Alvis Task T3.1 - Peer-Description Metadata Format

Milestone M3.1 - Month 6 (June 2004)

Mike Taylor

Alvis Task T3.1 - Peer-Description Metadata Format: Milestone M3.1 - Month 6 (June 2004)
by Mike Taylor

Table of Contents

Chapter 1. Abstract

The capabilities of individual peers in an Alvis network can be described in machine-readable form using a peer-description format expressed in XML. Descriptions in this format are called peer-description records.

A complete description record includes the peer's human-readable name and unique identifier, a set of addresses it can be contacted on, an indication of what subject areas its data falls into, specifications of its support for various parts of Alvis functionality and statistics about its performance. Support specifications include indications of what query-types, indexes, ranking schemes, metadata subsets and record formats are available.

An XML DTD is available to prescribe the format of peer description records, along with sample description records illustrating the format's expressive capabilities.

Chapter 2. Introduction

The EU-funded Alvis[1] project runs for three years from 2004-2006, and is tasked with building a semantic peer-to-peer search engine.

In order for any peer-to-peer system to operate effectively, it is necessary for each peer to understand the capabilities of nearby peers, so that appropriate operations can be requested. In very simple peer-to-peer systems such as classic Gnutella, this is achieved trivially, because all peers have the same capabilities; or at least they vary only in very parameterisable ways such as their network bandwidth. In Alvis, however, different peers may vary substantially in their strengths and weaknesses; so a mechanism is needed by which any given peer can express those capabilities in a machine-readable form suitable for other peers to understand.

Task T3.1 (Network Node Metadata Framework) in Workpackage WP3 (Data Model and Standards) is to provide such a format. This document describes the format as it stands at month six (June 2004) of the Alvis project, and consitutes Milestone M3.1.

Since the Alvis proposal document was written, the consortium partners have adopted more consistent terminology, so that what WP3 referred to as "nodes" are now called "peers". Accordingly, Task 3.1 is more properly defined as the provision of a peer-description metadata format.

Since the Alvis proposal document was written, the partners have adopted consistent terminology, so that what the WP3 part of the proposal referred to as "nodes" are now called "peers". Accordingly, Task 3.1 is now more properly defined as the provision of a peer-description metadata format.

Notes

1. http://alvis.info/

Chapter 3. Syntatic Issues

Creation of a format such as this consists of two essentially independent decisions - one semantic one and syntactic. The former involves deciding what information needs to be in the record; the latter with how that information is encoded.

The real work is done in the semantic area: deciding which aspects of peers' functionality should be described, what elements should represent that functionality, how the elements should be related to each other, what prescribed volcaularies they can draw their values from, etc. In contrast, the syntactic decision is relatively straightforward, so we deal with this first.

Choice of Meta-Format

In choosing a syntax with which to represent peer-description records the desiderata are as follows, in roughly descending order of preference:

1. The syntax should be easy for computers to understand.
2. It should be easy for humans to read.
3. It should be easy for humans to write, as at least early in the project, description records may often be hand-crafted rather than automatically generated.
4. It should be compact, so as to minimise network traffic.

Of these criteria, the last is best regarded as a luxury, since peer-description records will be of near-negligible size anyway compared with the content records that Alvis peers will deal with most of the time.

Of the remaining three criteria, 2 and 3 are correlated, and 1 is not incompatible with either. These criteria might easily enough be satisfied by a custom-designed format, but practical concerns dictate that it is preferable to use an "off-the-shelf" meta-syntax, so that peers implemented in various languages can take advantage of existing parsers rather than having to use specially developed code for parsing peer descriptions.

Three strong candidate metalanguages are in widespread use today. Two of these, SGML[1] and XML[2] are very close to each other in most respects: SGML is the older and more powerful of the two. XML is a deliberately cut-down descendant of SGML because its developers felt that simplicity and uniformity were more important that expressiveness. The third contender is YAML[3], a more elegant format that places more emphasis on human-readability and is also rather less verbose.

We regretfully decided to pass up the most elegant contender, YAML, on the pragmatic grounds that it is much less widely implemented than either of the others. In recent years, the pace of development of XML utilities has hugely outstripped that of its rivals, including SGML, and the addition of extrinsic functionality such as XML Namespaces[4], XPath[5], etc. means that the expressive power of the entire XML toolset now comfortably exceeds that of SGML. Accordingly, we selected XML as the metalanguage for Alvis peer-description records.

Choice of Constraint Language

The structure of XML records can be constrained using a number of different constraint languages, including the older DTDs (document type definitions) inherited from SGML, the newer and more complex XML Schema[6], and its more intuitive but less widely deployed rival, Relax NG[7].

These are all broadly equivalent in their ability to express the acceptable structure of XML documents, though with some important differences. In contrast to the older DTD technology, XML Schema and Relax NG both include support for XML namespaces and provide more control over the values that can be used for particular XML elements and attributes.

Despite these disadvantages of DTDs, we found that they were more than counterbalanced by the ubiquitous support for DTDs in XML toolkits such as the widely-used libxml2[8] which, perhaps surprisingly does not fully support XML Schema at the time of writing.

Accordingly, we provide in this document an XML DTD prescribing the format of peer-descriptions records, and accompany this with prose specifying the controlled vocabularies of attributes that can take only a small number of of pre-defined values.

Notes

1. http://www.w3.org/MarkUp/SGML/
2. http://www.w3.org/XML/
3. http://www.yaml.org/
4. http://www.w3.org/TR/REC-xml-names/
5. http://www.w3.org/TR/xpath
6. http://www.w3.org/XML/Schema
7. http://www.relaxng.org/
8. http://www.xmlsoft.org/

Chapter 4. Sample Description Record

The following sample peer-description record serves as a motivating example for much of the discussion to follow.

```
<?xml version="1.0" ?>
<!-- $Id: peer-description.xml,v 1.11 2004/06/15 16:20:00 mike Exp $ -->
<!-- Version is separated from namespace so we can upgrade the
     version of the metadata format without needing to change
     namespace.  This is useful for maintaining backwards
     compatibility -->
<peer xmlns="http://alvis.info/peer/"
      version="1.0"
      name="Mike Taylor's Example Peer"
      id="anyOpaqueAndUniqueIdentifierAYFGDYHFGAS">
  <addresses>
    <address type="tcp" bandwidth="128">192.168.0.106:12368</address>
    <address type="mail">alvis@miketaylor.org.uk</address>
  </addresses>

  <subjectAreas>
    <!-- These indicate possible words/phrases in queries -->
    <subject type="genomics"/>
    <subject type="vertebrate paleontology"/>
    <subject type="biomechanics"/>
  </subjectAreas>

  <support>
    <!-- query-type names are taken from a well-known enumerated set -->
    <query type="cql"><!-- See http://zing.z3950.org/cql/ -->
      <!-- List of document formats that can be searched with this language -->
      <searchFormat type="text/xml"/>
      <searchFormat type="text/plain"/>
      <searchFormat type="audio/mpeg"/>
      <index>author</index>
      <index>title</index>
      <index set="http://z3950.org/dc/1.0/">date</index>
    </query>

    <query type="bag"/><!-- bag of words -->

    <query type="xpath"><!-- subset of XQuery: xpath=term -->
      <searchFormat type="text/xml"/>
    </query>

    <rank type="field">author</rank>
    <rank type="field">title</rank>
    <rank type="algorithm">tfidf</rank>

    <!-- metadata subset of object that can be returned -->
    <subset type="id"/>
    <subset type="dc"/>
    <subset type="xpath"/>
    <subset type="fulltext"/>

    <recordFormat type="text/xml">
      <schema name="Dublin Core" tag="dc">
        <spec type="xmlschema">http://foo.com/dc.xsd</spec>
        <spec type="dtd">http://bar.com/quux/dc.dtd</spec>
        <spec type="relaxng">http://x.com/dublincCore.rng</spec>
      </schema>
      <schema name="Text Encoding Initiative" tag="tei"><!-- ... --></schema>
      <schema name="Some Random Schema" tag="x-abc"><!-- ... --></schema>
    </recordFormat>

    <!-- MIME-types of the kinds of object that can be returned -->
```

```
      <recordFormat type="text/plain"><!-- ... --></recordFormat>
      <recordFormat type="application/x-pdf"/>
      <recordFormat type="audio/mp3"/>
    </support>

    <statistics>
      <statistic type="meanSearchTime">1.56</statistic>
      <statistic type="medianSearchTime">0.67</statistic>
    </statistics>
  </peer>
```

Chapter 5. Discussion of the Description Record

The description record is discussed in detail here in order to elucidate the meanings of the elements and attributes that comprise it. In particular, where certain attributes may take only a small number of well-known values, there are enumerated and discussed.

The description record has a top-level element `<peer>`. This, like all the other elements and all attributes, is in the namespace `http://alvis.info/peer/`.

We now discuss the attributes of the top-level `<peer>` element, then the four sections within the record: `<addresses>`, `<subjectAreas>`, `<support>` and `<statistics>`.

Namespace, Version, Name and Identifier

The top-level `<peer>` element of the peer-description record carries four attributes, all of them mandatory:

`namespace`

The constant namespace URI `http://alvis.info/peer/`. This namespace qualifies all the elements and attributes of the record, indicating that they are the particular elements and attributes described in this document. It is an error to omit this attribute from a peer-description record, or to specify a different namespace URI.

`version`

A two-faceted number of the form `major.minor` indicating the version of the peer-description record specification that a record adheres to. At the time of writing, the version number is 1.0 - in other words, this document describes version 1.0 of the peer-description format.

This attribute is specified so that Alvis peers that understand a recent version of the peer-description format may recognise and adapt to peer-description records conforming to earlier versions of the format. Changes to the format that merely add new information will be indicated by incrementing the minor facet of version number (e.g. from `1.0` to `1.1`): such changes are easily handled by well-behaved readers, since all version `major.minor1` peer-description documents are also version `major.minor2` documents for all values of `minor2` greater than `minor1`. If it is ever necessary to make incompatible changes, such as removing or renaming elements or changing the record structure, this will be indicated by incrementing the major facet of the version number (e.g. from `1.4` to `2.0`).

An alternative approach, sometimes used in XML-based data-representation formats, would have been to embed a version-number in the namespace, like so: `http://alvis.info/peer/1.0/`. This approach suffers from the problem that the same-named elements in documents using different namespaces - even if those namespaces differ only in the minor facet of the version numbers - are actually different XML elements. This seemingly pedantic point has significant practical ramifications when such records have to be handled by inflexible software libraries such as SOAP toolkits. Implementation experience clearly indicates that the approach we have taken here, separating namespace and version, is more easily implementable and leads to better forward- and backward-compatibility.

`name`

This attribute holds a human-readable string to be displayed as the name of peer (in applications that show individual peers to users at all). Examples might include “Library of Congress Catalogue”, “University of Portsmouth Research

Papers", "Pipedreaming Music Soundtrack Demos" and "Mike Taylor's Totally Objective Music Reviews".

`id`

This is an opaque cookie that uniquely identitifies the peer. It is not intended for humans to see, but only for computers to use in determining peer identity.

We envisage no single co-ordinating server on an Alvis network to hand out these identifiers, since such a server would constitute a single point of failure, an unacceptable constraint on a robust, distributed system. Accordingly, the responsibility rests with individual peers and their maintainers to choose identifiers that are, in the immortal words of RFC 1341[1] "as unique as possible". One possible mechanism for generating such globally unique IDs is for the peer maintainer to use an Internet domain-name that they own, and append a locally unique token. Another is simply to generate a long string of random bits.

Addresses

In general, a single peer may be accessible on more than one address, and they may be of different types. For example, a peer hosted on a machine with a dynamically allocated IP address might have as its preferred peer address a direct TCP/IP connection to a listening socket, but such a peer address will become stale when its host is allocated a new IP address. It might, therefore, also be addressable by email to a static post-office domain. Connections made by means of the email address will furnish connecting peers with a peer-description record; after reading this, they might elect to switch to the new TCP/IP-connection peer address.

Each of the peer's addresses is indicated by an element of the form

```
<address type="xyz" bandwidth="n">addr</address>
```

in which the interpretation of `addr` is dependent on `type`.

The following `types` are recognised:

`tcp`

A direct TCP/IP connection to an open port. The content of the `<addr>` element consists of two components separated by a colon (`:`) - first an Internet host name or IP address, and second a port number. Possible examples include "192.168.0.106:12368", "alvis.info:9999" and "alvis.pipedreaming.org:9876". There is no default port-number, so the second element may not be omitted: "192.168.0.106" would be illegal.

`mail`

An email address to which suitably encoded Alvis network messages may be sent. For example, "alvis@miketaylor.org.uk", "ajasghd@alvis.info". The details of email transport are out of scope for this specification and will be documented elsewhere.

Additional address-types may be added in the future.

It is legal for a single peer to have addresses of multiple types, and to have multiple addresses of the same type.

Subject Areas

In general we expect Alvis peers to discover each others' coverage of subject areas by adaptive learning from the results of sending queries and receiving responses. However, a peer may "prime the pumps" by indicating *a priori* that it has good support for queries in particular subject areas. This is done by providing one or more elements of the formL

```
<subject>word-or-phrase</subject>
```

Each such element contains a word or phrase that may be used in successful queries against that peer.

Note that nothing is said about the inclusion of these candidate query terms in any ontology. This element is only an indication of the likelihood of getting good results when searching the peer for particular terms.

Support for Elements of Alvis Functionality

This is the most important part of the description records, carrying as it does information about the kinds of requests that can be made of the peer:

- We intend the Alvis architecture to support multiple incompatible query languages: therefore, each peer must specify which query-types it supports; and for some query-types, additional information is required.
- Similarly, different ranking algorithms may be supported, and peers need to advertise which algorithms they support so others can request the use of supported algorithms in their requests.
- Peers may return records using different metadata subsets: unique identifier only, Dublin Core summary, full text, etc.
- Finally, different peers are in general able to return records of different kinds - XML, PDF, MP3, etc., and to return records conforming to different schemas.

We will now consider each of these dimensions separately.

Support for Query Types

Different Alvis peers will in general be built on top of different existing databases using a variety of database toolkits that support a variety of types of query. Accordingly, those peers will support different query types, which they need to describe using `<query>` elements within the `<support>` portion of the record:

```
<query type="abc">
  <!-- ... -->
</address>
```

Some types of query require further specification as to what subsets of all possible functionality they support. For example, in structured queries, individual query terms may be submitted and specific indexes such as "author" , "title" and "date". Provision is made in the description record for peers to indicate such details.

The precise set of query types to be supported in Alvis has not yet been firmly established. For now, we consider three types, to be represented by `<query>` elements with `type` attributes taking the values `cql`, `bag` and `xpath`, as discussed below.

Support for CQL Queries

CQL[2] is an abstract language for expressing structured queries in a rigorous and powerful yet human-readable syntax. It is used by some Z39.50[3] servers and in all SRW/SRU[4] implementations.

Support for CQL queries is indicated by a `<query type="cql">` element, containing zero or more `<searchFormat>` and `<index>` elements further specifying the degree of CQL support.

The `<searchFormat>` element is common to all query-types, and is described below.

Each `<index>` element contains the name of a CQL index-name that is supported, such as `author`, `title` or `geographicName`. The optional `set` attribute may contain the URI of a sepcific CQL context set (analogous to an XML namespace) to which the specified index belongs. For example, `http://zing.z3950.org/cql/bath/2.0/` is the URI of the Bath Profile context set, version 2.0.

Support for "Bag-of-Words" Queries

Support for the standard information-retrieval "bag of words" query-type may be indicated by a `<query type="bag">` element. In this model, queries such as `sauropod cartilage stress biomechanics` are treated as short documents, to be associated with relevant target documents by similarity measures.

Zero or more `<searchFormat>` elements may be provided within the `<query type="bag">` element: see below.

Support for XPath Queries

XPath[5] is a language for selecting portions of XML records. In Alvis, an XPath query consists of an XPath specification together with a term; the query is satisfied by XML documents in which the portion specified by the XPath contains the indicated term.

Support for XPath queries is indicated by a `<query type="xpath">` element, containing zero or more `<searchFormat>` elements as described below. It is not clear that this type of query can be usefully used against records of formats other than XML.

Subsequent versions of the Alvis peer-description format may include means for peers to indicate which parts of the XPath specification they support.

Specifications Shared by Multiple Query Types

`<query>` elements of any `type` may contain zero or more `<searchFormat>` specifications. Each `<searchFormat>` element carries a `type` element, the value of which must be an IANA-registered MIME-type as listed at http://www.iana.org/assignments/media-types/. Examples include `text/plain`, `text/xml`, `audio/mpeg` (for MP3 files, see RFC 3003[7]), etc. The element indicates that queries of the appropriate type can find documents of the specified type.

Support for Ranking Schemes

Each ranking scheme supported by a peer can be indicated by a `<rank>` element with a `type` element which may have the value `field` or `algorithm`.

Rankings of type `field` are implented by sorting target documents on the fields whose names are specified in the content of the `<rank>` element. Such field-names are separated by commas, and listed in descending order of frequency; the

meaning of field-names is dependent on the record-format. For example, `<rank type="field">author,date</rank>` indicates the ability to sort of the value of the author field, with records having the same author sorted by date.

Rankings of type `algorithm` are taken from a list to be agreed on. Additional elements may be added to the peer-description record to allow the specification of parameters to algorithms.

Additional ranking types may be supported in subsequent versions of the peer-description format. Means of combining rankings may be introduced. This area of the specification is fluid due to the lack of implementation experience so far.

Support for Metadata Subsets

In search responses, target records may be returned from peers in various levels of completeness; that is, representing different subsets of the data and metadata. Search requests may indicate the level of completeness they require, and peer-description records can indicate the supported levels. This is done with a `<subset type="name"/>` element, in which the `type` may take the following values:

`id`

Only the identifier of the target record is returned. This is a short, opaque string that uniquely identifies the record within the peer providing the response. This identifier may subsequently be fed back to the same peer in a request for more of the record - typically, but not necessarily the full text.

`dc`

An XML record containing Dublin Core metadata (author, title, etc.) for the target record, typically including the record's identifier.

`xpath`

An XML record containing those parts of the full target record satisfying a specified XPath expression. This only makes sense for XML target records.

`fulltext`

The full text of the target record is returned.

Support for Record Formats

A given peer may support record in multiple formats (XML, plain text, email messages, PDF, etc.) Some of this information is communicated in the `<searchFormat>` elements of the query-support part of a peer-description record, but more detail may be exposed in a `<recordFormat>` section, including the schemas supported for each format and the ways in which those schemas can be expressed.

Each supported format is described by a `<recordFormat>` element:

```
<recordFormat type="mime/type">
  <!-- ... -->
</recordFormat>
```

This element has a mandatory `type` attribute, which must be an IANA-registered MIME-type[8].

<recordFormat> may contain zero or more <schema> elements, each of which describes an abstract schema (i.e. not necessarily an XML Schema in the W3C sense) to which some target records conform:

```
<schema name="Dublin Core" tag="dc">
  <!-- ... -->
</schema>
```

The name attribute contains a human-readable name, and the tag attribute contains a short opaque string which other peers can use when requesting records to be returned in this schema.

<schema> may contain zero or more <spec> elements, each of which describes a concrete schema using some specific constraint syntax or a human-readable description:

```
<spec type="xmlschema">http://foo.com/dc.xsd</spec>
```

The mandatory <type> element must take one of the values enumerated in the following list, and the content is a URI indicating the location of a schema specification of the appropriate type.

Possible <spec> types include:

`xmlschema`

: The specification is a W3C Schema[9] formally describing XML documents.

`dtd`

: The specification is a XML DTD[10] formally describing XML documents.

`relaxng`

: The specification is a Relax NG[11] specification formally describing XML documents.

`prose`

: The specification is simply human-readable prose describing the schema (much as this document describes the peer-description schema). Unlike the first three specification types, this is suitable for describing non-XML schemas as well as XML.

Statistics

This area provides a place for peers to communicate various statistics about themselves, such as their mean and median search times, number of records held, degree of connectivity to other peers, etc.

Each such statistic is expressed as a <statistic> element with a type element that contains the name of the particular statistic whose value is the content of the element.

No specific interpretation is here placed on any particular statistic.

Notes

1. http://www.ietf.org/rfc/rfc1341.txt
2. http://zing.z3950.org/cql/
3. http://lcweb.loc.gov/z3950/agency/

4. http://lcweb.loc.gov/srw/
5. http://www.w3.org/TR/xpath
6. http://www.iana.org/assignments/media-types/
7. http://www.ietf.org/rfc/rfc3003.txt
8. http://www.iana.org/assignments/media-types/
9. http://www.w3.org/XML/Schema
10. http://www.w3.org/TR/REC-xml/#sec-prolog-dtd
11. http://www.relaxng.org/

Chapter 6. DTD for Description Records

```
<!-- $Id: peer-description.dtd,v 1.6 2004/06/15 16:20:06 mike Exp $ -->

<!-- This DTD prescribes the format of Alvis peer-description records -->


<!ELEMENT peer (addresses?, subjectAreas?, support?, statistics?)>
<!-- "name" is human-readable, "id" is "as unique as possible" -->
<!ATTLIST peer xmlns CDATA #FIXED "http://alvis.info/peer/"
               version CDATA #REQUIRED
               name CDATA #REQUIRED
               id CDATA #REQUIRED>


<!-- A peer in general has multiple addresses -->
<!ELEMENT addresses (address*)>

<!ELEMENT address (#PCDATA)>
<!-- Address types include "tcp", "mail", etc.  Bandwidth is in kbps -->
<!ATTLIST address type CDATA #REQUIRED
                  bandwidth CDATA #IMPLIED>


<!ELEMENT subjectAreas (subject*)>

<!ELEMENT subject EMPTY>
<!ATTLIST subject type CDATA #REQUIRED>


<!ELEMENT support (query*, rank*, subset*, recordFormat*)>


<!-- Query-types include "cql", "bag" (of terms), "xpath", etc -->
<!ELEMENT query (searchFormat*, index*)>
<!ATTLIST query type CDATA #REQUIRED>

<!ELEMENT searchFormat EMPTY>
<!-- searchFormat types are MIME-types such as "text/xml" -->
<!ATTLIST searchFormat type CDATA #REQUIRED>

<!-- The "set" attribute is a context-set URI when used for CQL queries -->
<!ELEMENT index (#PCDATA)>
<!ATTLIST index set CDATA #IMPLIED>


<!-- The content of rank depends on its type -->
<!ELEMENT rank (#PCDATA)>
<!ATTLIST rank type CDATA #REQUIRED>


<!-- e.g. "id" to return record identifiers, DC for Dublin Core summaries -->
<!ELEMENT subset EMPTY>
<!ATTLIST subset type CDATA #REQUIRED>


<!-- recordFormat types are MIME-types such as "text/xml" -->
<!ELEMENT recordFormat (schema*)>
<!ATTLIST recordFormat type CDATA #REQUIRED>

<!ELEMENT schema (spec*)>
<!-- "name" is a human-readable string, "tag" is known to applications -->
<!ATTLIST schema name CDATA #REQUIRED
                 tag CDATA #REQUIRED>

<!-- Each schema may be specified in multiple ways, e.g. as a DTD -->
```

```
<!ELEMENT spec (#PCDATA)>
<!-- Schema types include "dtd", "xsd" (XML Schema), "rng" (Relax NG) -->
<!ATTLIST spec type CDATA #REQUIRED>


<!ELEMENT statistics (statistic*)>


<!ELEMENT statistic (#PCDATA)>
<!ATTLIST statistic type CDATA #REQUIRED>
```