the client to demonstrate CORES. delete edit			

Add new annotation

[close - logout](#)

4.1.4 Add/Edit/Delete Annotations

New annotations can be added by clicking the 'add new annotation' link on the annotation list. Then the login page opens. To add annotations the user have to register. If she has not registered yet then she can register by clicking the 'create new account' link.

Login

e-mail:

password:

Create new account

For annotations she can send the type, access and body of the annotation. The type can be comment, advice, example, explanation or question. The access can be 'public' or 'private'. Her private annotations can't be seen by others. To body of the annotation she can write HTML code.

Add new annotation for Test Element 1

Annotates: http://sztaki.hu#elementUsage1
Creator's name: Csaba Fulop
Creator's email: csabi@dspd.sztaki.hu
Creator's organisation: MTA SZTAKI
Type:
Access:
Annotation:

After she has logged in (the login link is in the footer or via annotation adding) she can edit or delete by the links in the bottom of her annotations.

Csaba Fulop (MTA SZTAKI)	Thu Oct 24 21:30:49 2002	Example	Public
This is a test annotation from the client to demonstrate CORES.			
delete edit			

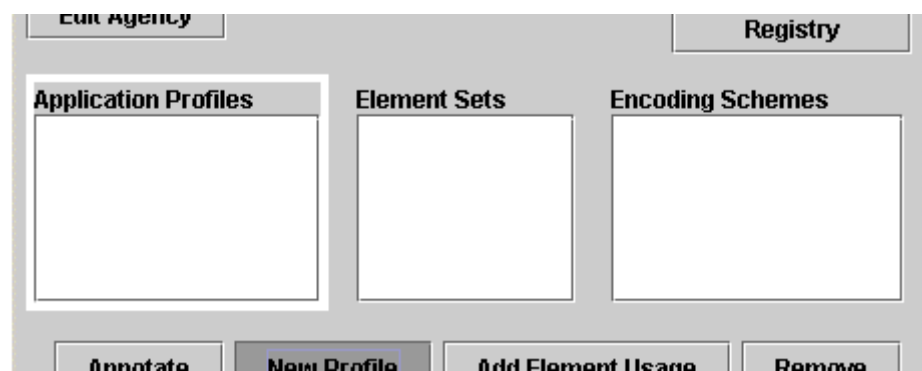
4.2 The Client

4.2.1 Create Schemas

Application Profiles, Element Sets or Encoding Schemes can be created. The Element Sets and Encoding Schemes browser can be seen only in Advanced Mode. The mode can be select from the Mode menu.

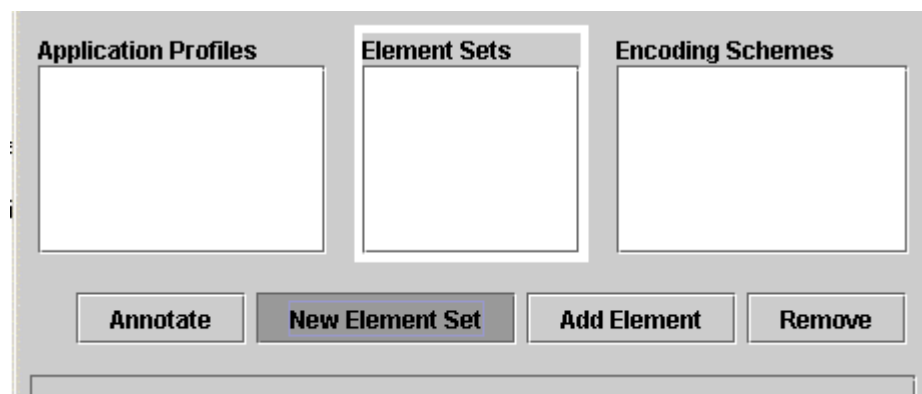


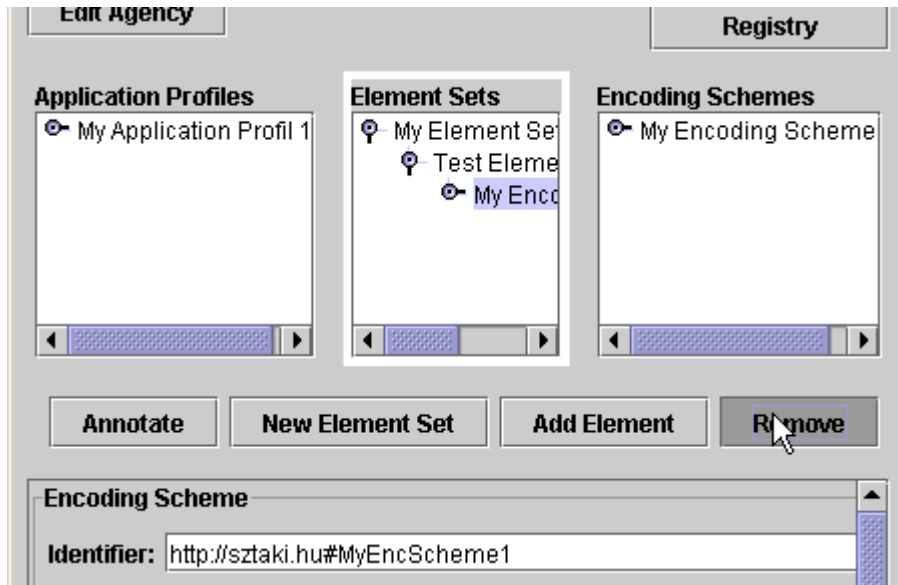
The active browser (Application Profiles, Element Sets or Encoding Schemes) can be selected by clicking on its **label**.



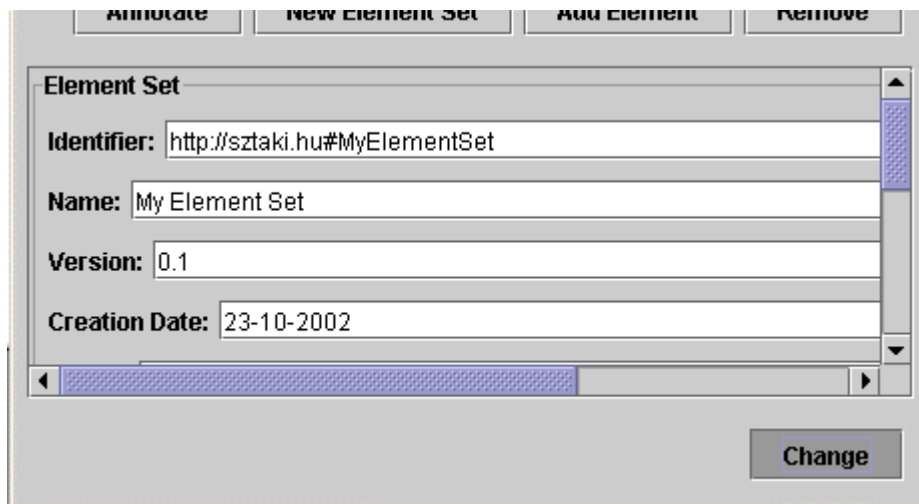
(you can not change the active browser by clicking **in** the browser because it is disturbing during Drag&Drop)

After selecting the proper browser the user can create Element Sets, Elements, etc. by the 'add...' buttons or remove the selected resource from the **selected browser** by the remove button.

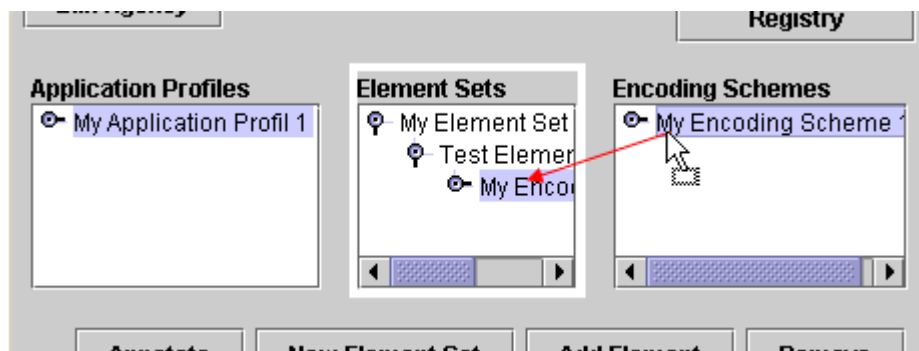


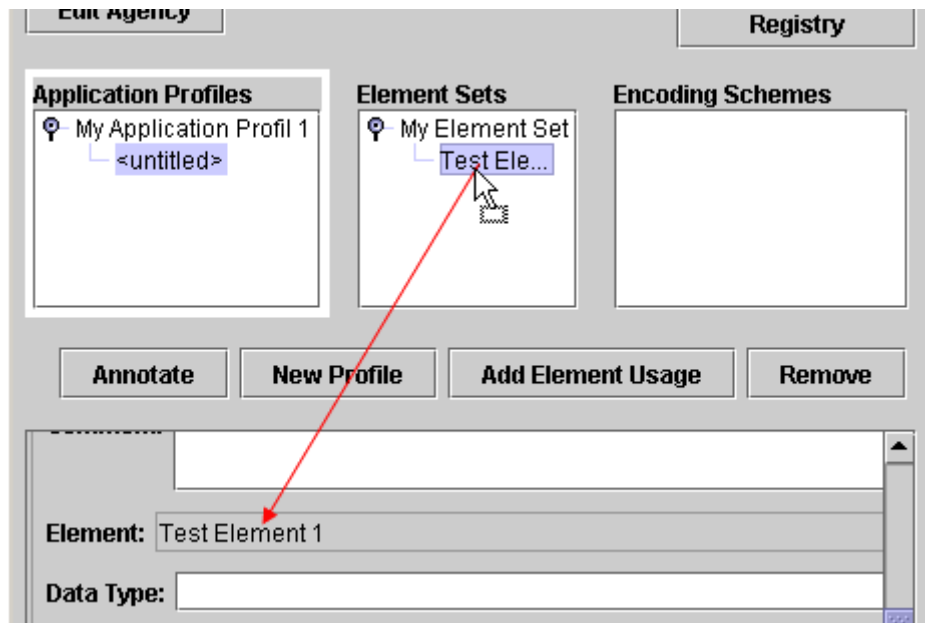


After she has filled out the details she has to click to the 'Change' button.



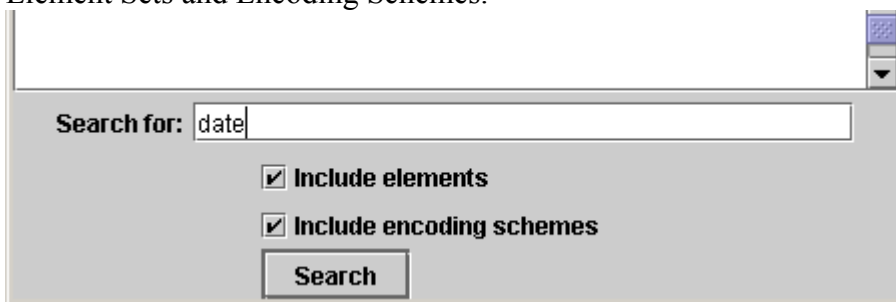
Se can associate Encoding Schemes to Elements or Element Usages and Elements to Element Usages by Drag&Drop.



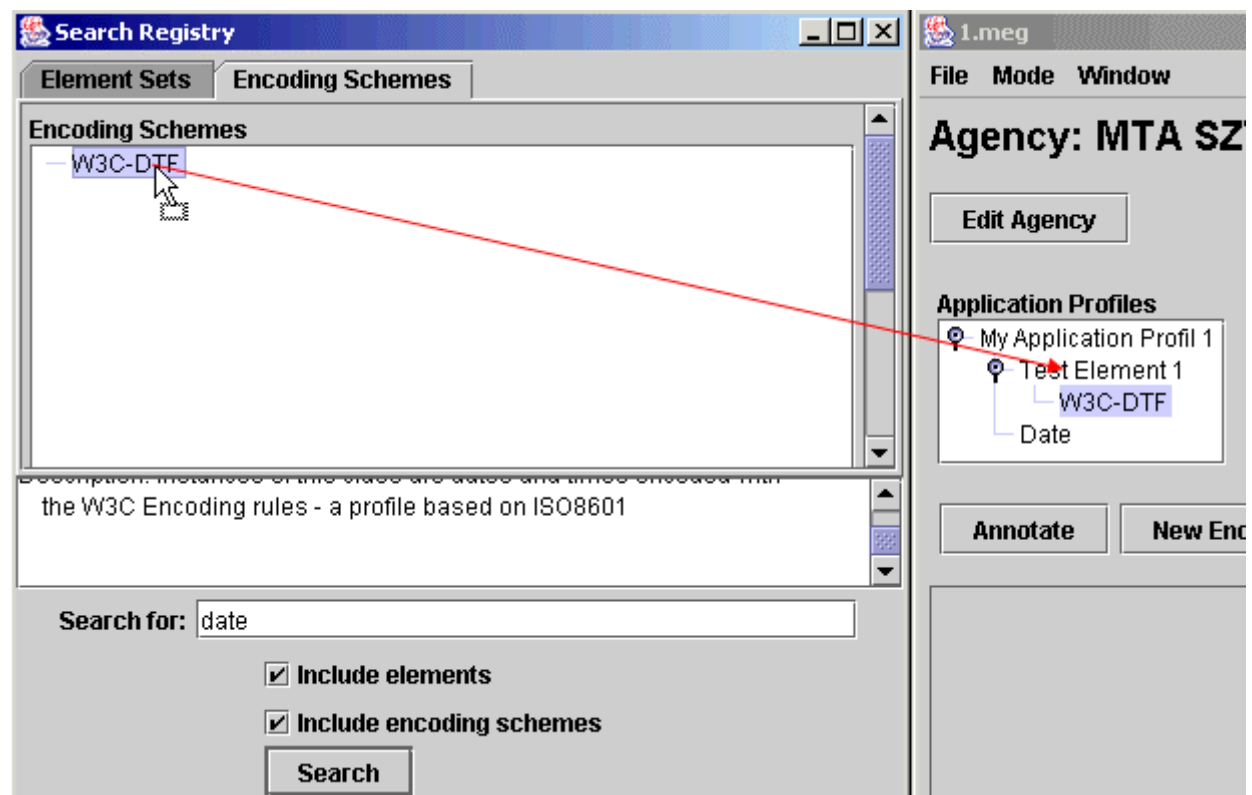
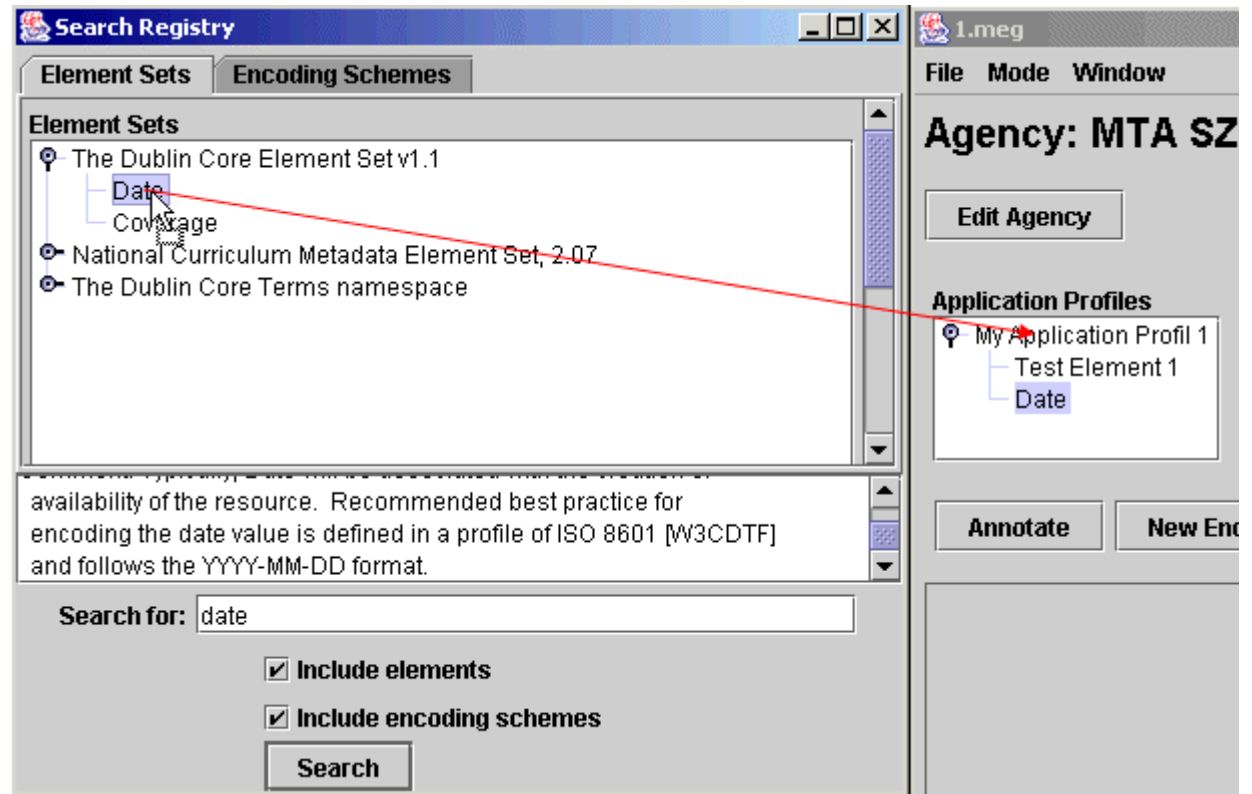


4.3 Search the registry

There is a search registry window. The user can search the CORES registry for Element Sets and Encoding Schemes.

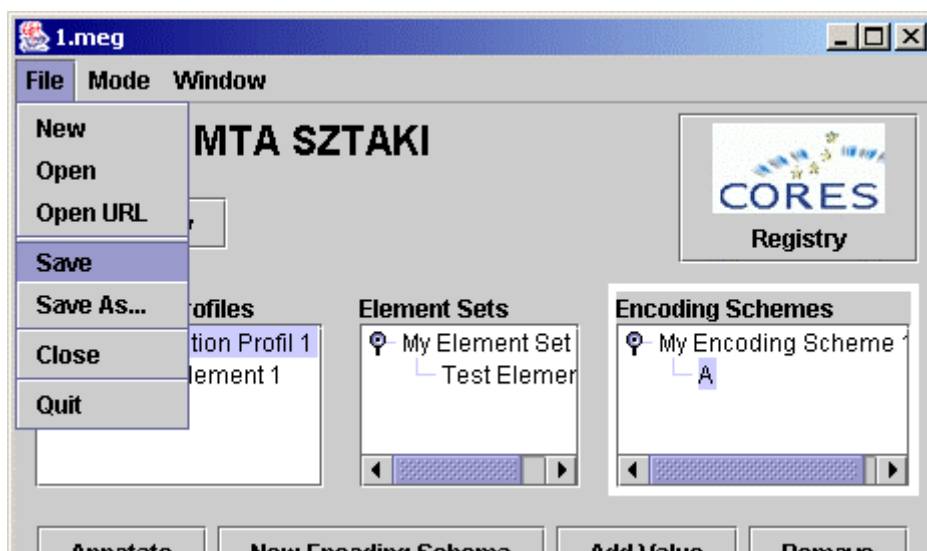


She can associate the hits to her schema with Drag&Drop.

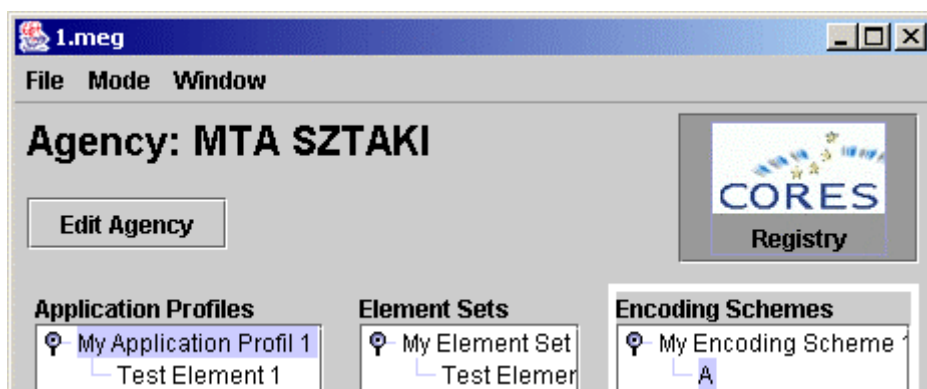


4.4 Store Schemas (save/upload)

After the schema has created the user can save it to her local computer or upload to the CORES registry. She can save from the file menu

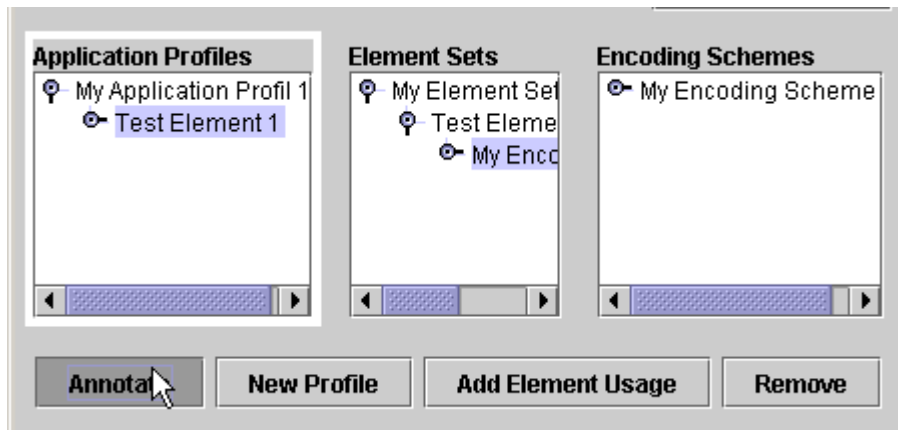


or upload with the 'cores registry' button in the upper right corner.



4.5 Add annotations

The user can add annotations to her schema from the client, too. She have to select the proper browser and resource then click the 'Annotate' button.



She can send the same fields like from the web registry and she has to use the same account.

The screenshot shows a form for sending an annotation. It has the following fields and controls:

- E-mail:** A text input field containing 'csabi@d.sd.sztaki.hu'.
- Password:** A text input field with masked characters '*****'.
- Example:** A dropdown menu with a downward arrow.
- Public:** A dropdown menu with a downward arrow.
- Annotation:** A large text area containing the text 'This is a test annotation from the client to demonstrate CORES.'
- Buttons:** 'Cancel' and 'Send' buttons at the bottom.

