# AGENTCITIES TECHNICAL NOTE

# An Ontology Server for Agentcities.NET

**Agentcities Task Force Technical Note**
actf-note-00008, *11 February, 2013*

Authors:
Monica Duke, UKOLN, University of Bath
Manjula Patel, UKOLN, University of Bath

Comments and requests should be addressed to tech-editor@agentcities.org.

This document and the information contained herein is provided on an "AS IS" basis and THE AGENTCITIES TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Status

*Final*

This version: http://www.agentcities.org/note/00008/actf-note-00008a.html
Latest version: http://www.agentcities.org/note/00008/

## Abstract

Within this six month deployment project[1] we have concentrated on taking forward the ideas and systems developed in a number of initiatives in which UKOLN has been involved, chiefly among these the EU-funded DESIRE[6] and SCHEMAS projects[7], the UK MEG Registry project[15] and the Dublin Core Metadata Initiative[5]. All of these projects explored approaches to declaring and sharing metadata vocabularies using RDF Schemas[18]. We have adapted software for a metadata vocabulary registry to serve as an ontology server which can be queried by agents on the Agentcities.NET network. The contents of the server comprises metadata vocabularies which may be regarded as simple forms of ontology.

# Contents

# 1 Introduction

This is a report on the work carried out between 1st September 2002 and 28th February 2003 at UKOLN, as part of the European Commission funded 5th Framework IST project Agentcities.NET [4]. UKOLN was awarded a grant under the Deployment support program, a "series of grants to support independent new innovative exploratory work related to the Agentcities.NET network. The intention is to enable members to connect their existing or new agent systems to the Agentcities network and carry out exploratory mini-projects - leading to innovative ideas, technology development and new larger scale collaborative projects."

UKOLN [3] is a centre of expertise in digital information management, providing advice and services to the library, information, education and cultural heritage communities. UKOLN is involved in many standardization activities, including the Dublin Core Metadata Initiative (DCMI)[5]; the Research and Development team at UKOLN has taken part in several EU projects including DESIRE[6] and SCHEMAS[7].

The aim of this project is to investigate the support of automated querying of metadata vocabularies by agents, to acquire the semantics associated with specific metadata terms. The approach taken is that of using a registry within which metadata vocabularies are expressed and through which they are communicated. In a registry environment, individual terms as well as whole vocabularies can be investigated by agents. The registry supports the discovery, sharing and re-use of vocabularies, facilitating the convergence of vocabularies (or ontologies), in particular for specific domains. The hope is that alignment in this way will improve the prospects of interoperability of systems in specific sectors.

# 2 Ontologies and Metadata Vocabularies

Ontologies provide a common vocabulary of an area and define, with different levels of formality, the meaning of the terms and the relations between them. They aim to capture domain knowledge in a generic way and provide a commonly agreed understanding of a domain, which may be reused and shared across applications and groups [10]. Ontologies are used by people, databases, and applications that need to share domain information. There are several other definitions and typologies of ontologies; for an overview [10, 11] are good sources. Some definitions may follow from the way that ontologies are built and used; distinctions are made between lightweight and heavyweight ontologies, where taxonomies are considered to be one of the former, whereas the latter kind of ontologies would be expected to include axioms. For example Sowa [12] defines a terminological ontology as "an ontology whose categories need not not be fully specified by axioms and definition". WordNet [27] is an exmple of such an ontology. Other distinctions are based on the kind of languages used to implement ontologies, such that some ontologies are rigourously formal if they are defined in a language with formal semantics, theories and proofs (e.g. of soundness and completeness). Others are only highly informal being expresssed only in natural language. Some ontologies are intended to be reusable across domains but several are specific to a domain.

Knowledge in ontologies is mainly formalized using five kinds of components: classes, relations, functions, axioms and instances. For a description of these components refer to [10]. However, in this project we are concerned with only a specific type of simple ontology, referred to in the SCHEMAS project as a vocabulary[13]:
"*In our usage, the term evokes a semantically rich dictionary environment, with pointers to related terms – more than just a flat word list. (Another common synonym for "vocabulary" is "element set". Similarly, though we prefer to speak of metadata "terms", the term "elements" is a close synonym.)* "

Further, the SCHEMAS project developed the notion of an *Application Profile*[9] which is a type of metadata vocabulary that draws on canonical vocabularies and customizes them for local use. The precise use of the terms vocabulary and application profile and how they are modeled in our work will be expanded on in section 3.1.

## 2.1 Ontology Description Languages

Semanticweb.org [25] provides an encapsulation of the history of the representation of ontologies on the Web. More recently the OWL Web Ontology Language[22] is being designed by the W3C Web Ontology Working Group[19] in order to provide a language that can be used for applications that need to understand the content of information instead of just understanding the human-readable presentation of content. OWL facilitates greater machine readability of web content than XML, RDF and RDF Schema[18] by providing an additional vocabulary for term descriptions. The OWL language is a revision of the DAML+OIL web ontology language incorporating learnings from the design and application use of DAML+OIL[36].

### 2.1.1 RDF Schema

The Resource Description Framework (RDF) is a general-purpose language for representing information on the Web. The RDF Schema specification [18] describes how to use RDF in order to describe RDF vocabularies.

### 2.1.2 DAML+OIL

DAML+OIL [21] is a semantic markup language for Web resources. It builds on earlier W3C standards such as RDF and RDF Schema, and extends these languages with richer modelling primitives. DAML+OIL provides modelling primitives commonly found in frame-based languages. A DAML+OIL knowledge base is a collection of RDF triples. DAML+OIL prescribes a specific meaning for triples that use the DAML+OIL vocabulary

### 2.1.3 DAML+OIL

The Web Ontology Language OWL [22] is a semantic markup language for publishing and sharing ontologies on the World Wide Web. OWL is developed as a vocabulary extension of RDFS and is derived from the DAML+OIL Web Ontology Language[21]. OWL is a language for defining and instantiating *Web ontologies*. Different subsets of the OWL language are defined, to suit different uses. OWL has been designed for maximal compatibility with RDF and RDF Schema, and an OWL ontology is represented as a set of RDF triples.

### 2.1.4 RDFS(FA)

RDFS(FA)[28] as a sub-language of RDFS introduces a Fixed layered metamodeling Architecture to RDFS, based on a relatively standard model-theoretic semantics. Therefore, first order languages, like DAML+OIL and OWL, can be built on top of *both* the syntax and semantics of RDFS(FA). On the other hand, all RDFS(FA) statements are still *valid* RDFS statements, since RDFS(FA) imposes the restriction of stratification on the syntax of RDFS. It is intended to address the 'dual-roles' problem in RDF.
RDFS(FA) is designed to be a clean schema layer language (as a sub-set of RDFS), such that

- it is easy to understand and to use
- first order logics (e.g. DAML+OIL and OWL/DL) can be built on top of both its syntax and semantics

RDFS(FA) is a Semantic Web schema language introducing a UML-like metamodeling architecture to RDFS. Built-in modelling primitives of RDFS are stratified into different strata (or layers) of RDFS(FA), so that certain modelling primitives belong to certain stratums (layers). The semantics of modelling primitives depend on the stratum they belong to. All these strata form the metamodeling architecture of RDFS(FA). Theoretically there can be infinite number of layers in the metamodeling architecture, while in practice, four layers are usually described:

Stratum 0 (Instance Layer)
Stratum 1 (Ontology Layer)
Stratum 2 (Language Layer)

# 3 Ontology Servers and Metadata Registries

As used in the SCHEMAS Project, the term "registry" refers to a database that harvests various types of metadata vocabularies from their maintainers over the Web. In response to queries, such a registry should provide term-level documentation of definitions and usage along with contextual annotations. It should in effect function as an indexing engine for dynamically updating, merging, and serving up a large corpus of definitions for metadata terms. The context for such a registry is the notion of a Semantic Web where anybody or any organisation can declare a metadata vocabulary and assert a relationship between that vocabulary and any other vocabulary on the Web.

## 3.1 The SCHEMAS Metadata Registry

The SCHEMAS project developed a metadata registry which was implemented using the EOR toolkit (Extensible Open RDF toolkit)[37]. An RDF approach offered the potential of a scaleable system based on a common data model (RDF) both for the schema and for the database. The project was looking towards implementation of a repository which would be populated with schemas harvested directly from their maintainers in an open Web environment. However, at that time software tools for such a solution proved immature and required a level of development effort beyond that available to the project. In addition the chosen standard for schema specification (RDF Schema) was itself still under development, and conventions for expressing metadata schemas, in particular *Application Profiles*[9], were still to emerge.

The primary motivation for the work on the SCHEMAS Registry "has been to help humans find out about metadata terms in use -- their official definitions, local variations and extensions, and the various schemas in which they are embedded. The purpose is to help designers of information services discover metadata terms that have already been created or standardized by others and align their own schemas with those of related information providers." [8]. However, the longer-term goal was "to build a corpus of machine-understandable schemas that can be accessed and processed directly by various software applications" [8].

## 3.2 BT's Ontology Server

The BT Ontology Server [31] is part of the Agentcities.RTD initiative. The Agentcities Ontology Service is an agent and web application for managing and accessing DAML+OIL ontologies and can be accessed by agents using open standards (the Agentcities interoperability stack). This allows ontologies to be created, managed and shared by agents [32].

## 3.3 The Dublin Core Metadata Initiative's Registry

The Dublin Core Metadata Initiative is an open forum engaged in the development of interoperable online metadata standards that support a broad range of purposes and business models. The overall goal of the DCMI Registry Working Group[35] is the development of a metadata registry providing authoritative information regarding the DCMI vocabulary and the relationship between terms in that vocabulary. The group aims to provide an operational registry with both user and machine interfaces over a phased development period, with the aim of supporting acceptance and use of the DCMI vocabulary and providing an authoritative source of information [35]. Work in this initiative is ongoing.

## 3.4 Other Initiatives

Other intiatives within the areas of ontologies, ontology representation, storage and exchange have undertaken reviews of repositories of ontologies:

- The OntoWeb Technical RoadMap [10] reported on repositories of ontologies, listing some of the 'best-known repositories'. The ontology repositories that are described include those in which ontologies are implemented in DAML, Ontolingua and SHOE.

- More recently, the SWAD Europe Project reviewed RDF storage systems [20] including ones that may include schema and ontological data such as RDF Schema and DAML+OIL.

The DAML Repository [30] is a web-accessible catalogue of ontologies expressed in DAML.

## 3.5 The MEG Registry

The *Metadata for Education Group* (MEG)[14] was formed following a meeting of key UK stakeholders and serves as an open forum for debating the description and provision of educational resources at all educational levels across the United Kingdom. This group seeks to reach consensus on appropriate means by which to describe discrete learning objects in a manner suitable for implementation in a range of educational arenas.

Preceding work undertaken in the DESIRE[6] and SCHEMAS[7] projects provided the basis for the MEG Registry Project[15], which adopted a slightly modified data model as described in the Appendix. The aim of the MEG registry is to provide implementers of educational systems with a means to share information about their metadata schemas and to re-use existing schemas. The benefit being a saving of time and effort currently spent in researching existing schemas and in re-inventing schemas.

In the next few sections we describe in some depth the models and definitions employed in the MEG Registry project as they have provided the framework for our work.

### 3.5.1 The MEG Registry model of metadata vocabularies

The registry is based on the following model of metadata vocabularies or element sets:

**Element Sets** are owned and maintained by **Agencies**. **Element Sets** are made up of **Elements**. An
**Element Usage** may:

- introduce constraints on the value of an **Element** by associating 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.

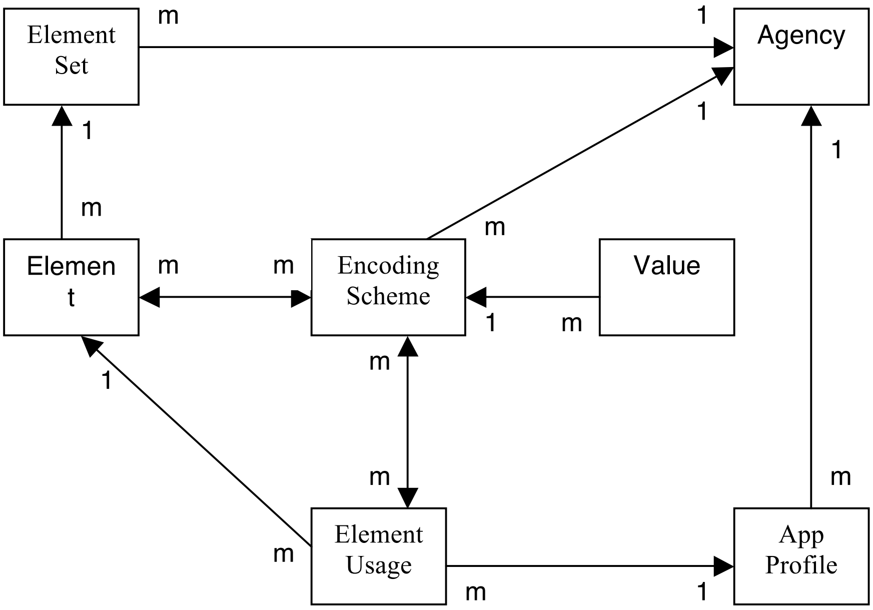
**Encoding Schemes** constrain the value space of **Elements**. An **Application Profile** defines a set of **Element Usages** of **Elements** drawn from one or more **Element Sets**.

The registry holds information on each of the entities and their relationships:

- **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;
- the **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;
- the **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; where an Encoding Scheme takes the form of an enumerated list, the **values** prescribed by that Encoding Scheme may be recorded;

- the **Agencies** who own/create/maintain Element Sets, Application Profiles, and Encoding Schemes

Diagrammatically, the relationship between the entities that are represented in the registry is modelled as follows (a more formal description is available in the Appendix).



The Meg Registry is implemented as a server based on the RDF toolkit, Redland [16]. The information about the above entities and their relationship is stored and made available in machine-processible format as RDF schemas. The existing registry API is developed in Perl and supports functions such as querying of the registry through an HTTP interface. The project also provided a tool that could support the creation and submission of metadata schemas in a distributed way, in particular promoting the re-use of elements and encoding schemes as described in [17].

The registry can be queried either through the schema creation tool so as to identify elements and encoding schemes for re-use, or directly through the HTTP APIs. One of the interfaces was intended for browsing and searching through a web browser, and returns HTML encoded representations of the structures and relationships of the element sets and related entites, which support easy navigation through the registry. Thus each of the entites (agency, element set, element, application profile, element usage and encoding schema) can be either searched or browsed and the relationships can be explored.

A second interface supports queries to search against element sets and encoding schemes, and returns RDF-encoded data.

# 4 The UKOLN Ontology Server

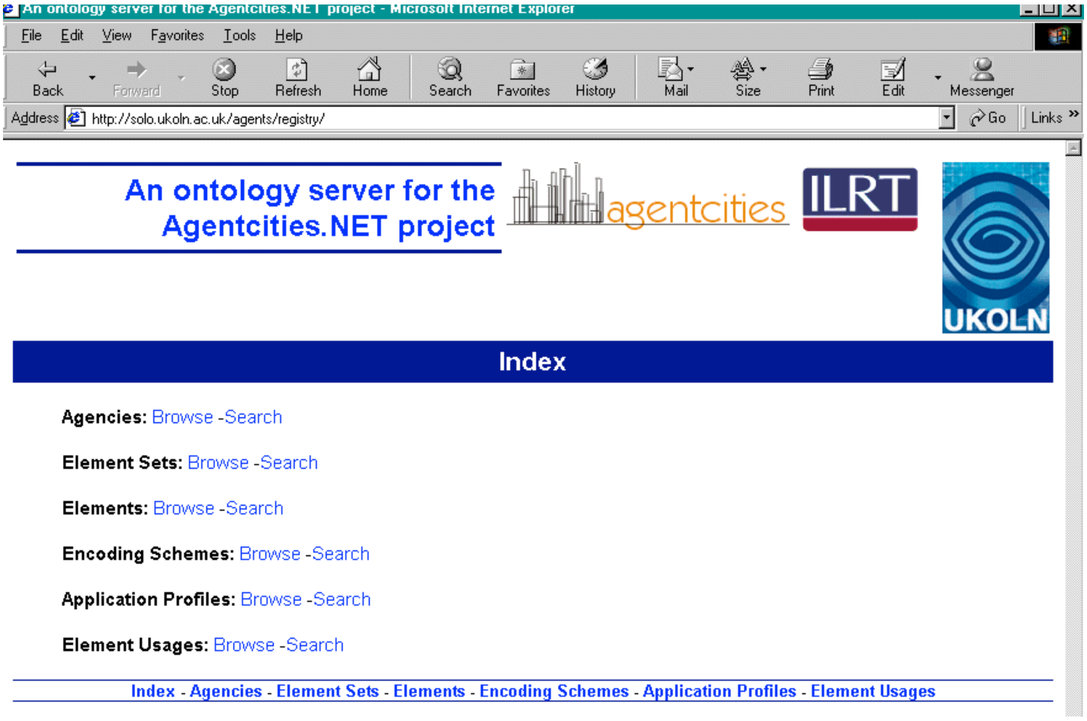
Recently, we have extended the work done in the MEG Registry project to re-deploy the interfaces to the registry within an agent environment, namely the Agentcities.NET[1]. The existing registry software stores information pertaining to metadata vocabularies and provides an interface for interacting with the information. We have thus transitioned from a human-centric to an agent-centric environment.

We have deployed the MEG Registry software within an agent-enabled environment, mediating communication to the registry of schemas through an agent. The schemas (or element sets) are modelled within the Server as outlined in previous sections and in the Appendix. Exploration of the element sets is organised around the categories described by the model, (i.e. agency, element, element set, application profile, encoding scheme and element usage).

## 4.1 Web Interface

Independent of the agent interface, the Server can also be explored through a web interface, which is linked from the web page: http://www.ukoln.ac.uk/metadata/agentcities/.

The following screen shots illustrate browsing of the Server using a web browser:


An ontology server for the Agentcities.NET project - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Messenger
Address
Go Links
agentcities
ILRT
UKOLN

http://solo.ukoln.ac.uk/agents/registry/

An ontology server for the Agentcities.NET project

Index

**Agencies:** Browse -Search

**Element Sets:** Browse -Search

**Elements:** Browse -Search

**Encoding Schemes:** Browse -Search

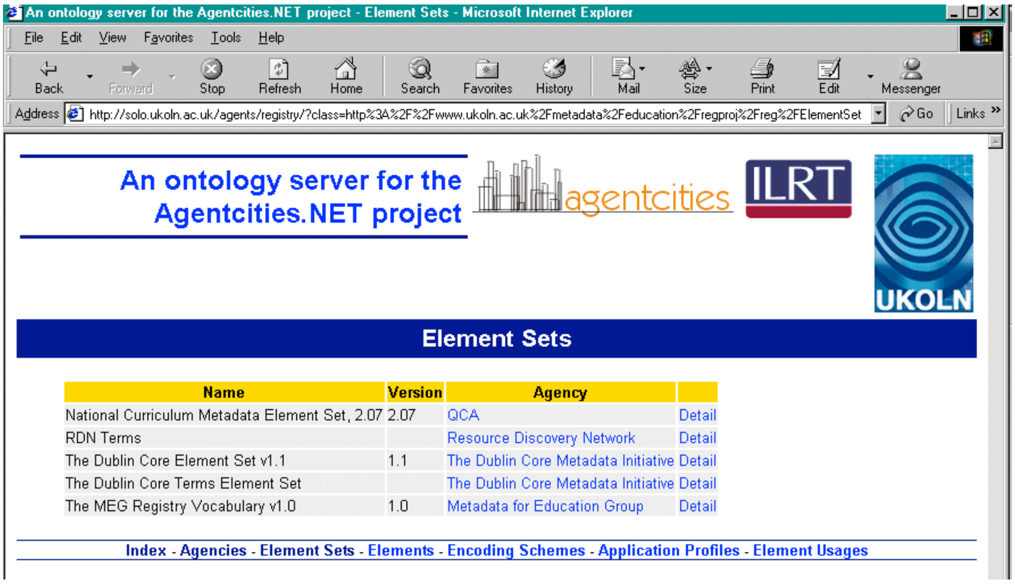
**Application Profiles:** Browse -Search

**Element Usages:** Browse -Search

**Index** - **Agencies** - **Element Sets** - **Elements** - **Encoding Schemes** - **Application Profiles** - **Element Usages**

**Figure 1**: The starting page for exploring the Server

Browsing a category reveals a list of all the resources of that class, with links to further detail


An ontology server for the Agentcities.NET project - Element Sets - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Messenger
Address
Go Links
agentcities
ILRT
UKOLN

http://solo.ukoln.ac.uk/agents/registry/?class=http%3A%2F%2Fwww.ukoln.ac.uk%2Fmetadata%2Feducation%2Fregproj%2Freg%2FElementSet

An ontology server for the Agentcities.NET project

Element Sets

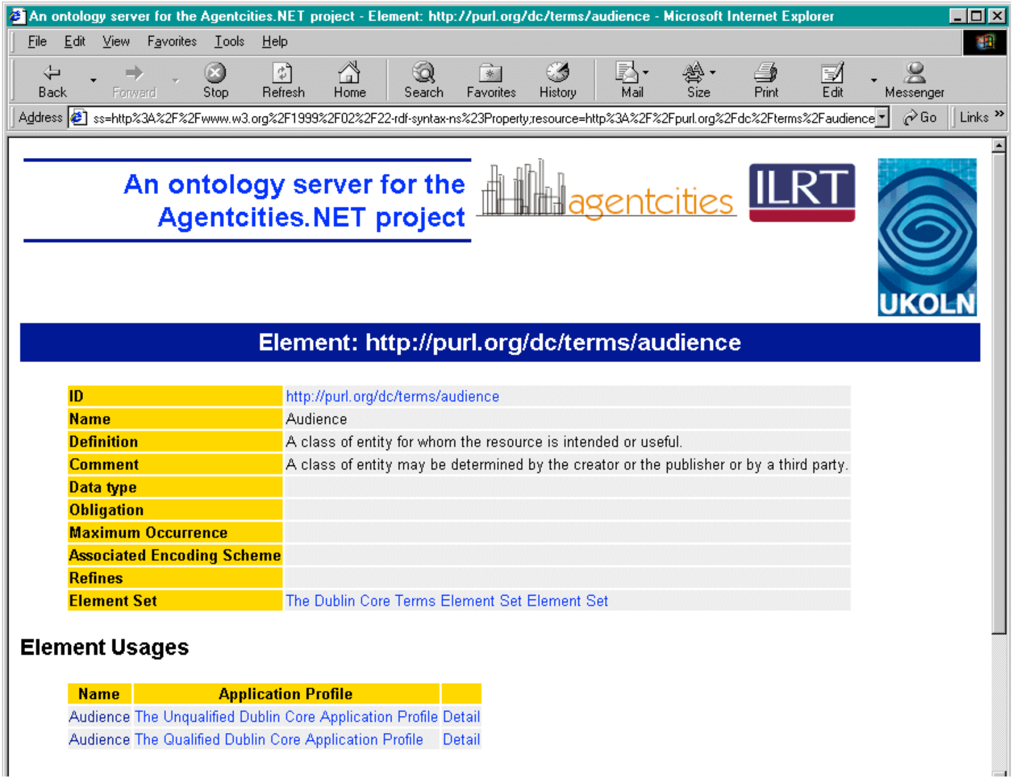
| Name | Version | Agency | |
|---|---|---|---|
| National Curriculum Metadata Element Set, 2.07 | 2.07 | QCA | Detail |
| RDN Terms | | Resource Discovery Network | Detail |
| The Dublin Core Element Set v1.1 | 1.1 | The Dublin Core Metadata Initiative | Detail |
| The Dublin Core Terms Element Set | | The Dublin Core Metadata Initiative | Detail |
| The MEG Registry Vocabulary v1.0 | 1.0 | Metadata for Education Group | Detail |

**Index** - **Agencies** - **Element Sets** - **Elements** - **Encoding Schemes** - **Application Profiles** - **Element Usages**

**Figure 2**: Browsing the list of all element sets in the Server

When browsing a specific resource, the details from the RDF description of that resource are displayed,

as well as links to related resources.


An ontology server for the Agentcities.NET project - Element: http://purl.org/dc/terms/audience - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Messenger
Address
Go
Links
An ontology server for the Agentcities.NET project
agentcities
ILRT
UKOLN

Address: ss=http%3A%2F%2Fwww.w3.org%2F1999%2F02%2F22-rdf-syntax-ns%23Property;resource=http%3A%2F%2Fpurl.org%2Fdc%2Fterms%2Faudience

Element: http://purl.org/dc/terms/audience

| | |
|---|---|
| ID | http://purl.org/dc/terms/audience |
| Name | Audience |
| Definition | A class of entity for whom the resource is intended or useful. |
| Comment | A class of entity may be determined by the creator or the publisher or by a third party. |
| Data type | |
| Obligation | |
| Maximum Occurrence | |
| Associated Encoding Scheme | |
| Refines | |
| Element Set | The Dublin Core Terms Element Set Element Set |

Element Usages

| Name | Application Profile | |
|---|---|---|
| Audience | The Unqualified Dublin Core Application Profile | Detail |
| Audience | The Qualified Dublin Core Application Profile | Detail |

**Figure 3**: Looking at the details of a specific element

## 4.2 The UKOLN Agent Platform

Our implementation work has been carried out using the JADE agent platform. JADE is one of the recommended platforms for developing agent systems. It is a software development platform aimed at developing multi-agent systems and applications conforming to FIPA standards for intelligent agents. It includes two main products, a FIPA-compliant agent platform and a package to develop Java agents. JADE has provided the environment within which to deploy the ontology service and for building agents.

Our platform has been registered with the platform directory at www.agentcities.net. Our platform name is ukoln.agentcities.net[2].

## 4.3 Overview of functionality

The Server Agent runs on the UKOLN agent platform and communicates with the Server using the Server API (over HTTP). It retrieves information on element sets and returns this information in response to requests from other agents.

We have modified the APIs from the MEG Registry software to support search and browse functions against agency, element set, element, application profile, element usage and encoding scheme. Results are returned as RDF-encoded data, rather than HTML. This is possible since the native store of the Server stores the element set descriptions as RDF, and uses the Redland RDF toolkit within the HTTP APIs.

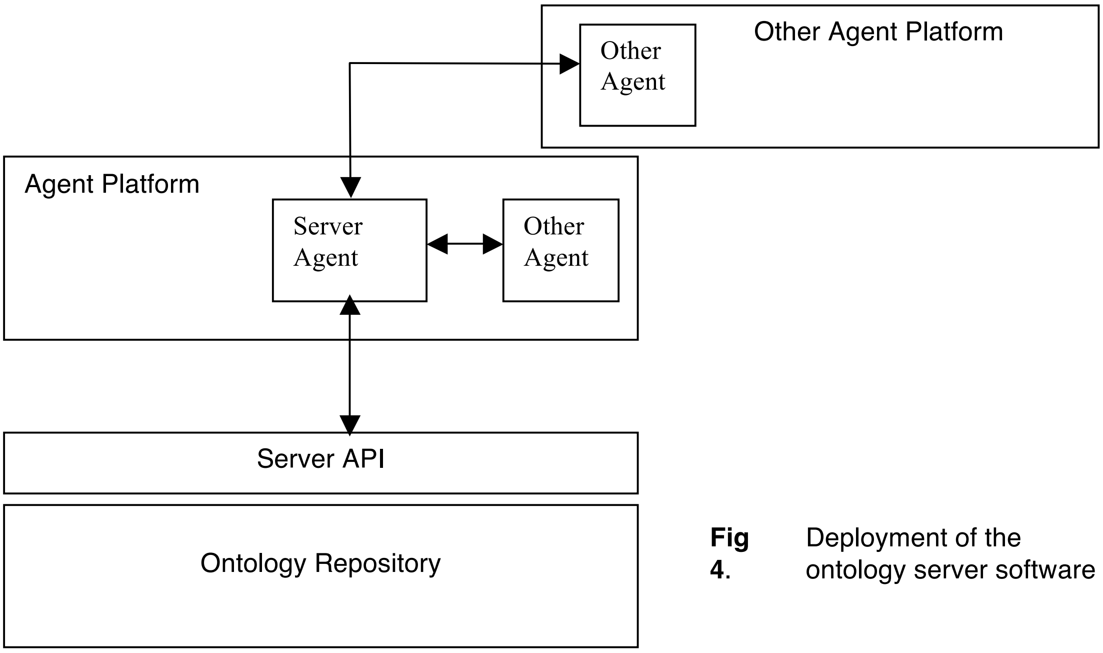


**Fig 4.** Deployment of the ontology server software

The Server Agent and two examples of requester agents are now described.

### 4.3.1 The Server Agent

The Server Agent can carry out search and browse requests on behalf of other agents, and passes on the results from the Server to the requester agents.

**Search**

Searches are carried out within a specific category (e.g. agency or elements) and the search term is matched with any part of the text between the RDF tags making up a description. If a part of the description matches, the whole description for that resource is returned in the result set. When the description is that of an element, the description of the associated element set is also presented.

**Browse**

Using the browse function, either a whole category is explored, or a specifically named resource from a category is specified. The RDF descriptions for all the resources in a category, or for a single resource are returned respectively.

Examples of the RDF (returned in response to both of these kinds of queries) are illustrated in the following sections.

**Implementation**

*Behaviours*

The Server Agent is implemented using one behaviour. This behaviour is cyclic and will wait for a message with a REQUEST performative. On receiving such a message, the behaviour

1. extracts components of the request (using an ontology)
2. constructs a URL from the request
3. connects to the Server using the URL
4. reads the response from the Server
5. places response into a reply message

Basic error checking is performed. Incorrect content or an unexpected performative will result in a NOT_UNDERSTOOD message being returned to the sender. At present, other error conditions are simply caught within the Java exception mechanism and reported on the System.err stream.

Thus the behaviour deals with one request at a time, sending a reply before attending to the next request message in the agent queue.
A more complex model of behaviour, for example starting a new agent or behaviour to deal with each request, was unnecessary at this stage, given the simple functionality of the Server and the agent. In a service level Server, the issue of how to deal with a large number of requests in a responsive manner would become important. The performance of a large Server capable of complex querying would also have to be taken into account, but to date such registries are largely an unknown factor.

### 4.3.2 Server Ontology

We have defined a simple ontology (ServerSearchOntology) in which requests to the Server Agent can be expressed. This ontology is intended to encapsulate the simple kinds of requests supported by the Server that we have experimented with, and is not intended to be an exhaustive or comprehensive ontology for all the kinds of queries that schema registries should or could support.

The ontology consists of two Action concepts, ReturnSearchResults and ReturnBrowseResults. The ReturnSearchResults action emulates a search request through a web browser; ReturnSearchResults has a searchRequest, made up of a Scope and a searchTerm. The scope limits the search for the searchTerm (which is a string) to one of the categories (agency etc.). ReturnBrowseResults emulates the browsing action carried out through the web browser. Thus a browseRequest takes a Scope (one of agency, element set, element, application profile, element usage and encoding schema) and a specific resource URI. The resource URI identifies a specific instance of the entity (e.g. a particular agency) and if a specific resource URI is specified in the browse request, the RDF description for that resource alone is returned.
If no resource URI is specified, the RDF descriptions of all the instances of that category are returned in a list (e.g. all the agencies are listed). The examples illustrate this behaviour.

***Examples***

Example 1: An encoding of a **search** request for the **term** "network" within the **scope** "agency":

```
(
        (action
                (agent-identifier :name UKOLNServer@solo.ukoln.ac.uk:1099/JADE)
                (ReturnSearchResults
                        (Search   :Scope agency    :SearchTerm network)
                )
        )
)
```

The RDF description of an agency with the term resource in its name is returned:

```
<rdf:Description rdf:about="http://purl.org/rdn/RDN/">
    <rdf:type
rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/Agency"/
>
    <reg:agencyName>Resource Discovery Network</reg:agencyName>
    <reg:agencyHomepage rdf:resource="http://www.rdn.ac.uk/"/>
  </rdf:Description>
```

Example 2: A **search** for the **term** "audience" in the **element** category .

```
(       (action
                (agent-identifier :name UKOLNServer@solo.ukoln.ac.uk:1099/JADE)
                (ReturnSearchResults
                        (Search : Scope element : SearchTerm audience)
                )
        )
)
```

This search finds two elements. In the first element the search term 'audience' is found within the useComment tag. The second element is the Audience element in the Dublin Core (The search term is highlighted here for emphasis). Both these elements are part of the Dublin Core Terms element set and the description for the element set is returned at the end.

```
<rdf:Description rdf:about="http://purl.org/dc/terms/mediator">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
    <rdfs:label>Mediator</rdfs:label>
    <rdfs:comment>A class of entity that mediates access to the resource and
for whom the resource is intended or useful.</rdfs:comment>
    <reg:useComment>The audience for a resource in the education/training
domain are of two basic classes: (1) an ultimate beneficiary of the resource
(usually a student or trainee), and (2) frequently,             an entity
that mediates access to the resource (usually a teacher or trainer).  The
mediator element refinement represents the second of these two
classes.</reg:useComment>
    <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/terms/audience"/>
    <reg:isElementOf
rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/elementS
et/dcterms"/>
  </rdf:Description>

<rdf:Description rdf:about="http://purl.org/dc/terms/audience">
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
    <rdfs:label>Audience</rdfs:label>
    <rdfs:comment>A class of entity for whom the resource is intended or
useful.</rdfs:comment>
    <reg:useComment>A class of entity may be determined by the creator or
the publisher or by a third party.</reg:useComment>
    <reg:isElementOf
rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/elementS
et/dcterms"/>
  </rdf:Description>

<rdf:Description
rdf:about="http://www.ukoln.ac.uk/metadata/education/regproj/reg/elementSet/
dcterms">
    <rdf:type
rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/ElementS
et"/>
    <dc:title>The Dublin Core Terms Element Set</dc:title>
    <dcterms:created>2000-07-11</dcterms:created>
    <reg:status>DCMI recommendation</reg:status>
    <dc:description>
```

The Dublin Core metadata vocabulary is a simple vocabulary intended to facilitate discovery of resources.

```
 </dc:description>
  <reg:responsibleAgency
rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/agency/d
cmi"/>
    <reg:xmlNamespacePrefix>dcterms:</reg:xmlNamespacePrefix>
    <reg:specification
rdf:resource="http://dublincore.org/usage/terms/terms-latest.html"/>
  </rdf:Description>
```

Example 3:A **browse** request for the whole of the **agency** category (no Resource URI is given)

```
(
        (action
            (agent-identifier :name UKOLNServer@solo.ukoln.ac.uk:1099/JADE)
            (ReturnBrowseResults
                (Browse     :Scope agency  :Resource "")
            )
    )
)
```

Returns a list of all the agencies (descriptions encoded in RDF)

```
<rdf:Description
rdf:about="http://www.ukoln.ac.uk/metadata/education/regproj/reg/agency/iso"
>
    <rdf:type
rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/Agency"/
>
    <reg:agencyName>International Standards Organisation</reg:agencyName>
  </rdf:Description>

<rdf:Description
rdf:about="http://www.ukoln.ac.uk/metadata/education/regproj/reg/agency/lc">
    <rdf:type
rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/Agency"/
>
    <reg:agencyName>Library of Congress</reg:agencyName>
  </rdf:Description>

  <rdf:Description
rdf:about="http://www.ukoln.ac.uk/metadata/education/regproj/reg/agency/meg"
>
    <rdf:type
rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/Agency"/
>
    <reg:agencyName>Metadata for Education Group</reg:agencyName>
    <reg:agencyHomepage
rdf:resource="http://www.ukoln.ac.uk/metadata/education"/>
  </rdf:Description>

  <rdf:Description
rdf:about="http://www.ukoln.ac.uk/metadata/education/regproj/reg/agency/oclc
">
    <rdf:type
rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/Agency"/
>
    <reg:agencyName>OCLC</reg:agencyName>
  </rdf:Description>

<rdf:Description rdf:about="http://purl.org/rdn/RDN/">
    <rdf:type
rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/Agency"/
>
    <reg:agencyName>Resource Discovery Network</reg:agencyName>
    <reg:agencyHomepage rdf:resource="http://www.rdn.ac.uk/"/>
  </rdf:Description>

 (elided)

  <rdf:Description
rdf:about="http://www.ukoln.ac.uk/metadata/education/regproj/reg/agency/dcmi
">
    <rdf:type
rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/Agency"/
>
    <reg:agencyName>The Dublin Core Metadata Initiative</reg:agencyName>
    <reg:agencyHomepage rdf:resource="http://dublincore.org/"/>
  </rdf:Description>
  <rdf:Description
rdf:about="http://www.ukoln.ac.uk/metadata/education/regproj/reg/agency/w3">
    <rdf:type
rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/Agency"/
>
    <reg:agencyName>World Wide Web Consortium</reg:agencyName>
  </rdf:Description>
```

Example 4: A browse request for a specific resource (http://purl.org/dc/terms/MESH/) from the encoding scheme category.

```
(       (action
                (agent-identifier :name UKOLNServer@solo.ukoln.ac.uk:1099/JADE)
                (ReturnBrowseResults
                        (Browse         :Scope encodingscheme         :Resource
                http://purl.org/dc/terms/MESH)
                )
        )
)
```

## 4.4 Interrogating the Server Agent

We have implemented two examples of Requester Agents, both of which are driven by a human user and make requests to the Server Agent. These two agents use the ServerSearchOntology to communicate requests to the Server Agent, and display the response returned by the Server. Results to queries are contained within the content slot of an INFORM message from the Server Agent, and consist of RDFS descriptions. Thus the ontology is only used to communicate requests; responses are simply wrapped up in the content slot of the message.

### 4.4.1 The GUI Agent

This agent presents the user with a graphical interface implemented with Java Swing. This is realized through two classes:

ServerAgentGui class extends the Swing JFrame class, and defines the appearance of the interface;

ServerGuiAgent class extends the Jade GuiAgent class, and defines the behaviours that are instantiated in response to user actions at the interface.

Each instance of the Agent class is associated with one instance of the Gui class (and vice versa).

The appearance of the interface is shown in Figure 5. It contains the following main components:

- a pull-down list of categories
- a button for triggering the display of a whole category (the latter obscured in the first screenshot)
- a text entry for resource URIs, and an associated button for displaying;
- a text entry for search terms, with an associated search button
- a display area for results

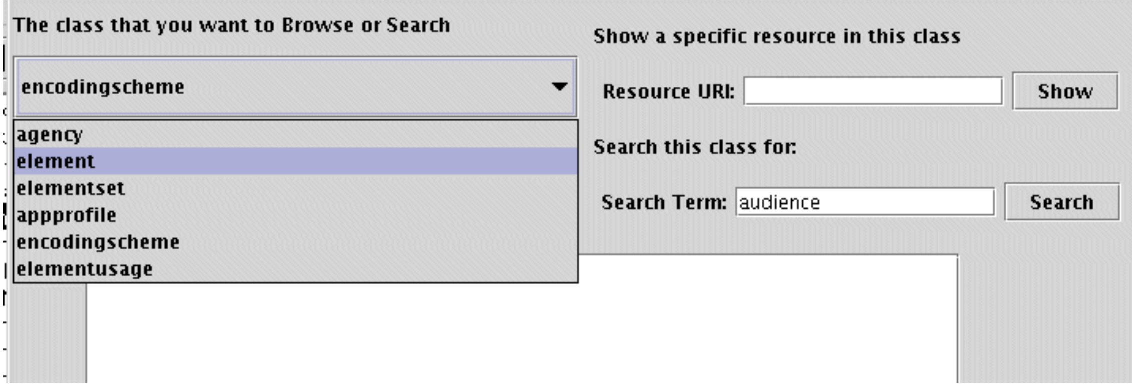
After selecting a category, the user can then choose to browse the whole category, or to enter a resource URI for a known resource. Alternatively, the search box can be used to interrogate the Server. The three tasks that the interface supports reflect the kinds of requests that can be expressed in the ServerSearchOntlogy:

The ServerAgentGui class implements ActionListener; on Action events, the handler (ActionPerformed) invokes the JADE postGUIEvent method to communicate with the ServerGuiAgent class; this is the path by which user actions on the interface trigger behaviours in the agent.

Within the agent, the onGuiEvent method handles the events from the interface (invoked through postGuiEvent). A message is built (using the ServerSearchOntology) corresponding to the action invoked; the message is sent using a SenderBehaviour (which extends OneShotBehaviour). A cyclic behaviour listens for response messages from the ServerAgent and when an INFORM message arrives, it invokes a displayResults method in the gui, so that the content of the message (containing RDF-encoded descriptions) is displayed (Figure 6).

The interface has been design to support one outstanding request at a time. In theory multiple requests could be launched before the first response arrives, and at present there is no control to prevent this. In practice the system response is sufficiently fast that no major control is required at present to synchronise requests and responses. If such control were required, this could best be implemented through the Gui by disabling the sending controls until a response is received. An alternative would be an interface that supported multiple outstanding requests, but this would require a more complicated design that is beyond the scope of the present project. This also requires a more complicated coordination model between the interface and the agent(s) for managing requests.

The link between the ServerGUIAgent and the Server Agent is hardwired and the Server Agent is assumed to be running locally.


The class that you want to Browse or Search

encodingscheme

agency
element
elementset
appprofile
encodingscheme
elementusage

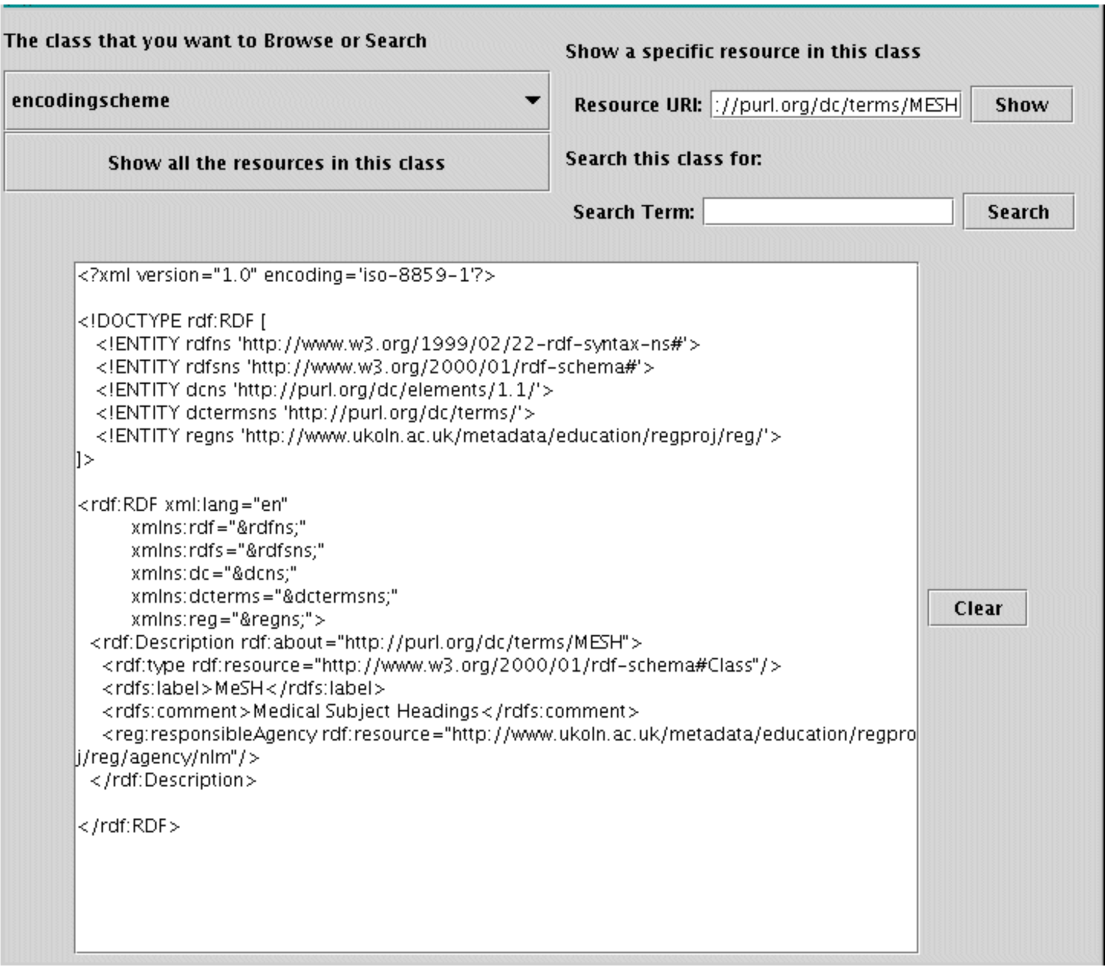
Show a specific resource in this class

Resource URI:

Search this class for:

Search Term: audience

**Figure 5**. Using the interactive GUI of the ServerGuiAgent to enter requests



The class that you want to Browse or Search: encodingscheme

Show a specific resource in this class

Resource URI: ://purl.org/dc/terms/MESH

Search Term:

```
<?xml version="1.0" encoding='iso-8859-1'?>

<!DOCTYPE rdf:RDF [
  <!ENTITY rdfns 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <!ENTITY rdfsns 'http://www.w3.org/2000/01/rdf-schema#'>
  <!ENTITY dcns 'http://purl.org/dc/elements/1.1/'>
  <!ENTITY dctermsns 'http://purl.org/dc/terms/'>
  <!ENTITY regns 'http://www.ukoln.ac.uk/metadata/education/regproj/reg/'>
]>

<rdf:RDF xml:lang="en"
       xmlns:rdf="&rdfns;"
       xmlns:rdfs="&rdfsns;"
       xmlns:dc="&dcns;"
       xmlns:dcterms="&dctermsns;"
       xmlns:reg="&regns;">
  <rdf:Description rdf:about="http://purl.org/dc/terms/MESH">
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:label>MeSH</rdfs:label>
    <rdfs:comment>Medical Subject Headings</rdfs:comment>
    <reg:responsibleAgency rdf:resource="http://www.ukoln.ac.uk/metadata/education/regpro
j/reg/agency/nlm"/>
  </rdf:Description>

</rdf:RDF>
```

**Figure 6**. Results are displayed in a window in the GUI.

### 4.4.2 The Command Line agent

A second Agent Class, ServerRequesterAgent, has been provided to interact with the user through the command line. On setup() this agent first establishes which Server the user would like to use, with a choice of either the UKOLN Server, or a local one.

### 4.4.3 Behaviours

The Agent then instantiates a main sequential behaviour (HandleRequestsBehaviour) which prompts for and reads input from the terminal. The onStart() method of the main behaviour interacts with the user to define what kind of transaction the user is performing (browse or search) and its parameters: scope, search term or resource URI:

```
ENTER the local name of the Server agent or press enter to use the
UKOLN Server-->
ENTER s for search or b for browse -->
s
Class to Search ---> element
Enter a SearchTerm ---> audience
```

A suitable message is then built and a Sender Behaviour is scheduled (as a sub behaviour) to send the message to the Server Agent. The next subBehaviour added then handles the response from the Server Agent and displays the result to the user.

The onEnd() method then checks if the user would like to carry out another transaction. If the user stops, the agent is terminated; if the user wishes to continue, all the behaviours are reset.

# 5 Conclusions

We have successfully deployed an ontology server onto the Agentcities.NET network, where it is available for either browsing over the Web or querying by agents. It should be noted that the server accepts metadata vocabularies encoded in RDF Schema. Further, the vocabularies need to adhere to the model described in the Appendix. The work presented has advanced the work begun in previous projects to investigate an approach based on automated querying and processing of simple ontologies by software agents rather than through human interaction.

# 6 Acknowledgements

The software used in this project for the ontology server was originally developed in the MEG Registry project which was funded by JISC and BECTa. The ideas in this project have been developed from work in the DESIRE, SCHEMAS and MEG Registry Projects. Thanks to Pete Johnston, UKOLN, University of Bath, for help with the MEG Registry and associated software. Thanks also to Owen Cliff, Department of Computer Science, University of Bath for assistance with setting up the UKOLN server.

UKOLN is funded by Resource: The Council for Museums, Archives & Libraries, the Joint Information Systems Committee (JISC) of the Higher and Further Education Funding Councils, as well as by project funding from the JISC and the European Union. UKOLN also receives support from the University of Bath where it is based.

# 7 References

[1] UKOLN web-page for Agentcities.NET deployment project
http://www.ukoln.ac.uk/metadata/agentcities/

[2] UKOLN Ontology Server for the Agentcities.NET network
http://solo.ukoln.ac.uk/agents/server/

[3] UKOLN
http://www.ukoln.ac.uk/

[4] Agentcities.NET
http://www.agentcities.org/EUNET/

[5] Dublin Core Metadata Initiative (DCMI)
http://www.dublincore.org/

[6] DESIRE
http://www.ukoln.ac.uk/metadata/desire/

[7] SCHEMAS Project
http://www.schemas-forum.org/

[8] Thomas Baker, Makx Dekkers, Rachel Heery, Manjula Patel, Gauri Salokhe, *What Terms Does Your Metadata Use? Application Proifles as Machine Understandable Narratives*, Journal of Digital Information Vol 2 (2), November 2001
http://jodi.ecs.soton.ac.uk/Articles/v02/i02/Baker/baker-final.pdf

[9] Heery H and Patel M., *Application Profiles: mixing and matching metadata schemas,* Ariadne Issue 25, September 2000
http://www.ariadne.ac.uk/issue25/app-profiles/intro.html

[10] OntoWeb Technical Roadmap v 1.0
http://babage.dia.fi.upm.es/ontoweb/wp1/OntoRoadMap/index.html

[11] Numbered Hypernotes in J.Hendler
http://www.sciencemag.org/cgi/content/full/299/5606/520?ijkey=1BUgJQXW4nU7Q&keytype=ref&siteid=sci
http://www.sciencemag.org/cgi/content/full/299/5606/520?ijkey=1BUgJQXW4nU7Q&keytype=ref&siteid=sci#note9

[12] Sowa, J.F. Building, Sharing and Merging Onotologies: Glossary
http://www.jfsowa.com/ontology/ontoshar.htm#s6

[13] SCHEMAS Project Glossary
http://www.schemas-forum.org/info-services/d74.htm

[14] MEG Website
http://www.ukoln.ac.uk/metadata/education/

[15] MEG Registry Project
http://www.ukoln.ac.uk/metadata/education/regproj/

[16] Beckett D., *The Design and Implementation of the Redland RDF Application Framework* Proceedings of WWW10, Hong Kong, May 2-5 2001
http://www10.org/cdrom/papers/frame.html

[17] *The MEG Registry and SCART: complementary tools for creation, discovery and re-use of metadata schemas.* Rachel Heery, Pete Johnston, Dave Beckett (ILRT, University of Bristol) & Damian Steer (ILRT, University of Bristol) - October 2002
In: Proceedings of the International Conference on Dublin Core and Metadata for e-Communities, 2002. Florence: Firenze University Press, 2002, pp. 125-132.
http://www.bncf.net/dc2002/program/ft/paper14.pdf

[18] RDF Schemas
http://www.w3.org/TR/rdf-schema/
[Note: this is the latest version of the W3C Working Draft, released on 12 November 2002, which is a work in progress; The registry development took place before the release of this draft.]

[19] W3C Web Ontology Working Group
http://www.w3.org/2001/sw/WebOnt/

[20] SWAD-Europe: Scalability and Storage: Survey of Free Software /Open Source RDF storage systems
http://www.w3.org/2001/sw/Europe/reports/rdf_scalable_storage_report/

[21] DAML+OIL
http://www.w3.org/TR/daml+oil-reference

[22] OWL
http://www.w3.org/TR/owl-ref/

[23] OWL2
http://www.w3.org/TR/2003/WD-webont-req-20030203/

[24] Wilson, Michael
http://www.w3c.rl.ac.uk/pasttalks/slidemaker/EPS_DTI/Overview.html

[25] SemanticWeb.org
http://www.semanticweb.org/knowmarkup.html#ontologies

[26] Hendler Web version of IEEE article
http://www.cs.umd.edu/~hendler/AgentWeb.html

[27] WordNet
http://www.semanticweb.org/library/

[28] RDFS(FA)
http://dl-web.man.ac.uk/rdfsfa/

[29] The Meg Registry Client Software (SCART)
http://www.ukoln.ac.uk/metadata/education/regproj/scart/

[30] DAML Repository
http://www.daml.org/ontologies/

[31] BT Ontology Server
http://193.113.27.14/ontology-server-demo/index.jsp

[32] BT Ontology Server Service Description
http://193.113.27.14/services/OntologyService/ServiceDescription.htm

[33] Another Ontology Page
http://burningluigi.com/another_ontology_page/aop.htm

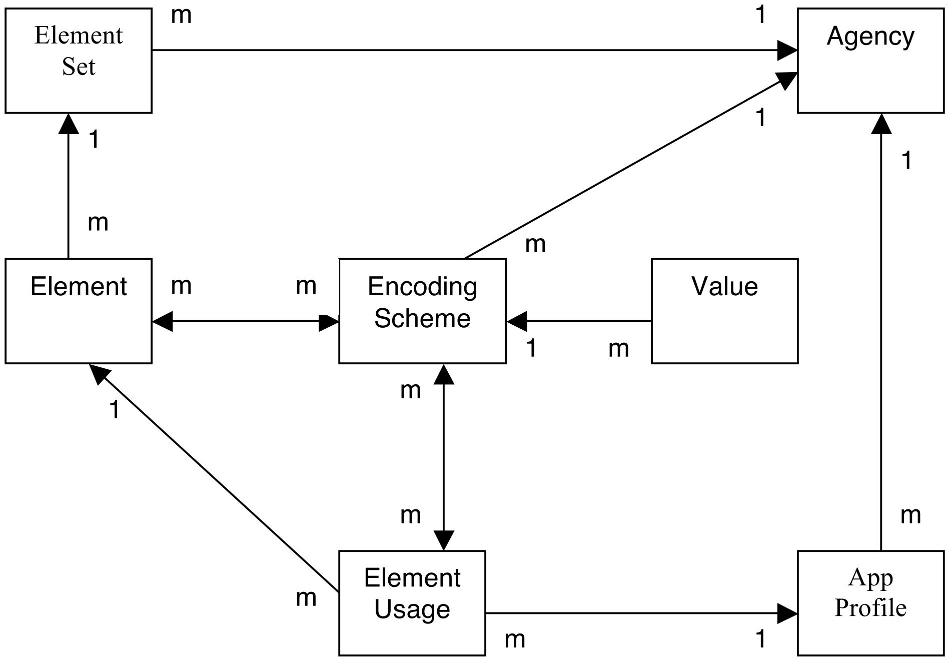
[34] SCHEMAS: Best practice guidelines for managing a registry
http://www.schemas-forum.org/info-services/d52.htm

[35] DCMI Registry Working Group
http://uk.dublincore.org/groups/registry/

[36] OWL Overview
http://www.w3.org/TR/2002/WD-owl-features-20020729/

[37] EOR (Extensible Open RDF) Toolkit
http://eor.dublincore.org/

# Appendix: The MEG Registry Data Model



**Agency:** *An organisation or individual responsible for managing one or more Element Sets, Application Profiles or Encoding Schemes*

Relationships

Element Set → *is-Managed-By* (m-1) → **Agency**
Encoding Scheme → *is-Managed-By* (m-1) → **Agency**
Application Profile → *is-Managed-By* (m-1) → **Agency**

Agency Properties

| | |
|---|---|
| Identifier (URI) | |
| Name | The name or title of the Agency |
| Home Page URL | A source of further info about the Agency |

**Element Set:** *A set of metadata Elements that is managed as a coherent unit by an Agency. The Elements of an Element Set are "functionally" related, by virtue of having been defined for the purpose of usefully describing the characteristics of a resource*

Relationships

**Element Set** → is-Managed-By (m-1) → Agency
Element → is-Element-Of (m-1) → **Element Set**

Element Set Properties

Identifier (URI)

| | |
|---|---|
| Title | The name or title of the Element Set |
| Version | The version of the Element St |
| Date created | Date this version created |
| Status | Draft/recommendation etc |
| Description | Including any notes of scope/purpose |
| Classification | |
| Specification | Prose description of/guidelines for use of Element Set |

**Element:** *A formally defined term that is used to describe a characteristic or attribute of a resource*

<u>Relationships</u>

**Element** → is-Element-Of (m-1) → Element Set
**Element** → associated-Encoding-Scheme (m-m) → Encoding Scheme
**Element** → refines (m-1) → **Element**
Element Usage → uses (m-1) → **Element**

<u>Element Properties</u>

| | |
|---|---|
| Identifier (URI) | |
| Name | A human-readable version of the property name |
| Definition | A statement that clearly represents the concept and essential nature of the Element |
| Comment | A remark concerning the application/use of the data element |
| Data type | Indicates the type of data that can be represented in the value of the data element |
| Obligation | Indicates whether the Element is always or sometimes required to be present |
| Maximum occurrence | Indicates any limit to the repeatability of the Element |

**Encoding Scheme:** *A set of contextual information or parsing rules that aids in the interpretation of the value of a metadata Element. Encoding Schemes include*

- *controlled vocabularies, which enumerate a list of values, and;*
- *formal notations or parsing rules, which define precisely how a lexical representation of a value is to be interpreted*

<u>Relationships</u>
**Encoding Scheme** → is-Managed-By Agency (m-1) → Agency
Element → associated-Encoding-Scheme (m-m) → **Encoding Scheme**
Element Usage → associated-Encoding-Scheme (m-m) → **Encoding Scheme**
Value –type (m-1) → **Encoding Scheme**

<u>Encoding Scheme Properties</u>

| | |
|---|---|
| Identifier (URI) | |
| Name | The name or title of the Encoding Scheme |
| Version | The version of the Encoding Scheme |
| Date created | Date this version created |
| Status | Draft/recommendation etc |
| Description | Including any notes of scope/purpose |
| Classification | |
| Specification | Prose description of/guidelines for use of Encoding Scheme |

**Controlled Vocabulary Value:** *An individual value or term in a controlled vocabulary*

<u>Relationships</u>

**Value** → type (m-1) → Encoding Scheme

| | |
|---|---|
| Identifier (URI) | |
| Value | Value |
| Label | Human-readable form of value |
| Description | Explanation or definition of value |

**Application Profile:** *A set of Element Usages that is managed as a coherent unit by an Agency. An Application Profile is optimised for the resource description requirements of a particular application or context.*
*Like the Elements of an Element Set, the Element Usages within an Application Profile are "functionally" related, by virtue of having been defined for the purpose of usefully describing a resource.*
*Within an Application Profile, the Element Usages may reference Elements from multiple Element Sets*

<u>Relationships</u>

**Application Profile** → is-Managed-By Agency (m-1) → Agency
Element Usage → is-Usage-In (m-1) → **Application Profile**

<u>Application Profile Properties</u>

| | |
|---|---|
| Identifier (URI) | |
| Title | The name or title of the Application Profile |
| Version | The version of the Application Profile |
| Date created | Date this version created |
| Status | Draft/recommendation etc |
| Description | Including any notes of scope/purpose |
| Classification | |
| Associated XML Schema | |
| Specification | Prose description of/guidelines for use of Application Profile |

**Element Usage:** *A deployment of a (previously defined) metadata Element in the context of a particular domain or application. The used Element may be tailored for the context by:*

- *a narrowing of its semantic definition;*
- *association with specified datatypes or Encoding Schemes;*
- *specification of obligation/occurrence constraints*

<u>Relationships</u>

**Element Usage** → is-Usage-In (m-1) → Application Profile
**Element Usage** → uses (m-1) → Element
**Element Usage** → associated-Encoding-Scheme (m-m) → Encoding Scheme

<u>Element Usage Properties</u>

| | |
|---|---|
| Identifier (URI) | |
| Name | A human-readable version of the Element name. |
| Definition | A statement that clearly represents the concept and essential nature of the Element |
| Comment | A remark concerning the application/use of the Element. |
| Data type | Indicates the type of data that can be represented in the value of the Element |
| Obligation | Indicates whether the Element is always or sometimes required to be present |

| | |
|---|---|
| Maximum occurrence | Indicates any limit to the repeatability of the Element |