Remote Search Standards in Future Library Applications

Report to Danish Agency for Culture

Prepared by

Poul Henrik Jørgensen

Portia

PHJ (at) Portia.dk

24. March 2013

Introduction

This report titled *Remote Search Standards in Future Library Applications* was produced for the *Danish Agency for Culture*[1] (DAC). The report was written by Poul Henrik Jørgensen from *Portia I/S*[2] *PHJ(at)Portia.dk* and the work was carried out in cooperation with Leif Andresen from DAC *LEA(at)Kulturstyrelsen.dk*.

On 24. January 2013 a draft version was presented to a workgroup established by the DAC in order to investigate IT-standards for future library search applications in Denmark. The present final version dated 2013-03-24 incorporates comments and feedback from this meeting. Besides various minor changes, a description of the US Library of Congress *Bibliographic Framework Transition Initiative* (BIBFRAME) has been added.

DAC has accepted this final version, which will be distributed to relevant parties and presented to the Danish Library standards Group (danZIG) during the year 2013.

Poul Henrik Jørgensen, Nivå 24. March 2013.

Contents

1 Background

The Danish national advisory committee for library standards (*danZIG*), decided at a meeting on 19 January 2012[3] to establish two groups to investigate a replacement for the *NISO Z39.50* standard (aka ISO 23950) between library applications in Denmark.

Z39.50 was primarily designed for bibliographic search, but is also used in connection with Interlibrary Loans (ILL) and other library transactions. The aim of one group is therefore to investigate search, while the other group deals with ILL.

During 2012 the Search workgroup has identified a number of common search scenarios. Members of the danZIG committee have also identified a number of standards, that are potentially relevant for future search applications at Danish libraries.

The Danish Agency for Culture has commissioned this report as supplemental input to the Search workgroup and to an open discussion about the future of remote search. The report was prepared by Poul Henrik Jørgensen from the consulting firm Portia and the work was carried out in cooperation with Leif Andresen from the Agency for Culture. Both have many years of experience from working with Z39.50 and other library IT standards. Among other things, Leif Andresen and Poul Henrik Jørgensen were also authors of the *danZIG Profile*, which has guided library applications in Denmark during the past decade[4].

2 Issues

The Z39.50 Search and Retrieval standard is still being used for searches between library systems in Denmark, but internationally Z39.50 is slowly losing market share even as a tool for bibliographic metadata searches. Z39.50 has never caught on outside of the library domain and its technical foundations are alien to current development environments.

Back in 2001 the *Z39.50 maintenance Agency*, which is hosted by the *US Library of Congress*, initiated an activity to develop a possible successor to Z39.50. The resulting SRU standard uses more familiar technical elements such as HTTP transmission and XML encoding. Conceptually SRU was intended to implement the same principles and abstract models as Z39.50 – but by means of more modern technologies.

Within the library world, SRU seems to be spreading slowly as a possible alternative to old school Z39.50, but is has not gained much traction elsewhere. In February 2013 the OASIS organization announced the publication of *searchRetrieve Version 1.0* OASIS Standard[5], which includes SRU along with the more widespread *OpenSearch* alternative.

More or less self appointed bodies such as ISO, NISO, W3C and OASIS could previously lend considerable weight to their standards solely by nature of their semi-official status. But today open or proprietary standards put forward by dominant systems vendors or service providers can rapidly attract a considerable following. It is therefore often necessary for developers to support different alternative official or proprietary standards with a low common denominator.

Specialized vendors or institutions with operational Z39.50 solutions may not see any pressing need to replace Z39.50 from an economical or purely technical point of view. But libraries are subject to other challenges, that necessitates a change of things:

New information providers, electronic media, mobile units and alternative business models are now offering attractive alternatives to the libraries venerable electronic catalogues.

In order to remain relevant as general information sources, it is imperative for libraries to provide search facilities that are easy to use and integrate with current tools and applications – also for developers without archaic special know how.

Emerging new metadata models based on dynamically linked concepts (e.g. RDF[6], BIBFRAME[7]) represent another challenge to traditional bibliographic search standards: Traditional standards were designed to search indexed sets of textual attributes gleaned from consolidated metadata records (e.g. danMARC2[8]); but alternative solutions may be required to retrieve relevant information via logical relationships and links between dispersed abstract objects.

3 Objectives

The objective of this work is to describe a long term strategy for modernization of the technical communications standards supported by Danish library search services. Different search scenarios may require different solutions.

The proposed strategy must facilitate a gradual phasing out of Z39.50 for bibliographic searches, but must also safeguard existing functions during the transition period. Relevant parties must be given reasonable time to implement the recommended new technology. But the transition should be accomplished before Z39.50 knowhow and tools becomes too scarce or impractical.

The proposed strategy should describe possible solutions in relation to different types of remote search; not only to traditional library applications but also to future *Linking Open Data* (LOD)[9]. Suggested general communications standards should be identified; but not described in detail.

The strategy must designate open non-proprietary standards with significant international support in order to facilitate international cooperation and to ensure a reasonable selection of potential vendors and products.

The strategy should also address national search services operated by the Danish Library Centre (DBC) or other contractors.

4 Scope

This work is primarily concerned with search to- and between libraries in Denmark:

- Search between traditional Integrated Library Systems (ILS)
- Other library transactions transmitted via Z39.50 (e.g. ILL)
- Search between extended library systems (e.g. data repositories)
- Search from external parties to both traditional and extended library systems

The strategy must address technical communications standards for external search services. The content of these services is beyond the scope of this work.

Specific transactions, which are exchanged between the systems, are not investigated as part of this work. Internal processes and functions are also beyond the scope of the present work.

Existing library systems are not treated specifically; except in the case of national search services operated by DBC or other contractors.

The Z39.50 standard is also used for other library applications besides bibliographic search; e.g. for exchange of Interlibrary Loan (ILL) transactions. This type of Z39.50 usage is considered in relation to the migration from Z39.50 to other standards.

The investigation should evaluate the following general open communications standards as possible conduits for the designated types of searches.

- OASIS searchRetrieve[10]
- Google Custom Search API[11]
- OASIS OpenSearch[12] (née Amazon OpenSearch)
- OASIS Open Data Protocol (OData)[13]
- NISO Open Discovery Initiative[14]
- DBC Open Search[15]

5 Standards

The following general search standards and initiatives are investigated with a view to their possible future role in the Danish national library infrastructure for search:

- ISO 23950 (Z39.50)
- OASIS Search/Retrieval via URL (SRU)
- OASIS OpenSearch (nee Amazon OpenSearch)
- OASIS Open Data Protocol (OData)
- DBC Open Search
- Google Custom Search
- OASIS searchRetrieve
- NISO Open Discovery Initiative
- LC Bibliographic Framework Transition Initiative BIBFAME[16]

5.1 ISO 23950 Information Retrieval (Z39.50)

The *ISO 23950 Information Retrieval* standard is better known by its ANSI/NISO designation *Z39.50 Information retrieval Protocol*[17]. Z39.50 was developed during the period from the 1970s through 2002 by an informal group of specialists (ZIG) under the auspices of the US Library of Congress.

Z39.50 provides an abstract query mechanism that supports searches based on a traditional index search model, i.e. nested attribute-operator-value triples. The *Z39.50* standard also includes an *Explain* service that provides information about the capabilities of a *Z39.50* server.

The response to a Z39.50 search request usually includes a reference to a result set, which identifies a set of objects. Actual records are subsequently retrieved via separate function calls with reference to a given result set.

Clients may request the designated records to be returned in different formats, e.g. *MARC*[18].

5.1.1 Z39.50 Development

Development work on the Z39.50 standard itself all but ended a decade ago, but many integrated library systems (ILS) including open source systems such as *Koha* and *Evergreen* continue to provide Z30.50 facilities.

Z39.50 may appear challenging and obscure to current developers because of the unfamiliar pre-web technology used. But there are a number of tried and tested software toolkits, which can shield aspiring Z39.50 implementers from the more inscrutable parts of the Z39.50 protocol.

5.1.2 Z39.50 Applications

Z39.50 was primarily designed for remote search and retrieval of bibliographic metadata records from electronic library catalogue. Many libraries still use Z39.50 for this purpose, e.g. to fetch *MARC* records for copy cataloguing or to check the holdings of other libraries. End-user client applications for Z39.50 have always been very rare and have now all but disappeared.

One Z39.50 directory[19] lists more than 2.000 *Z39.50* servers including 193 in the Danish domain, although the actual number of operational systems may be higher or lower.

Z39.50 plays an important role between libraries in Denmark; but not primarily for bibliographic catalogue search and retrieval. Libraries in Denmark chiefly use the Z39.50 protocol to check local library holdings status[20] and to exchange Inter Library Loan (ILL) transactions[21] between the national interlibrary loan service (*DanBib*[22]) and local holding libraries. Some public library systems in Denmark use Z39.50 between branch libraries, but this is considered an internal application in relation to this report.

The Z39.50 protocol is rarely used outside the library domain and there are several widespread modern alternatives. Future library search applications are therefore unlikely to implement and use the Z39.50 standard, cf. this quote from 2002:

"Whilst Z39.50 is used by the majority of library systems, its complexity has resulted in a lack of mainstream implementations outside of the library sector."[23]

5.2 OASIS SRU Search/Retrieval via URL

Back in 2001 the *Z39.50 Maintenance Agency* at the US *Library of Congress (LOC)* convened a group of active Z39.50 experts (including this writer) to design a new search and retrieval standard. The objective of this initiative called *Z39.50 International Next Generation (ZING)* [24]was to preserve the relevant parts of the Z39.50 abstract model, but to implement it by means of web standards:

- *HTTP GET/POST* functions
- *URI* parameter encoding
- Text based query language

- *XML* based record syntax
- *SOAP* transaction protocol

The result of this activity evolved into a set of related standards called *Search Retrieval via URL (SRU)*[25].

The SRU protocol comes in various flavors with similar semantics but different transport mechanisms:

- *SRU VIA HTTP GET*: URI[26] encoded requests and XML encoded responses.
- *SRU VIA HTTP POST*: HTML form-urlencoded[27] requests and same XML responses
- *SRU VIA HTTP SOAP (aka SRW)*: Requests and responses are carried within XML *SOAP*[28] messages.

5.2.1 CQL Contextual Query Language

SRU uses a query language called *Contextual Query Language (CQL)*[29]. The expressive power of *CQL* is comparable to that of the Z39.50 default query with nested sets of attribute-operator-value tuples.

Here is an example query string adapted from the CQL specification:

```
dc.title any fish or (dc.creator any sanderson and dc.identifier = "id:1234567")
```

CQL search terms can be URI encoded as part of an SRU request. Here is an example CQL query adapted from the SRU specification:

```
http://z3950.loc.gov:7090/voyager?version=1.1&operation=searchRetrieve&query=din
osaur&maximumRecords=1&recordSchema=dc
```

The CQL query language is not closely tied to the SRU protocol and may be used together with other standards. The widespread *Amazon OpenSearch* standard for example does not prescribe any specific query language and some systems use CQL as the query language for OpenSearch requests.

5.2.2 SRU Responses

The response to an SRU search request is a proprietary structure encoded as XML. Returned records may for example use different MARC/XML versions or other syntax types.

Here is a SRU response example adapted from the SRU introduction[30]:

```
<?xml version="1.0"?>
<zs:searchRetrieveResponse xmlns:zs="http://www.loc.gov/zing/srw/">
<zs:version>1.1</zs:version>
<zs:numberOfRecords>2380</zs:numberOfRecords><zs:records><zs:record>
<zs:recordSchema>info:srw/schema/1/dc-v1.1</zs:recordSchema>
<zs:recordPacking>xml</zs:recordPacking>
<zs:recordData>
<srw_dc:dc xmlns:srw_dc="info:srw/schema/1/dc-schema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://purl.org/dc/elements/1.1/"
xsi:schemaLocation="info:srw/schema/1/dc-schema
http://www.loc.gov/standards/sru/resources/dc-schema.xsd">
  <title>3-D dinosaur adventure.</title>
  <creator>Knowledge Adventure, Inc.</creator>
  <creator>Copyright Collection (Library of Congress) DLC</creator>
  <type>software, multimedia</type>
  <type>Educational games.</type>
  <type>Video games.</type>
  <publisher>Glendale, CA : Knowledge Adventure,</publisher>
  <date>c1995.</date>
  <language>eng</language>
  <description>Employs a dinosaur theme-park setting to introduce users to
Triassic, Jurassic, and Cretaceous periods. Features hypertext dinosaur
encyclopedia covering 150 million years of paleontology. Includes animated video
simulations, three-dimensional dinosaur museum, narration, games, activities,
and color illustrations.</description>
  <description>Source used: Copyright catalog online.</description>
  <subject>Dinosaurs--Juvenile software.</subject>
  <identifier>URN:ISBN:1569972133</identifier>
</srw_dc:dc></zs:recordData><zs:recordPosition>1</zs:recordPosition></zs:record>
</zs:record>
</zs:searchRetrieveResponse>
```

5.2.3 SRU Explain

SRU provides an *Explain*[31] operation to describe the capabilities of a given SRU server. SRU Explain identifies the location (URL) of a SRU service and lists its search indexes (search attributes) and presentation formats.

Here is a SRU Explain example adapted from the SRU Specification:

```
<sru:explainResponse xmlns:sru="http://www.loc.gov/zing/srw/">
  <sru:version>1.1</sru:version>
  <sru:record>
    <sru:recordPacking>XML</sru:recordPacking>
    <sru:recordSchema>http://explain.z3950.org/dtd/2.1/</sru:recordSchema>
    <sru:recordData>
      <zr:explain xmlns:zr="http://explain.z3950.org/dtd/2.1/">
        <zr:serverInfo protocol="SRU" version="1.2" transport="http" method="GET
POST SOAP">
          <zr:host>myserver.com</zr:host>
          <zr:port>80</zr:port>
          <zr:database>cgi/mysru</zr:database>
        </zr:serverInfo>
        <zr:databaseInfo>
          <title lang="en" primary="true">SRU Test Database</title>
        </zr:databaseInfo>
        <zr:indexInfo>
          <zr:set name="dc" identifier="info:srw/cql-context-set/1/dc-v1.1"/>
          <zr:index>
            <zr:map>
              <zr:name set="dc">title</zr:name>
            </zr:map>
          </zr:index>
        </zr:indexInfo>
        <zr:schemaInfo>
          <zr:schema name="dc" identifier="info:srw/schema/1/dc-v1.1">
            <zr:title>Simple Dublin Core</zr:title>
          </zr:schema>
        </zr:schemaInfo>
        <zr:configInfo>
          <zr:default type="numberOfRecords">1</zr:default>
          <zr:setting type="maximumRecords">50</zr:setting>
          <zr:supports type="proximity"/>
        </zr:configInfo>
      </zr:explain>
    </sru:recordData>
  </sru:record>
</sru:explainResponse>
```

5.2.4 SRU Development

Development of the *Z39.50* standard has been driven by the venerable Z39.50 Maintenance Agency at the Library of Congress together with a small but dedicated group of specialists, including veterans from the Z39.50 days.

The SRU standards together with the *Amazon OpenSearch*[32] standards have recently been incorporated under the umbrella of the *OASIS Search Web Services*[33] (SWS) framework. The *Oasis SWS Technical Committee* has published mappings from the SRU and OpenSearch standards to an abstract generic search-retrieve protocol.

Effective 30 January 2013 SRU version 1.2 and SRU version 2.0 together with CQL Version 1.0 and OASIS OpenSearch Version 1.0 are incorporated within the OASIS *searchRetrieve Version 1.0* standard[34]. The standard may therefore now informally be designated as *OASIS SRU*. Amazon/OASIS OpenSearch is not related to the *DBC Open Search* service.

The SRU protocols do not represent an insurmountable obstacle to implementers using modern development tools, for the reason that SRU utilizes well-known generic web technologies including HTTP, HTML, XML and SOAP. The most challenging SRU feature to implement is the somewhat over-engineered *CQL query* syntax. Some ostensible SRU servers do not support all of the more esoteric CQL features, but this is hardly noticed by most real world users.

There is a reasonable selection of specialized software toolkits, which can be used facilitate SRU implementations. Popular SRU tools include CQL parsers and Z39.50⇔SRU adapters, which can be retrofitted to legacy *Z39.50* Servers.

5.2.5 SRU Applications

Due to its Z39.50 origins, it is conceptually simple to map SRU operations and transactions to their respective Z39.50 counterparts. A number of pre-existing Z39.50 servers have therefore been outfitted with SRU front-ends that support SRU access as a supplement to Z39.50.

SRU Client development is not too difficult with the aid of current general development systems. Old school *Z39.50* on the other hand, is challenging to most developers except for a diminishing community of Z39.50 specialists.

The primary reason for using SRU gateways to existing Z39.50 servers is probably, that it is less painful to implement new SRU client applications than Z39.50 client applications.

SRU was envisioned as a technological update to the venerable Z39.50 standard, but has never gained much traction even within the traditionally insular library domain. This is probably because institutions with operational Z39.50 applications see few immediate benefits in a possible migration to the SRU protocol, which is even less well-known than Z39.50.

Instead of switching to SRU, it could be useful to adopt one of the prevalent alternatives such as the Amazon OpenSearch standards for basic search and retrieval applications.

Library systems in Denmark do not currently use the SRU protocol as part of the national infrastructure for library applications, although some local implementations are operational.

5.3 OASIS OpenSearch

The *Amazon OpenSearch*[35] standards were developed by an *Amazon Inc.* subsidiary called *A9* and presented by the *Amazon* founder Jeff Bezos during a conference in 2005.

This standard should perhaps rather be called ***OASIS** OpenSearch*, since the mapping from OpenSearch to the Oasis Search Web Service has been designated as the official OpenSearch specification[36].

The OASIS OpenSearch standards include a set of request parameters and a *description* document to describe the URI syntax, location and other facilities of a given OpenSearch server.

5.3.1 OpenSearch Requests

The *OASIS OpenSearch* standard uses HTTP to submit queries consisting of predefined elements plus the actual query string.

Here is an *OASIS OpenSearch* request example adapted from the *OCLC WorldCat Search API*[37] specification:

```
http://www.worldcat.org/webservices/catalog/search/opensearch?q=civil%20war&form
at=atom&start=6&count=5&cformat=mla&wskey=[key]
```

5.3.2 OpenSearch CQL Query Extension

The OASIS OpenSearch request format includes provisions for simple query string with keyword-value pairs, but it can also accommodate alternative query formats. There is a proposed *OASIS OpenSearch* extension to support all of the same CQL query features as the SRU protocol[38].

Here is an *OpenSearch* request with a *CQL* query. The example is adapted from the *Nature.com OpenSearch API* specification[39].

```
http://api.nature.com/content/opensearch/request?queryType=cql&query=cql.keyword
s+any+darwin+OR+cql.keywords+any+lamarck&api_key=<API key string here>
```

5.3.3 OpenSearch Results

OpenSearch results are (usually) returned as collections of records encoded in the widely used *ATOM Syndication Format* (aka *IETF RFC4287*)[40], which is also used by the OASIS OData standard.

The *ATOM* format is open ended, so that the returned collections may include optional custom elements, navigation links and relations by means of the usual namespace qualification technique.

Below is an *OpenSearch* response example (i.e. *ATOM* structure) adapted from the *OpenSearch* documentation.

The example also demonstrates the OpenSearch extension mechanism by embedding an *xhtml microformat*[41] (i.e. *vcard*) structure within the OpenSearch ATOM feed response:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/">
  <title>Example.com Phonebook Search</title>
  <entry>
    <title>Jane Smith</title>
    <link href="http://example.com/people/jsmith"/>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2003-12-13T18:30:02Z</updated>
    <summary type="xhtml">
      <div xmlns="http://www.w3.org/1999/xhtml" class="vcard">
        <div class="n">
          <span class="honorific-prefix">Ms.</span>
          <span class="given-name">Jane</span>
          <span class="family-name">Smith</span>
        </div>
        <div class="tel">
          <span class="type">Work</span>:
          <span class="value">(212) 555-0101</span>
        </div>
      </div>
    </summary>
  </entry>
</feed>
```

5.3.4 OpenSearch description document

An OpenSearch services is specified by a standard *OpenSearch description* document[42]. An OpenSearch description document describes the URL template syntax and optional parameters etc. supported by the designated OpenSearch server.

Client applications use the OpenSearch description document to construct the HTTP/URL requests to a given server.

Here is an example OpenSearch description document adapted from the OpenSearch specification:

```
<?xml version="1.0" encoding="UTF-8"?>
<OpenSearchDescription xmlns="http://a9.com/-/spec/opensearch/1.1/">
  <ShortName>Web Search</ShortName>
  <Description>Use Example.com to search the Web.</Description>
  <Tags>example web</Tags>
  <Contact>admin@example.com</Contact>
  <Url type="application/rss+xml"
template="http://example.com/?q={searchTerms}&amp;pw={startPage?}&amp;format=rss
"/>
</OpenSearchDescription>
```

5.3.5 OpenSearch Autodiscovery

OpenSearch services may include *ATOM links*[43] pointing to relevant OpenSearch description documents within search responses or web pages. This will enable compatible client applications to submit requests to the designated *OpenSearch* service.

Here is an example OpenSearch link adapted from the OpenSearch specification:

```
<link rel="search"
```

```
        href="http://example.com/opensearchdescription.xml"
        type="application/opensearchdescription+xml"
        title="Content Search" />
```

Many web browsers can use OpenSearch description documents to add the designated *OpenSearch* service to the browsers list of search providers.

The following *HTML* example adapted from the *Internet Explorer* documentation[44] implements a button, which the user can click to install a search provider. The designated *Provider.xml* file must contain the *OpenSearch description document*.

```
<INPUT TYPE="button" VALUE="Add Search Provider"
onClick='window.external.AddSearchProvider("http://www.example.com/Provider.xml"
);'>
```

5.3.6 OpenSearch Development

OASIS OpenSearch builds upon well known web standards including *HTTP, HTML, XML* and the *ATOM Syndication Format* which are supported out of the box by many prevalent development tools and applications from major software vendors.

There are also several specialized tools available to implement OpenSearch servers.

5.3.7 *OpenSearch* Applications

The OASIS OpenSearch standards are used by numerous information services outside of the library community[45].

OASIS OpenSearch client applications are ubiquitous. Many common web-browsers for example can understand *OpenSearch description documents* and use the information to communicate with *OpenSearch* providers.

Widespread server applications also support *OpensSearch* standards including for example MS *SharePoint, Wordpress, Google, Bing, Twitter and Amazon* etc.

The *OASIS Search Web Services* specification positions OpenSearch side by side with SRU via mappings to the OASIS searchRetrieve abstract model; which incidentally seems to descend from the ancestral Z39.50 model.

OASIS OpenSearch is not only conceptually similar to the OASIS SRU standard but is also more widespread than SRU outside of the library domain. OASIS OpenSearch and CQL could therefore represent a viable alternative to SRU and Z39.50 for traditional library search services.

Luckily, libraries do not necessarily have to choose between SRU or OpenSearch: It is possible to combine a SRU service with OpenSearch by means of the proposed OpenSearch SRU Extension. The OpenSearch SRU Extension references some parameters from the draft SRU Version 2.0, but these can just be ignored by earlier versions.

The *Nature.com*[46] search service for example offers SRU facilities via an OpenSearch interface.

5.4 OASIS OData Open Data Protocol

The *OASIS Open Data Protocol (OData)*[47] is a standard to support *Create, Read, Update and Delete (CRUD)* functions via a general web service interface.

OData was originally proposed by Microsoft, but is now being developed within the OASIS OData TC[48].

OData may be seen as a modern alternative to the venerable Microsoft *Open Database Connectivity ODBC*[49] standard from 1991 or the *SUN Systems Java Database Connectivity (JDBC)*[50].

ODBC provides a method to submit a text string containing an SQL statement to a relational database system and fetch the result set, but OData offers more than that:

- SQL was designed to handle relational data, i.e. rectangular two dimensional tables of data, where all of the rows (*records*) within a given table have the same set of columns (*fields*). SQL handles simple 1:N relationships well (e.g. parent-to-children), but N:N relations are tricky. The OData on the other hand is based on XML and HTTP and is better suited to handle heterogeneous hierarchical structures with arbitrary types of logical relations.
- SOAP WSDL can describe the record types and functions supported by a given service. OData provides comparable features in addition to descriptions of the logical relationships and navigation paths offered by a given database.
- ODBC utilize various proprietary transmission methods while OData uses a REST[51] architecture based on HTTP and XHTML. OData requests and responses can therefore be sent and presented via widespread web browsers and web applications.

OData is primarily based on the *IETF RFC 5023 Atom Publishing Protocol*[52] (*AtomPub*) which is used to exchange Create Read Update and Delete (CRUD) transactions. AtomPub is generally considered to be a canonical implementation of the REST architecture.

OData supplements the AtomPub standard with a comprehensive query facility and a general metadata tool to describe the resources, i.e. Entities and Relations provided by a given OData service.

5.4.1 OData Requests

OData uses the *AtomPub* standard to create, retrieve, update or delete designated entities by means of the standard *HTTP* verbs:

- GET: Fetch a *feed document* (collection) or an *entry document*.
- POST: Create an entry.
- PUT: Update an entry
- DELETE: Remove an entry

AtomPub (and by implication OData) exchanges objects serialized according to the *IETF RFC 4287 Atom Syndication Format*[53] which is also used by other standards such as OpenSearch.

Requests to an OData service are expressed in line with a recommended (but not prescribed) set of URI Conventions[54].

Here is a working example *OData* request adapted from the OData introduction. The example does the following:

1. Find *Product(1)*
2. Find the *Supplier* of the designated *Product*
3. Return all of the *Products* from the designated *Supplier*

http://services.odata.org/OData/OData.svc/Products(1)/Supplier/Products

5.4.2 OData Query Strings

OData also supplements the AtomPub standard by providing a query facility, which is equivalent to the SRU CQL query language[55].

Below is a working OData request with two query Query String Options[56]. The example, which is adapted from the OData description, returns the first five *Products* sorted by their *Name* property.

http://services.odata.org/OData/OData.svc/Products?$top=5&$orderby=Name

OData also provides the *$filter Query Option*, which supports Boolean expressions including functions.

The following example request including a $filter parameter is adapted from the OData description:

http://services.odata.org/Northwind/Northwind.svc/Customers?$filter=startswith(CompanyName, 'Alfr') eq true

OData query options are extensible and support the use of custom attributes and functions from other designated namespaces.

5.4.3 OData Service Document

An OData service is described by an *OData Service Document*[57] *(cf. IETF RFC5023 AtomPub),* which identifies the service and the top-level collections provided. The OData Service Document can be used by client applications to discover OData services.

Here is an example of the link to an OData demonstration service hosted by the *OData* website:

http://services.odata.org/OData/OData.svc/

This link will return the corresponding OData Service Document:

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<service xmlns="http://www.w3.org/2007/app"
xmlns:app="http://www.w3.org/2007/app" xmlns:atom="http://www.w3.org/2005/Atom"
xml:base="http://services.odata.org/OData/OData.svc/">
  <workspace>
    <atom:title>Default</atom:title>
    <collection href="Products">
      <atom:title>Products</atom:title>
    </collection>
    <collection href="Categories">
      <atom:title>Categories</atom:title>
    </collection>
    <collection href="Suppliers">
      <atom:title>Suppliers</atom:title>
    </collection>
  </workspace>
</service>
```

The above OData Service Document *i*dentifies three top-level collections called *Products*, *Categories*, and *Suppliers* respectively. Armed with this information, OData client applications can access the top level collections and from there navigate to related entities.

The basic *OData Service Document* may be supplemented by a corresponding *OData Service Metadata Document*.

OData Service Metadata Documents provide a comprehensive description of the data structures, relationships, navigation paths and functions provided by the designated *OData* service. This may be compared to the *W3C Web Services Description Language (WSDL)*[58] that primarily describes functions, data types and enumerations.

OData Service Metadata Documents are conventionally accessed via the service *$metadata* function as demonstrated below:

http://services.odata.org/OData/OData.svc/$metadata

Here is a snippet from the designated Service Metadata Document:

```
     <EntityType Name="Supplier">
        <Key>
          <PropertyRef Name="ID"/>
        </Key> <Property Name="ID" Nullable="false" Type="Edm.Int32"/>
        <Property Name="Name" Nullable="true" Type="Edm.String"
m:FC_KeepInContent="true" m:FC_ContentKind="text"
m:FC_TargetPath="SyndicationTitle"/>
        <Property Name="Address" Nullable="false" Type="ODataDemo.Address"/>
        <Property Name="Concurrency" Nullable="false" Type="Edm.Int32"
ConcurrencyMode="Fixed"/>
        <NavigationProperty Name="Products" ToRole="Product_Supplier"
FromRole="Supplier_Products"
Relationship="ODataDemo.Product_Supplier_Supplier_Products"/>
      </EntityType> -<ComplexType Name="Address">
        <Property Name="Street" Nullable="true" Type="Edm.String"/>
        <Property Name="City" Nullable="true" Type="Edm.String"/>
        <Property Name="State" Nullable="true" Type="Edm.String"/>
        <Property Name="ZipCode" Nullable="true" Type="Edm.String"/>
        <Property Name="Country" Nullable="true" Type="Edm.String"/>
```

5.4.4 OData Responses

The OData Service Document provides sufficient information, so that client applications can access the service and navigate to related entities via links in the responses.

Using the above service document one could for example construct the following URL to fetch the set of *Products*:

http://services.odata.org/OData/OData.svc/Products

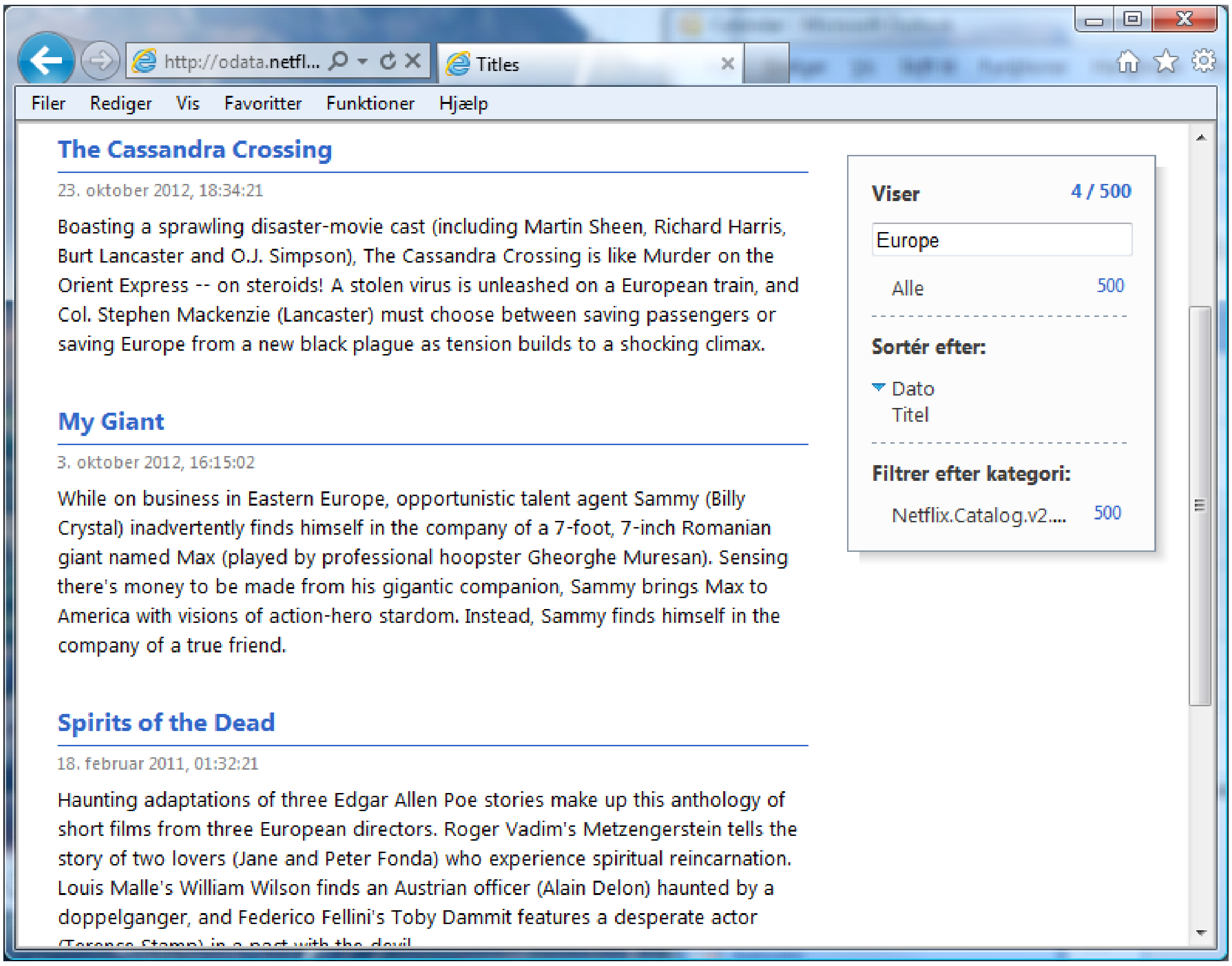
The response is the following *ATOM feed XML* document, which many web browsers can interpret out-of-the box.

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<feed xmlns="http://www.w3.org/2005/Atom"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
xml:base="http://services.odata.org/OData/OData.svc/">
  <title type="text">Products</title>
  <id>http://services.odata.org/OData/OData.svc/Products</id>
  <updated>2013-01-09T21:08:13Z</updated>
  <link title="Products" href="Products" rel="self"/>
  <entry>
    <id>http://services.odata.org/OData/OData.svc/Products(0)</id>
    <title type="text">Bread</title>
    <summary type="text">Whole grain bread</summary>
    <updated>2013-01-09T21:08:13Z</updated> -<author>
      <name/>
    </author> <link title="Product" href="Products(0)" rel="edit"/>
    <link title="Category" type="application/atom+xml;type=entry"
href="Products(0)/Category"
rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Category"/>
    <link title="Supplier" type="application/atom+xml;type=entry"
href="Products(0)/Supplier"
rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Supplier"/>
    <category
scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"
term="ODataDemo.Product"/>
    <content type="application/xml">
      <m:properties>
        <d:ID m:type="Edm.Int32">0</d:ID>
        <d:ReleaseDate m:type="Edm.DateTime">1992-01-01T00:00:00</d:ReleaseDate>
        <d:DiscontinuedDate m:type="Edm.DateTime" m:null="true"/>
        <d:Rating m:type="Edm.Int32">4</d:Rating>
        <d:Price m:type="Edm.Decimal">2.5</d:Price>
      </m:properties>
    </content>
  </entry>
```

Below is another OData example. This ATOM document was returned from the *Netflix* OData service in response to the following URL:

http://odata.netflix.com/v2/Catalog/Titles

Many web browsers including this one (IE 9 with Danish settings) can automatically handle OData (ATOM) documents as illustrated below. Notice, that the *titles* were dynamically filtered within the browser. Only four titles containing the term *Europe* are displayed out of the 500 titles:



The Cassandra Crossing

23. oktober 2012, 18:34:21

Boasting a sprawling disaster-movie cast (including Martin Sheen, Richard Harris, Burt Lancaster and O.J. Simpson), The Cassandra Crossing is like Murder on the Orient Express -- on steroids! A stolen virus is unleashed on a European train, and Col. Stephen Mackenzie (Lancaster) must choose between saving passengers or saving Europe from a new black plague as tension builds to a shocking climax.

My Giant

3. oktober 2012, 16:15:02

While on business in Eastern Europe, opportunistic talent agent Sammy (Billy Crystal) inadvertently finds himself in the company of a 7-foot, 7-inch Romanian giant named Max (played by professional hoopster Gheorghe Muresan). Sensing there's money to be made from his gigantic companion, Sammy brings Max to America with visions of action-hero stardom. Instead, Sammy finds himself in the company of a true friend.

Spirits of the Dead

18. februar 2011, 01:32:21

Haunting adaptations of three Edgar Allen Poe stories make up this anthology of short films from three European directors. Roger Vadim's Metzengerstein tells the story of two lovers (Jane and Peter Fonda) who experience spiritual reincarnation. Louis Malle's William Wilson finds an Austrian officer (Alain Delon) haunted by a doppelganger, and Federico Fellini's Toby Dammit features a desperate actor

5.4.5 OData Development

From a developer's point of view, OASIS OData is comparable to *SOAP*, although less well known.

Selected integrated development systems (e.g. MS *VisualStudio*) can automatically generate OData metadata service descriptions plus client and server interfaces, so that the protocol aspects and encoding of parameters etc. are transparent to the programmer. This is analogous to the handling of SOAP services and WSDL within modern development environments.

Many developers for example use OData applications in guise of Microsofts *WCF Data Services*, which automatically supports OData[59].

OData toolkits are freely available for most widespread system platforms. The OData website includes a directory of OData libraries for different environments[60].

5.4.6 OData Applications

The scope of the OData standard is conceptually different from traditional search-retrieve standards like OASIS OpenSearch and from function oriented remote procedure call standards such as SOAP.

Traditional search standards are designed to query indexes with textual attribute values, which point to corresponding metadata records (e.g. MARC). OData on the other hand, can also handle more complex searches based on logical links and navigation between abstract objects (e.g. BIBFRAME).

SOAP is primarily designed to implement custom remote procedure calls with composite parameters. *SOAP* is therefore suitable for a *Service Oriented Architecture (SOA)* with a relatively limited set of well defined functions; i.e. as traditional operational business applications.

An OData service is better suited to expose sets of linked resources and entities described by machine readable metadata. Client applications can create, read, update and delete resources via standard HTTP operations. OData is therefore applicable to *Resource-oriented Infrastructure* applications*;* e.g. data repositories with complex linked entities.

A few well known applications and public services provide OData interfaces[61], but most OData applications are probably private, just like most SOAP or ODBC applications are.

Many web browsers support the ATOM feed standard used by OASIS OData, and can therefore present structured information returned by OData services.

Several prevalent commercial applications support OData including *MS SharePoint 2010, SQL Server 2008, SAP NetWeaver, IBM WebSphere* and *Office 2010 Excel. (SAP inc*identally has published a succinct introduction to OData[62]).

OData could be especially useful for the emerging type of library applications that expose repositories of heterogeneous objects and media via linked atomic entities and relations; cf. the LC BIBFRAME.model.

OData can also implement custom remote procedure calls, but SOAP is still the preferred tool for this type of applications.

OData or other general Web API based alternatives may be particularly relevant to mobile applications, where the SOAP infrastructure might seem too bulky.

5.5 DBC Open Search

DBC Open Search[63] is a service provided by the Danish Library Center (DBC)[64]. DBC Open Search is not related to the homonymous Amazon/OASIS OpenSearch standards. Conceptually and technically the DBC Open Search service is actually more similar to the SRU standard – although the actual parameters and functions are different.

The DBC Open Search service can be used to search collections of bibliographic metadata hosted by DBC. The service can return metadata as well as full text data and relations for selected objects.

The DBC Open Search service provides two functions:

- *searchRequest*: Returns a subset of records specified by the *CQL* query string in the request parameters.
- *getObject*: Returns a record corresponding to the record-Id in the request parameters. The record is returned in the format requested.

DBC Open Search supports three types of communications methods:

- *HTTP GET* requests and responses with URI encoded requests and XML or JSON encoded responses
- *HTTP POST* requests with HTML FORMS encoded requests and XML or JSON encoded responses
- *SOAP* encoded requests and responses

The DBC Open Search service interface is specified by a WSDL service description[65].

Here is an example search request with a CQL query to the DBC Open Search service. The example is adapted from the DBC Open Search demonstration service.

```
<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://oss.dbc.dk/ns/opensearch">
  <SOAP-ENV:Body>
    <ns1:searchRequest>
      <ns1:query>dc.title=zorro AND dc.type=bog</ns1:query>
      <ns1:agency>100200</ns1:agency>
      <ns1:profile>test</ns1:profile>
      <ns1:start>1</ns1:start>
      <ns1:stepValue>10</ns1:stepValue>
    </ns1:searchRequest>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

DBC Open Search service uses the same Contextual Query Language (CQL)[66] as the SRU search standard.

The service can provide search responses in a variety of formats:

- DKABM[67] XML and JSON representations
- ISO 25577 MarcXchange[68] [69]
- DocBook[70]

DBC Open Search is part of the *DBC Open Library System* which is distributed under the *GNU Afferu General Public License*[71].

5.5.1 DBC Open Search Development

The interfaces provided by the DBC Open Search service are very similar to those offered by the SRU standard. The task of implementing a DBC Open Search client application is therefore similar to implementing a SRU client.

One thing which favors SRU in relation to DBC Open Search is the number of public articles, SRU toolkits and demonstration implementations, which can make life easier for SRU implementers.

The DBC Open Search service WSDL service description makes it relatively easy to implement corresponding *SOAP* client interfaces with the aid of modern integrated development systems.

Developers, who target mobile client devices or prefer to hand-code the interfaces, can use the alternative Web API provided by the DBC Open Search service.

5.5.2 DBC Open Search applications

The DBC Open Search service is used by systems and services offered by DBC, but there are not many third party tools or reference implementations.

The DBC Open Search service is relevant to its users, but others may see few compelling reasons to implement this standard instead of SRU or Amazon OpenSearch for example.

5.6 Google Custom Custom Search API

The *Google Custom Search API*[72] is an *OASIS OpenSearch* based interface to the *Google Custom Search Service*. The service can accept query requests and return results in JSON or the ATOM Syndication formats.

The *Google Custom Search Service* can index and search the static contents of a designated web site.

5.6.1 Google Custom Search Development

It is It is possible to implement an *OpenSearch* service for a given web site by means of the *Google Custom Search Service* with very little effort.

5.6.2 Google Custom Search Applications

The Google Custom Search Service is designed to search static content from websites, while most library data is provided via dynamic database interfaces etc.

Use of the Google Custom Search service is also subject to various restrictions, which may render it inappropriate for some library applications.

5.7 OASIS searchRetrieve

OASIS searchRetrieve is a general framework to specify search and retrieve standards.

OASIS has produced bindings for the *SRU* standard and the *OASIS OpenSearch* standard under the umbrella of searchRetrieve[73].

5.8 NISO Open Discovery Initiative

The *NISO Open Discovery Initiative (ODI)*[74] is not primarily concerned with search and retrieval standards.

The objective is to establish a standard set of practices for content discovery services as well as practices for the interaction between the service providers and the content providers.

5.9 BIBFRAME

The Library of Congress *Bibliographic Framework Transition Initiative* (BIBFRAME) was launched by the US Library of Congress in May 2011. The objective is to define a replacement for the MARC21 exchange format, which is better suited to decentralized web based linked metadata including bibliographic descriptions, authority data and holdings data for all types of library holdings.

The scope of the BIBFRAME initiative is analogous to that of MARC21: BIBFRAME specifies an abstract data model, vocabulary and a representation method, but not exchange protocols or searching functions.

The preliminary results of the BIBFRAME initiative were presented at the ALA Midwinter Meeting on January 27, 2013[75].

BIBFRAME is similar to the solution developed by the present author as part of the *VisualCat* cataloguing system back in 2001[76]:

- Bibliographic objects are represented by linked RDF structures and relations.
- Authority information is represented by linked atomic RDF structures and relations.
- The current representation format is based on the RDF/XM syntax.
- The bibliographic model is a simplified pragmatic version of the FRBR[77] concepts and relations with only two levels: *Works* (abstract entities) and *Instances* (physical or electronic manifestations).
- Related local or variable information (as opposed to immutable attributes such as author and date of publication) is represented by *Annotations*.

The BIBFRAME initiative has progressed rapidly and both the data model vocabulary as well as the technical implementation may be subject to change. But BIBFRAME addresses the pressing requirement for tools and standards designed to handle bibliographic registration of Linked Open Data (LOD).

5.9.1 BIBFRAME Development

Some of the published BIFRAME samples use an obsolete version of the W3C RDF/XML syntax[78]. Presumably an updated RDF syntax may be designated later.

The BIBFRAME initiative has spawned various tools and web services to demonstrate conversions between MARC21 and BIBFRAME RDF[79]. Here is a sample produced by the LC BIBFRAME demonstrator[80]:

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
  xmlns:ns3="http://id.loc.gov/vocabulary/identifiers/"
  xmlns:ns1="http://www.loc.gov/mads/rdf/v1#"
  xmlns:ns2="http://bibframe.org/vocab/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
>
  <ns2:Text rdf:about="http://bibframe.org/resources/sample-lc-1/16839103">
    <ns2:subject rdf:resource="http://bibframe.org/resources/sample-lc-
1/topic62"/>
    <ns1:authoritativeLabel>Dibdin, Thomas, 1771-1841. The Heart of Mid-Lothian
: a melo-dramatic romance in three acts</ns1:authoritativeLabel>
    <ns2:creator rdf:resource="http://bibframe.org/resources/sample-lc-
1/person23"/>
    <ns2:relatedWork rdf:resource="http://bibframe.org/resources/sample-lc-
1/work3"/>
```

```
    <ns3:lccn>2011563951</ns3:lccn>
    <ns2:language rdf:resource="http://id.loc.gov/vocabulary/languages/eng"/>
    <rdf:type rdf:resource="http://bibframe.org/vocab/Work"/>
    <ns2:derivedFrom rdf:resource="http://id.loc.gov/resources/bibs/16839103"/>
    <ns2:title>The Heart of Mid-Lothian : a melo-dramatic romance in three
acts</ns2:title>
    <ns2:instance rdf:resource="http://bibframe.org/resources/sample-lc-
1/instance17"/>
    <ns1:isMemberOfMADSScheme rdf:resource="http://id.loc.gov/resources/works"/>
    <ns2:generalNote>Includes songs; composer not named.</ns2:generalNote>
  </ns2:Text>
</rdf:RDF>
```

There are also numerous toolkits to convert and handle general RDF/XML data[81] in particular for Java environments.

Since the normative RDF syntax is based on XML, RDF structures including BIBFRAME data can be handled by general XML tools.

The W3C has designated a standard RDF query language called *SPARQL Query Language for RDF*[82] and an associated HTTP based protocol called (unsurprisingly) *SPARQL Protocol for RDF*[83].

Numerous other query languages have been developed to search RDF data, but SPARQL appears to be most widely supported.

SPARQL query language is derived from SQL and the well known attribute-relation-value triples with optional dynamic variable bindings.

Many specialized RDF database systems supports SPARQL, but the OASIS OData query language for example could probably also be used to convey RDF queries.

The same goes for the SPARQL protocol. SPARQL queries could be exchanged via the widespread OASIS OpenSearch protocol or OData instead of the relatively unknown SPARQL Protocol.

Although there are tools available, which can handle RDF/XML objects and SPARQL queries; these tools are generally not integrated with current mainstream development environments.

This author has therefore developed a prototype service to investigate search and retrieval of BIBFRAME objects and relations via a standard OData service where BIBFRAME objects and relations are encoded by the ATOPM Pub standard.

The only issue discovered so far is, that several types of BIBFRAME properties (e.g. *Associated Agent*) are specified to contain either a Literal or a reference to another type of object (e.g. *Agent*)[84]. An alternative approach could be to designate separate BIBFRAME properties for different types of data elements; e.g. *AgentLiteral* and *AgentLink*.

5.9.2 BIBFRAME Applications

The BIBFRAME specifications are still under development, but LC has processed over a million MARC records and OCLC Office of Research has converted all of WorldCat to BIBFRAME data[85].

BIFRAME is potentially important, not only because major bibliographic institutions have designated BIBFRAME as the replacement of MARC. BIBFRAME also important, because it is designed to handle emerging resource oriented bibliographic repositories with linked bibliographic objects and metadata, which are challenging to traditional bibliographic methods and standards such as Z39.50 and SRU.

Dynamic linked data models such as BIBFRAME may represent a challenge to bibliographic search and retrieval systems that consolidate information from distributed sources. But this is an internal implementation issue which is beyond the scope of Remote Search standards.

6 Strategy

A comprehensive strategy for search in future library systems should address the following issues:

- Z39.50 is considered obsolete and the technical basis is unfamiliar to many developers.
- Customers wish to protect past investments in existing Z39.50 applications.
- Vendors and customers are hesitant to invest in Z39.50 application developments.
- Library staff may use Z39.50 for bibliographic search, but end-users generally don't.
- The majority of Z39.50 traffic between library applications in Denmark is used to exchange ILL requests and other standardized transactions, which is not what Z39.50 was originally designed for.
- Emerging library services includes large repositories of heterogeneous linked objects and media, which are poorly served by traditional index search models
- Library services need to support the same mainstream standards as their prospective users
- The Bibliographic Transition Initiative (BIBFRAME) aims to replace MARC with linked RDF metadata structures and relations.
- Remote search and other bibliographic applications must deal with Linked Open Data and similar structures that are very difficult to handle with established solutions as Z39.50 and SRU.

The ultimate goal is to replace Z39.50 with other search standards sometime in the future. But a migration strategy should not unduly upset relevant existing applications during the transition.

6.1 Bibliographic search from library systems

Many integrated library systems (ILS) provide access to bibliographic metadata information via a Z39.50 Server interface.

Most library systems use proprietary standards and tools that are optimized to search and retrieve bibliographic data from their **internal** databases. Z39.50 Servers are therefore primarily utilized by **remote** library systems acting as Z39.50 Clients.

Library systems often use Z39.50 to search and retrieve catalogue data and holdings information from remote library systems.

The majority of Z39.50 searches between library systems do not utilize the advanced *Z39.50* structured search facilities however, but only simple keywords or record identifiers. From a functional point of view, man of the actual Z39.50 searches between library systems could therefore be handled by less complex standards such as OASIS OpenSearch.

The abstract data model behind *Z39.50* is designed to retrieve sets of unrelated objects (e.g. catalogue records), that are indexed by keywords, i.e. attribute-operator-value triples (e.g. *title*= "Madame Bovary"). This also corresponds to the implicit OASIS OpenSearch model.

OASIS OpenSearch Query elements come with a number of built in parameter names and operators that are used to compose search queries. The standard OpenSearch parameters can be combined with Z39.50 attributes or other attribute types designated by means of a namespace prefix mechanism.

Alternatively, it is also possible to extend an OpenSearch service with the full set of Z39.50 inspired search facilities represented by the OASIS SRU CQL query syntax.

Using OASIS OpenSearch with optional SRU Extensions for library-to-library bibliographic search may provide the following benefits:

- OASIS OpenSearch is widespread, while SRU has failed to gain any significant tracking - even within the library domain. Libraries and their ILS vendors can therefore leverage software tools and knowhow from the rest of the World.
- There is a wealth of public documentation, examples and tools available to OpenSearch developers.
- OpenSearch is less complex than SRU and presumably easier for software developers.
- Libraries can reuse OASIS OpenSearch facilities in relation to client applications outside of libraries.

CQL support might only be required by services for professional librarians; not by end-users.

OASIS SRU is descended from the Z39.50 model but uses different technologies. It is therefore conceptually simple although technically challenging to develop gateways from SRU to Z35.50 or vice versa. Fortunately this has been done by others.

In contrast, gateways between OASIS SRU and OASIS OpenSearch are easier, since both standards use the same technologies and OASIS has mapped both standards to the same formal model.

The transition from Z39.50 search to OASIS SRU/CQL and other traditional string-based index search languages is aided by an existing *Mapping to danZIG Profile from Danish Praxis rules for Search Codes*[86] published by the Danish Agency for Culture.

A transition strategy for bibliographic search applications would entail the age-old solution of using adapters to convert between different standards – while systems using the old standards are gradually retired:

- Install symmetrical Z39.50-SRU gateway(s)
  - This technique is used by some existing services.
  - Requires no changes to existing Z39.50 Servers.
- Implement symmetrical OpenSearch-SRU gateway(s).
  - Probably easy to implement and operate with limited overhead.

This approach would protect existing investments in legacy Z39.50 Servers and allow new applications to use the prevalent OASIS OpenSearch standards as an alternative to Z39.50 or OASIS SRU Clients and Servers without additional effort.

During a possibly drawn out transition period, old and new systems could co-exist and interoperate seamlessly via the following pathways.

1. Z39.50 Client applications talk to existing Z39.50 Servers directly.
2. Z39.50 Clients talk to OASIS SRU Servers via the Z39.50=>SRU gateway.
3. Z39.50 Clients talk to OASIS OpenSearch Servers via the Z39.50=>SRU and the SRU=>OpenSearch gateways
4. OASIS OpenSearch Clients talk to the OpenSearch Servers directly
5. OpenSearch Clients talk to SRU Servers via the OpenSearch=>SRU gateway.
6. OpenSearch Clients talk to Z39.50 Servers via the Open-Search=>SRU and SRU=>Z39.50 gateways.
7. SRU Clients talk to SRU Servers directly
8. SRU Clients talk to Z39.50 Servers via the SRU=>Z39.50 gateway.
9. SRU Clients talk to OpenSearch Servers via the OpenSearch=>SRU gateway.

The different logical paths are summarized in the following table:

No	Client	Step-1 GW	Step-2 GW	Server
1	Z39.50	-	-	Z39.50
2	Z39.50	Z39.50=>SRU	-	SRU
3	Z39.50	Z39.50=>SRU	SRU=>OpenSearch	OpenSearch
4	OpenSearch	-	-	OpenSearch
5	OpenSearch	OpenSearch=>SRU	-	SRU
6	OpenSearch	OpenSearch=>SRU	SRU=>Z39.50	Z39.50
7	SRU	-	-	SRU
8	SRU	SRU=>Z39.50	-	Z39.50
9	SRU	SRU=>OpenSearch	-	OpenSearch

6.2 Transactions between library applications

Different library systems routinely exchange information about holdings, inter library loans (ILL), user names and access rights etc. These types of information are usually exchanged by means of different proprietary methods.

Although Z30.50 was designed with metadata search and retrieval in mind, some library-to-library applications employ it as an expensive carrier mechanism for other types of data; e.g. Inter Library Loan transactions. Library applications in Denmark for example use Z39.50 extensively for interchange of holdings information and Interlibrary Loan transactions as specified by the Danish national *danZIG* Profile[87]

But metadata search and retrieval standards such as Z39.50, OpenSearch and SRU are not optimal to handle general client server applications.

Most types of interactions between library applications are basically not different from other types of transactions between systems outside of the library community. These transactions involve structured objects that are logically related to other objects; e.g. patrons, loans and returns or vendors, orders, bills and payments.

Furthermore; client-server applications often utilize remote function calls with object parameters in order to create, retrieve, update and delete objects and relationships. Many applications use the popular *SOAP* protocol for this task.

Microsoft and others are now promoting the OASIS Open Data Protocol (OData) as a more flexible and efficient alternative to SOAP and ODBC[88] etc., although it is not yet widely used for transaction processing applications.

OData could provide the following benefits to relevant applications:

- Support for structured strongly typed objects and linked data relationships.
- Support for methods (remote function calls) with parameters.
- Support for *CRUD* (Create, Read, Update, Delete) operations
- Comprehensive machine readable metadata Service description can be used for automatic code generation, cf. SOAP WSDL.
- Provides migration path for existing SOAP applications.
- OData is supported by widespread general applications including SAP[89] and SharePoint[90].

OData Server and Client interface code can be generated from OData Service Metadata Documents the same way as SOAP Server and Client interface code is generated from *SOAP WSDL* specifications.

Some of the special applications that use the Z39.50 protocol to exchange transactional data (e.g. ILL) utilize general Z39.30 features like diagnostic messages and special record formats. Furthermore, these types of applications are primarily used within the library community.

During a transition period it could therefore be better to replace Z39.50 as carrier protocol with SRW (i.e. SRU over SOAP) for the following reasons:

- SRU supports the same types of features and record formats as Z39.50. Much of the underlying application logic would therefore be unaffected by a change of carrier protocol.
- Systems vendors are generally familiar with using SOAP for remote procedure calls.
- The SRW service is specified by machine readable WSDL service specifications that facilitate development of compliant Client- and Server applications.

6.3 Data repository search from library systems

Libraries and other information aggregators are building virtual and physical repositories containing heterogeneous data objects and digital media.

Client applications need to search, retrieve and sometimes update objects from these repositories based on traditional metadata as well as contextual relationships. E.g. "*find sci-fi movies where any of the lead actors graduated from the same school as the producer*".

OData is suggested as the most suitable standard for this kind of **resource** oriented of applications, while e.g. SOAP is primarily designed for **function** oriented services. Unlike the dedicated search standards examined here, OData services can incorporate Entity-Relation (ER) models which describe the logical associations between local objects as well as external hypermedia links.

The ER models may serve as documentations (i.e. metadata), but can also be used for navigation and retrieval of objects. The logical relations can be used to navigate between linked entities. Something that is difficult with the aid of traditional indexed keyword searches.

OData query may be compared with the SQL language; but SQL was designed to handle relational data represented as tables of uniform rows. OData is better suited to a handle evolving repositories including heterogeneous hierarchical structures and linked objects, e.g. BIBFRAME data.

It is not necessary to convert all of the data within existing repositories in order to provide an OData service. Relevant logical subsets of object types and relational links may be described by the OData service documents and made available via a supplementary standard ATOM Pub compliant service interface.

This author has developed a prototype to demonstrate search and retrieval of BIBFRAME objects and relations via a standard OData / ATOM Pub service implemented on top of a traditional relational database; i.e. without alien RDF databases or RDF Query languages.

6.4 General search from non-library systems

In order to encourage the use of library resources, it is important to facilitate access from external client applications and end users.

While specialized ILS vendors may for example be familiar with Z39.50 tools and techniques, these are not widely known elsewhere. It is therefore important for library applications to provide services which support more popular standards and tools.

OASIS OpenSearch is an obvious choice for basic search and retrieval services, because this standard is reasonably familiar to relevant developers and is supported by a wide selection of client applications, tools, and implementation examples.

Google Custom Search API can provide a quick and easy method to establish an OASIS OpenSearch service with information from selected websites; but it comes with a price tag and other limitations.

OASIS SRU is conceptually and technically similar to OpenSearch, but has failed to make much of an impact outside the library domain. OASIS OpenSearch with the proposed SRU Extensions provides similar facilities as OASIS SRU. This makes the case for SRU even less compelling.

OASIS OData may be the best solution for more demanding end-user applications, updates and resource oriented data repositories involving heterogeneous interrelated objects.

7 Conclusions

The venerable Z39.50 standard is becoming obsolete and the designated successor OASIS SRU has failed to make much of an impact outside of the library world.

The ubiquitous OASIS OpenSearch is probably the most widespread standard for keyword based search and retrieval. OASIS OpenSearch is supported by web browsers and by many other popular services and applications. OASIS OpenSearch can provide the same type of facilities as OASIS SRU including support for the same *CQL* query language used by OASIS SRU. OASIS OpenSearch is therefore proposed as the preferred alternative to Z39.50 for traditional bibliographic search and retrieval.

Both OASIS SRU and OASIS OpenSearch are inspired by the REST architectural style, but SOAP is still the principal standard method for remote transaction processing. Existing applications that use the Z39.50 protocol to exchange transaction data such as ILL messages could replace Z39.50 by SRU over SOAP (SRW).

Emerging library applications include large repositories of linked heterogeneous objects and media with BIBFRAME metadata structures. Remote access and maintenance of these repositories is beyond the scope of traditional index based search standards. A possible solution could be to combine BIBFRAME with the OData standards, which supplement the widespread ATOM standards by a flexible query language and a general Entity-Relation metadata model.

---

[1] Danish Agency for Culture http://www.kulturstyrelsen.dk/english/
[2] Portia I/S http://www.portia.dk/websites/index.htm
[3] danZIG minutes 2012-01-19 http://biblstandard.dk/danzig/doc/danzig50-ref.htm
[4] danZIG Profile http://www.bs.dk/publikationer/andre/danzig/01/
[5] OASIS searchRetrieve http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/part1-apd/searchRetrieve-v1.0-os-part1-apd.html
[6] RDF Resource Description http://www.w3.org/TR/rdf-primer/
[7] BIBFRAME Model Overview http://bibframe.org/
[8] danMARC2 http://www.kat-format.dk/danMARC2/
[9] Linking Open Data LOD http://linkeddata.org/
[10] https://www.oasis-open.org/news/announcements/searchretrieve-version-1-0-published-as-a-committee-specification
[11] Google Custom Search API https://developers.google.com/custom-search/v1/overview
[12] OASIS OpenSearch http://www.opensearch.org/Home
[13] OASIS Open Data Protocol OData http://www.odata.org/
[14] NISO Open Discovery Initiative http://www.niso.org/workrooms/odi/
[15] DBC Open Search http://oss.dbc.dk/plone/services/open-search
[16] BIBFRAME http://www.loc.gov/marc/transition/
[17] NISO Z39.50 http://www.niso.org/standards/resources/Z39.50_Resources
[18] MARC http://www.loc.gov/marc/
[19] Z39.50 Target directory http://irspy.indexdata.com/
[20] danZIG Holdings retrieval http://www.bs.dk/publikationer/andre/danzig/01/html/chapter06.htm
[21] danZIG Ordering and ILL http://www.bs.dk/publikationer/andre/danzig/01/html/chapter07.htm
[22] DanBib http://www.danbib.dk/index.php?doc=english#webservices
[23] Z39.50 and XML - Bridging the old and the new http://www2002.org/CDROM/alternate/XS2/
[24] Z39.50 Primer http://www.niso.org/publications/press/Z3950_primer.pdf
[25] LOC SRU http://www.loc.gov/standards/sru/
[26] URI Syntax http://tools.ietf.org/html/rfc3986
[27] Form-urlencoded http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4.1
[28] W3C SOAP http://www.w3.org/TR/soap12-part1/
[29] CQL http://www.loc.gov/standards/sru/specs/cql.html
[30] LOC SRU Introduction http://www.loc.gov/standards/sru/simple.html
[31] SRU Explain http://www.loc.gov/standards/sru/specs/explain.html
[32] OASIS Open Search http://www.opensearch.org/Home

[33] OASIS Search Web Services https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=search-ws
[34] OASIS searchRetrieve Version 1.0 https://www.oasis-open.org/news/announcements/searchretrieve-version-1-0-oasis-standard-published
[35] OASIS OpenSearch standards http://www.opensearch.org/Home
[36] OASIS SWS Binding for OpenSearch http://docs.oasis-open.org/search-ws/v1.0/opensearch-v1.0.html
[37] OCLC WorldCat OpenSearch http://oclc.org/developer/documentation/worldcat-search-api/opensearch
[38] OASIS OpenSearch CQL Extension http://www.opensearch.org/Community/Proposal/Specifications/OpenSearch/Extensions/SRU/1.0/Draft_1
[39] Nature.com OpenSearch API http://developers.nature.com/docs/apis/OpenSearch_API
[40] ATOM Format http://tools.ietf.org/html/rfc4287
[41] Microformats http://microformats.org/
[42] OASIS OpenSearch description document http://www.opensearch.org/Specifications/OpenSearch/1.1#OpenSearch_description_document
[43] ATOM link http://tools.ietf.org/html/rfc4287#section-4.2.7
[44] AddSearchProvider OpenSearch http://msdn.microsoft.com/en-us/library/aa744112(v=vs.85).aspx
[45] OASIS OpenSearch providers http://www.opensearch.org/Community/OpenSearch_search_engine_directories
[46] OASIS OpensSearch and SRU Integration http://dlib.org/dlib/july10/hammond/07hammond.html
[47] OASIS Open Data Protocol OData http://www.odata.org/
[48] OASIS OData TC https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=odata
[49] ODBC Open Database Connectivity http://support.microsoft.com/kb/110093
[50] JDBC http://www.oracle.com/technetwork/java/javase/jdbc/index.html
[51] Representational State Transfer (REST) http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
[52] Atom Protocol AtopmPub http://www.ietf.org/rfc/rfc5023.txt
[53] Atom Syndication Format http://www.ietf.org/rfc/rfc4287.txt
[54] OData URI Conventions http://www.odata.org/documentation/uri-conventions
[55] CQL and OData query https://tools.oasis-open.org/issues/browse/ODATA-3?focusedCommentId=30910&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#action_30910
[56] OData filter query option http://www.odata.org/media/30002/OData.html#thefiltersystemqueryoption
[57] OData Service Documents http://www.odata.org/documentation/atom-format#ServiceDocuments
[58] Web Services Description Language WSDL http://www.w3.org/TR/wsdl
[59] WCF and OData http://msdn.microsoft.com/en-us/data/aa937697
[60] OData toolkit libraries http://www.odata.org/libraries
[61] OData applications http://www.odata.org/ecosystem
[62] OData in a Nutshell http://wiki.sdn.sap.com/wiki/download/attachments/233739132/02_OData_In_A_Nutshell.pdf?version=1&modificationDate=1315543843596
[63] DBC Open Search http://oss.dbc.dk/plone/services/open-search
[64] DBC http://www.dbc.dk/english
[65] DBC Open Search WSDL http://opensearch.addi.dk/2.2/opensearch.wsdl
[66] LOC CQL http://www.loc.gov/standards/sru/specs/cql.html
[67] DKABM http://biblstandard.dk/abm/doc/ABMindholdsogtransportformat_2011.pdf
[68] MarcXchange http://www.loc.gov/standards/iso25577/
[69] ISO 25577:2008 MarcXchange http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=43005
[70] DocBook http://www.loc.gov/standards/iso25577/
[71] GNU Affero General Public License http://www.gnu.org/licenses/agpl.html
[72] Google Custom Search API https://developers.google.com/custom-search/v1/overview .
[73] searchRetrieve Overview http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/cs01/part0-overview/searchRetrieve-v1.0-cs01-part0-overview.pdf
[74] NISO Open Discovery Initiative http://www.niso.org/workrooms/odi/
[75] BIBFRAME at ALA January 2013 http://www.loc.gov/marc/transition/presentations/index.html

[76] FRBR and RDF http://eprints.rclis.org/4114/1/jorgensen_eng.pdf
http://www.portia.dk/pubs/VisualCat/RomeER/FrbrRdfRomeNov2001.pdf
[77] FRBR http://www.ifla.org/publications/functional-requirements-for-bibliographic-records
[78] RDF/XML Syntax http://www.w3.org/TR/REC-rdf-syntax/
[79] BIBFRAME Tools http://bibframe.org/tools/
[80] BIBFRAME samples http://bibframe.org/resources/sample-lc-1/exhibit.html#./exhibit.html?&_suid=136179064063906588405045527381
[81] RDF Toolkits
http://www.w3.org/2001/sw/wiki/index.php?title=Special:Ask&offset=0&limit=500&q=%5B%5BCategory%3ATool%5D%5D+%5B%5BSW+Technology%3A%3ARDF%5D%5D&p=format%3Dul%2Ftemplate%3DToolDisplayLinkWithNameWithcategoryAndLanguage%2Flink%3Dnone%2Fcolumns%3D1&po=%3FTool+Name%3D%0A%3FCategory%3D%0A%3FProgramming+language%3D%0A
[82] SPARQL Query Language for RDF http://www.w3.org/TR/rdf-sparql-query/
[83] SPARQL Protocol for RDF http://www.w3.org/TR/rdf-sparql-protocol/
[84] BIBFRAME Associated Agent Property http://bibframe.org/vocab/associatedAgent.html
[85] BIBFRAME Early Experimenters Experience
http://www.loc.gov/marc/transition/pdf/ALAmw2013_bibframe_OCLC_publicUpdate_Fons.pdf
[86] Mapping to danZIG profile from Danish Search Codes http://biblstandard.dk/danzig/mapping.htm
[87] danZIG Profile Specification http://www.bs.dk/publikationer/andre/danzig/01/pdf/danZIG_Profile_Specification.pdf
[88] Open Database Connectivity (ODBC) http://support.microsoft.com/kb/110093
[89] SAP NetWeaver Gateway http://whealy.com/sap/Gateway/Technical%20Brief/index.html
[90] Sharepoint OData support http://technet.microsoft.com/en-us/library/fp161238(v=office.15)#section1