U.S. Federal Institute of Museum and Library Services
National Leadership Grant

Realizing the Vision of
Networked Access to Library Resources:
An Applied Research and Demonstration Project to Establish and Operate a Z39.50 Interoperability Testbed

Final Report for The Z39.50 Interoperability Testbed Project Phase 2:

Developing an Alternative Approach for Interoperability Testing of Library Z39.50 Servers

William E. Moen, Ph.D.
<wemoen@unt.edu>
Principal Investigator

School of Library and Information Sciences
Texas Center for Digital Knowledge
University of North Texas
Denton, TX 76203

Project Team
Jason Thomale
JungWon Yoon
School of Library and Information Sciences
Texas Center for Digital Knowledge

December 3, 2005

Table of Contents

The Z39.50 Interoperability Testbed Project, Phase 2: Developing an Alternative Approach for Interoperability Testing of Library Z39.50 Servers

1. Introduction

The U.S. Federal Institute of Museum and Library Services (IMLS) awarded a National Leadership Grant in 2000 to support the research and demonstration project, *Realizing the Vision of Networked Access to Library Resources: An Applied Research and Demonstration Project to Establish and Operate a Z39.50 Interoperability Testbed*. In our December 31, 2003 interim status report on the Z39.50 Interoperability Testbed Project (Z-Interop) <http://www.unt.edu/zinterop/Documents/InterimReport31Dec2003.pdf>, we highlighted the challenges of individual libraries to participate in the Z-Interop testbed and we suggested an alternative method for interoperability testing for Z39.50 servers that could accommodate the limitations of individual library systems. In May, 2004, IMLS awarded an extension to the Z-Interop Project for additional research to develop and test an alternative approach for interoperability testing and also awarded approximately $50,000 in additional funding to carry out the research. We refer to this extension to the original award as Z-Interop Phase 2 or Z-Interop2. Research during Z-Interop2 built on the conceptual and technical infrastructure developed during the Z-Interop Project <http://www.unt.edu/zinterop>.

The project team for Z-Interop2 consisted of:

- Principal Investigator, Dr. William E. Moen
- Research Assistants:
    - JungWon Yoon, Ph.D. student in the Interdisciplinary Information Science Ph.D. Program
    - Jason Thomale, Masters student in the School of Library and Information Sciences

In addition, the project used the services of the following consultants/contractors:

- Index Data, Copenhagen, Denmark (Sebastian Hammer and Mike Taylor)
- Penelope Benardino, former Masters student in the School of Library and Information Sciences.

The goal of Z-Interop2 was to develop an approach to interoperability testing using specially created diagnostic metadata records and automatic testing scripts to conduct interoperability testing in the context of Z39.50 servers and online library catalogs. From June 2004 through August 2005, the Principal Investigator, contractors, and staff produced a proof of concept of this alternative approach for interoperability testing. This document serves as a final report for the Z-Interop2 Project.

2. The Problem Addressed by the Z-Interop2 Project

During the first phase of the Z-Interop Project, we discovered that most libraries only have a production system for their integrated library system and online catalog; they did not have a testing environment where they could load the Z-Interop test dataset of 400,000 MARC records, index those records separately from the production bibliographic database, and have that available for interoperability testing. As we noted in our December 31, 2003 interim report:

> The Principal Investigator had assumed at the time of submitting the proposal that individual libraries would be interested in submitting their Z39.50 server implementations to interoperability testing. This assumption proved incorrect for two primary reasons. First, the size of the test dataset (over 400,000 records) was too large to be accommodated by individual libraries. Second, most individual libraries' Z39.50 implementations did not include a test environment in which the test dataset could be loaded and indexed separately from the production bibliographic database of the individual library. This made it impossible for most libraries to participate in the testbed. As part of a project the Principal

> Investigator carried out in 2002 for the Illinois State Library, these barriers to participation became clear. The Illinois State Library encouraged the Illinois Regional Library Systems that hosted large shared bibliographic systems to go through the interoperability testbed to improve statewide resource discovery and sharing. Yet, even with encouragement and support by the Illinois State Library, the Regionals were not able to go through the testbed for the reasons listed above.

The approach of the original Z-Interop testbed environment worked well for the vendors; they were able to set up a separate implementation of their Z39.50 server and online catalog products, prepare it for testing, and have the Z-Interop Project conduct interoperability testing.

Z-Interop2 was intended to address this situation. In that same interim report, we suggested an alternative method for interoperability testing for Z39.50 servers that could accommodate these limitations of individual library systems. The idea for this approach came from Sebastian Hammer, a principal in Index Data, which is a company specializing in Z39.50 and networked information retrieval. The alternative method proposed would a small set of very special MARC records (we refer to these as "radioactive MARC records," explained below) that can serve as diagnostic mechanisms for assessing system functionality, performance, and interoperability. At the time of the interim report, we believed that this alternative approach would have potential for providing interoperability testing services to individual libraries. In addition, we believed that this approach could be adaptable to other protocol and metadata contexts beyond Z39.50 and MARC.

This alternative approach for interoperability testing does not invalidate the work accomplished in the first phase of the Z-Interop Project. All of the ground-breaking work in data analysis, testing procedures, test searches, and identification of continuing interoperability challenges informed our work on this alternative approach for interoperability testing.

We believed that a small set of very special MARC records can serve as diagnostic mechanisms for assessing system functionality, performance, and interoperability. The metaphor of a "radioactive MARC record" is based on current medical diagnostic techniques for people. When a person has a particular medical condition, there may be two approaches for diagnosis. One could be considered invasive, where the person would undergo a surgical technique for physically examination of the problematic area or anomaly. The other approach could be considered less invasive, where the patient is injected with a dye, possibly radioactive, and once it has spread throughout the body, scanning techniques allow a medical professional to identify structural or mechanical problems or anomalies.

A radioactive MARC record approach for interoperability testing is less "invasive" for an individual library. It does not require loading a large test dataset such as used in the original Z-Interop testbed. Nor does it require a separate testing environment on the local implementation. Instead, the library loads these special MARC records into its production online catalog system, and the Z-Interop staff conducts a series of tests to assess system functionality, performance, and interoperability. The radioactive MARC records are legitimate instances of MARC records that a library system can import and process, and then remove when the testing is completed. These records, however, have very special characteristics. We discuss the details of these records in a subsequent section.

3. Z-Interop2 Project Target Activities

In our request for an extension on this project to explore an alternative approach to interoperability testing, we identified a number of separate activities to be undertaken. More detailed descriptions about these activities follow in subsequent sections and the status of each of these activities is discussed. Briefly, the activities we proposed were:

- Develop a Framework for Interoperability Testing
- Create the Radioactive MARC Records
- Develop Testing Procedures and Scripts
- Further Development of Indexing Guidelines

- Verify Procedures through the Z-Interop Reference Implementation
- Testing with Individual Libraries

Associated with these activities were a set of proposed project deliverables:

- Framework for interoperability testing using radioactive MARC records
- A set of very specialized MARC diagnostic (i.e., radioactive) records; initial estimate is approximately 20 records; these will be publicly and freely available
- Documented and verified interoperability testing procedures and automatic processes for testing and analysis; any resulting software will be made available as open source software under the GPL
- Expanded guidelines for indexing policies for online catalogs
- Interoperability testing with 3-5 individual libraries.

These activities were in line with the overarching goal of the original Z-Interop Project, namely, ***improve Z39.50 semantic interoperability among libraries for information access and resource sharing.***

Upon award of the extension in May 2005, the PI developed a preliminary work plan to guide all project activities. Appendix A contains a copy of the work plan.

4. Research and Development in the Z-Interop2 Project

The following sections describe in more detail the research and development carried out in the Z-Interop2 Project.

4.1. Developing a Framework for Interoperability Testing

One of the first steps in planning an alternative approach for interoperability testing was to develop conceptually the components and functions necessary for the testing. In June and July 2004, we developed a preliminary framework for testing that outlined the necessary components. Figure 1 presents a graphical view of the framework.

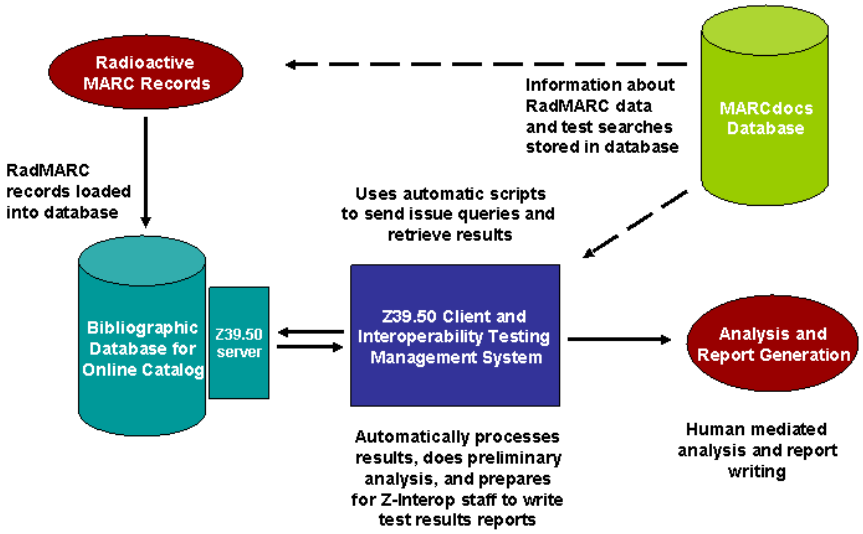


Figure 1. Interoperability Testing Framework

Figure 2 presents the testing framework for the original Z-Interop testbed. An objective for the Z-Interop2 Project was to explore how more of the interoperability testing activities could be automated, and if the level of manual effort for testing and analysis could be reduced.

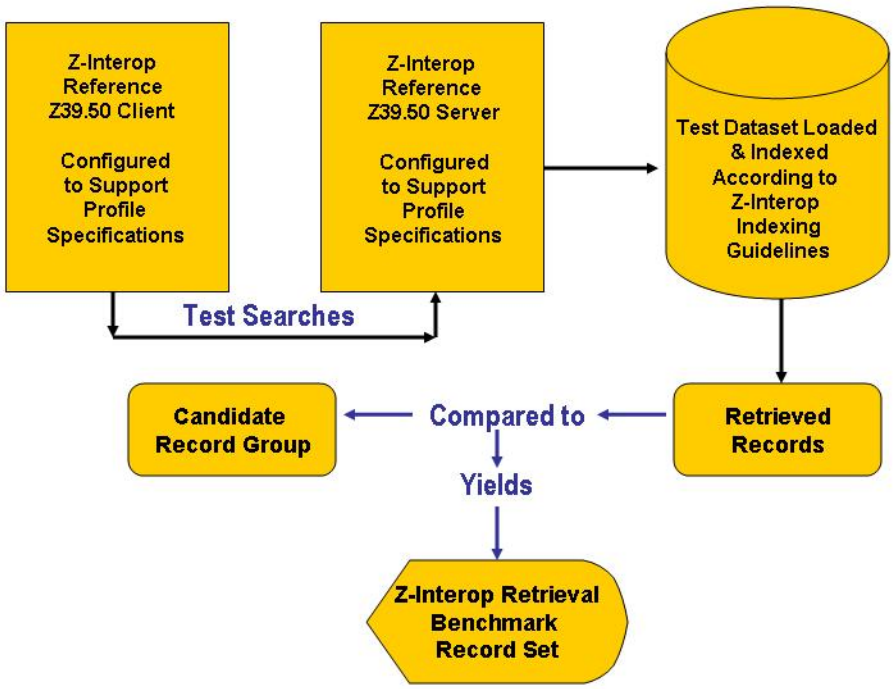


Figure 1. Original Z-Interop Testing Framework

The primary components for the Z-Interop2 framework identified in Figure 1 are:

- The Radioactive MARC Records: The specially designed diagnostic MARC records
- The Z39.50 Client and Interoperability Testing Management System: This includes the set of test searches, and the automatic testing script that issues searches, retrieves records, and develops reports on the search and retrieval results
- The MARCdocs Database: A database of MARC documentation that enables the automatic identification of types of searches to issue.

Appendix B contains the preliminary document that describes the testing framework in more detail.

4.2. The Question Space for Z-Interop2 Interoperability Testing

During the deliberations related to the conceptual framework for interoperability testing, the PI responded to a request from the Index Data contractors to specify the “questions” we would use interoperability testing to answer. In the course of a research project, new ideas for methodologies often surface, and the discussions about the “question space” helped to more clearly identify the characteristics of and requirements for the Radioactive MARC Records, and for the types of test searches that would be required.

Discussions resulted in an understanding that we could ask questions at four different levels, each with a different focus. In part, this reflected our understanding of the threats to interoperability that the PI had identified during the original Z-Interop Project, which are:

In the first phase of the Z-Interop testbed, the PI anticipated several levels at which interoperability needs to occur, and we identified some of the threats to such interoperability. The research in that project confirmed the reality of these threats. The PI (Moen, 2001b) reported on the diverse factors diverse factors that can affect interoperability in networked information retrieval applications, including:

- Multiple and disparate operating and Information retrieval systems

- Multiple protocols
- Multiple metadata schemes
- Multiple data formats
- Multiple languages and character sets
- Multiple vocabularies, ontologies, and disciplines.

In the context of the Z-Interop Project, we identified key factors threatening interoperability:

- Differences in implementation of the standard
- Differences in local information retrieval systems

In the latter case, this includes search functionality available in the system, indexing policies affecting the access points in the database, word extraction and processing choices, and character set and character encoding and normalization. As a way to indicate the scope of our investigations, the PI (Moen, 2001a) identified the levels of interoperability of concern in the Z39.50 context:

- Low-level protocol (syntactic): Do Z-client and Z-servers interchange protocol messages according to the standard?
- High-level protocol (functional): Do Z-client and Z-servers support appropriate Z39.50 information retrieval services for user tasks?
- Semantic level: Can Z-clients and Z-servers and local information retrieval systems preserve and act on meaning of information retrieval tasks?
- User Task level: Do systems support information retrieval tasks of one or more user groups?

Within the context of our previous investigations and the maturity of Z39.50, the syntactic level is of little concern. The development of the Bath Profile: An International Z39.50 Specification for Library Applications and Resource Discovery (2004), and the U.S. National Z39.50 Profile for Library Applications (National Information Standards Organization, 2003) addressed many issues related to the functional level. The original Z-Interop testbed was successful in part because the profiles defined expected Z-client and Z-server behaviors and interactions. The biggest challenge to reliable interoperability appears at the semantic level. Semantic interoperability here is not addressing the concerns of two words meaning the same thing or other problems related to linguistics and meaning. Instead, semantic interoperability concerns the ability of two systems to present and process user information tasks in a way that meanings of those tasks are retained. For example, if a user does a title search for information resources, the search is actually executed on a search target against words from titles in the record. A common sense idea, yet often search targets do not process searches as the user intended (e.g., processing an exact match search for a title as a set of keywords combined using Boolean operators and matching the words not only in title access points but in other access points as well).

With this in mind, we refined the Z-Interop2 question space to address the following levels, and we indicate for each level, the types of questions that can be asked and answered through interoperability testing.

> **Profile conformance level**: This level addresses the interoperability between the Z-client and Z-server. Assessing this level of interoperability relies up the use of a Z39.50 profile that identifies Z39.50 specifications for search and retrieval. Questions that can be addressed at this level include:
>
> - Does the Z-server process each query successfully.
> - If the Z-serve cannot process the query as sent, does it send the appropriate diagnostic message?
>
> For example, when the Z-client issues a query, it sends a Z39.50 attribute combination such as “1”, “1003”, and a search term. The Z-server should execute the query and either (a) identifies that 0 or more records in the database match the query criteria, or (c) returns an appropriate diagnostic (e.g., a bib-1 diagnostic number that might mean “database records do not contain data associated with

access point."). Return of some diagnostics may indicate that a lack of functionality available in the information retrieval system.

Information retrieval system level: This level addresses the capability of the information retrieval (IR) system underlying the online catalog application. The questions addressed at this level are captured in the following:

- What search functionality does the information retrieval system have?

For example, if the Z-client issues a Boolean search or a keyword with right truncation search, does the IR system have the capability to process such a search?

Bibliographic record level: This level is also an IR system oriented level, but its focus is on how the IR system indexes fields in the bibliographic record to provide access points or searchable components of the record. The searches defined in the profile assume that certain access points are provided, yet because of the richness of the MARC record, many fields may be selected for indexing to populate the index for a particular access point. Therefore, the questions address by this level take the following form:

- Does the information retrieval system index the appropriate fields in the records for specific access points?
- Do the system's indexing policies support searches for the searches defined in the Z39.50 profile?

Indexing policies on the IR system has established which MARC fields/subfields are indexed (e.g., words or phrases are taken from the specific fields and placed in an index). To assist in this testing, we assume that the fields that should be indexed are those defined in the document from the first phase Z-Interop Project, "Indexing Guidelines to Support Z39.50 Profile Searches" (Moen, 2002). This means that to do the testing for this level of questions, the RadMARC records will need to contain data values in all the fields/subfields identified for indexing per the above document. This will enable us to determine whether all such fields/subfields are being indexed by the target system.

Our approach to interoperability testing may allow us to confirm if certain fields are not indexed for particular access points. We may or may not be able to know whether unexpected fields are indexed for these access points.

Data content level: This level addresses a more difficult arena, but an arena that may have affect interoperability. Here we are addressing how the IR system processes the data content of the records, such as questions related to normalization of the data, dealing with hyphenated works, and special characters and diacritics. This level was not addressed in the Z-Interop2 Project.

The question space is also informed by two Z39.50 profiles:

- ANSO/NISO Z39.89, The U.S. National Z39.50 Profile for Library Applications (National Information Standards Organization, 2003) <http://www.niso.org/standards/resources/Z39_89final.pdf>
- Bath Profile: An International Z39.50 Specification for Library Applications and Resource Discovery, Release 2.0 (The Bath Group, 2004)

These specifications provide well-defined searches and expected client and server behaviors at several conformance levels. For initial interoperability testing, we used the profile-defined searches listed Table 1. These searches also pointed to the data necessary in the RadMARC records to support the testing using these searches.

4.3. The Radioactive MARC Record Structure

At the heart of the Z-Interop2 Project was the concept of a Radioactive MARC (RadMARC) Record, a specially designed diagnostic record that would contain specific data values that enabled the interoperability testing. This section provides the details on these records and is based on the document contained in Appendix C, which contains more discussion of the specifications for the RadMARC records.

As noted before, two Z390.50 profiles provide the specifications that are the basis for the test searches. Searches defined at Level 0 and Level 1 require appropriate RadMARC records for the test searches. In addition, different types of records (as indicated in the MARC Leader/06) are needed since systems may index MARC fields/subfields differently depending on the type of record. Finally, the RadMARC records need to be clearly identified as to the type of searching they are intended to assess, and therefore some version information about each record is included in the record.

MARC records can describe different types and formats of bibliographic materials. MARC Leader/06 indicates the type of record (i.e., the information object type being described by the MARC record). The *Anglo-American Cataloguing Rules* (AACR), the standard for descriptive cataloging, specifies rules for describing different types of materials. MARC Leader/07 indicates the bibliographic level of the record. Table 2 summarizes the values of the MARC Leader/06, Leader/07, and the AACR categories of materials to show the complexity of coding and labeling for what the MARC records describe.
The coding of Type of Record in MARC Leader/06 is not aligned directly with the 10 format types of information objects as addressed by AACR (in the third column of the table above). In some cases, two code values in the Leader/06 are addressed by the same AACR format of material. Additionally, the Leader/06 doesn't indicate if the material is a serial publication (or Continuing Resource). For this, the Leader/07 – Bibliographic Level indicates serial with a value of s. To summarize, the types of records or materials for which RadMARC records were created address those in the table.

The specially constructed MARC records for this approach to interoperability testing are the foundation, and the design of these records was a key intellectual challenge. The fundamental data unit in the RadMARC records is a token. A token is a string of characters that has a specific structure and semantics that will serve as "words" or other data values in specific fields/subfields. A field/subfield may have a sequence of tokens. The specially designed tokens populate selected field/subfields in the RadMARC records. Several sets of RadMARC records are used in interoperability testing. The sets are distinguished by the amount of content designation populated in the records (see discussion below). All selected content designation use the special tokens. The following is the structure of content-rich tokens being used in the RadMARC records:

- A single alpha character for left-hand padding.
    - Value = r
- A single alpha character to indicate the format of the material being described or type of record
    - Value = Selected values as defined in MARC Leader/06 – Type of Record or the Leader/07 – Bibliographic Level
- Three numbers indicating the Field Tag
    - Value = Defined in MARC 21 specifications
- A single integer to indicate number of occurrence the Field Tag
    - Value = Sequential number starting with 1
- A single alpha character to indicate the Subfield Code
    - Value = Defined in MARC 21 specifications
- A single integer indicating the offset within subfield
    - Value = Use the following scheme: 1=first token in subfield, 2=second token in subfield; 3= third token in subfield, etc.
- A single alpha character for right-hand padding
    - Value = r

An example token that shows this structure is **ra2451a1r**, which can be parsed as:

- r - Left-hand padding
- a - Type of record -- this is a books type record
- 245 - Field code
- 1 – First occurrence of field in record
- a - Subfield code
- 1 - Offset within subfield, where 1 = first token in subfield
- r - Right-hand padding

If there was a second instance of this field (in this example the 245), the token element for the second occurrence would be `rm2452a11r`. An example of a complete MARC 245 with a sequence of tokens in selected subfields follows:

```
245 $a rm2451a1r rm2451a2r rm2451a3r $b rm2451b1r rm2451b2r rm2451b3r $c
rm2451c1r rm2451c2r
```

In any given subfield, a sequence of tokens will not exceed three tokens. The limitation allows the appropriate interoperability testing of various phrase-oriented searches as defined in the profiles without adding undue complexity to the RadMARC records.

In addition to the field- and subfield-specific tokens, each RadMARC record contains additional information to uniquely identify the record, the version of the record, and other details about the source and purpose of the record (e.g., in the 001, 040, and the 583 fields). Figure 3 presents an example of a RadMARC record in human-readable form built according to the specifications.

001	UNTRadMARC001
040	$a ZinteropUNT
100	$a rm1001a1r, rm1001a2r, $d rm1001d1r.
245	$a rm2451a1r rm2451a2r rm2451a3r : $b rm2451b1r rm2451b2r rm2451b3r / $c rm2451c1r rm2451c2r rm2451c3r.
440	$a rm4401a1r rm4401a2r rm4401a3r
490	$a rm4901a1r rm4901a2r rm4901a3r
583	$a RadMARC $b www.unt.edu/zinterop/001 $d 1 $e ATS $i 1 $k JungWon Yoon $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E. Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 19, for the 19 most commonly occurring indexable author, title, and subject fields discovered in a separate analysis. This is the first version of this record.
600	$a rm6001a1r rm6001a2r, $d rm6001a1r.
650	$a rm6501a1r rm6501a2r rm6501a3r $x rm6501x1r $v rm6501v1r $z rm6501z1r.
651	$a rm6511a1r rm6511a2r $x rm6511x1r.
653	$a rm6531a1r rm6531a2r rm6531a3r
700	$a rm7001a1r rm7001a2r, $d rm7001d1r.
710	$a rm7101a1r rm7101a2r.

Figure 3. Sample RadMARC Record

The structure specified above allows us to create either very minimal or very large MARC records. We can populate any content designation available in the MARC record with tokens conforming to the

structure specified. The question addressed by the project team was what content designation needs to be populated for purposes of interoperability testing. This is discussed next.

4.4. Populating RadMARC Records with Tokens

In the discussion above about the RadMARC records, we indicated that we need to know what content designation in a MARC record (i.e., which fields/subfields in the record) to populate with tokens to support the interoperability testing of profile-defined searches. This decision is tied closely with a specific level of the question space for interoperability testbed, namely:

- **Bibliographic record level:** This level focuses on how the information retrieval system indexes fields in the bibliographic record to provide access points or searchable components of the record. Questions address by this level include:
    - Does the information retrieval system index the appropriate fields in the records for specific access points?
    - Do the system's indexing policies support searches for the searches defined in the Z39.50 profile?

As part of the original Z-Interop testbed, we identified more than 500 MARC fields/subfields in a MARC record that could be indexed to support author, title, and subject searching. The complete list of this content designation is contained in *Indexing Guidelines to Support Z39.50 Profile Searches* (Moen, 2002). In other analyses for that project, we analyzed occurrences of MARC content designation in the Z-Interop test dataset of more than 400,000 MARC records from OCLC's WorldCat database (Moen and Benardino, 2003). We also examined the occurrence of content designation that could be indexed to support author, title, and subject searches, and discovered that 19 of the more than 500 subfields that could be indexed accounted for 80% of all occurrences. Table 1 shows these 19 subfields.

Marc 21 Field	Subfield	Description	Index
650	a	Subject added entry Topical Term Subfield a = Topical term or geographic name as entry element	Subject
245	a	Title Statement Subfield a = Title	Title
245	c	Title Statement Subfield c = statement of responsibility	Author
650	x	Subject added entry Topical Term Subfield x = General subdivision	Subject
100	a	Main entry Personal Name Subfield a = personal name	Author
650	z	Subject added entry Topical Term Subfield z = Geographic subdivision	Subject
700	a	Added entry Personal Name Subfield a = personal name	Author
245	b	Title Statement Subfield b = Remainder of title	Title
100	d	Main entry Personal Name Subfield d = dates associated with a name	Author
651	x	Subject added entry Geographic Name Subfield x = General subdivision	Subject
651	a	Subject added entry Geographic Name Subfield a = Geographic name	Subject
650	v	Subject added entry Topical Term Subfield v = Form subdivision	Subject
700	d	Added entry Personal Name Subfield d = dates associated with a name	Author

600	a	Subject added entry Personal Name Subfield a = personal name	Subject
710	a	Added entry Corporate Name Subfield a = corporate name or jurisdiction name	Author
440	a	Series Statement Added Entry Title Subfield a = title	Title
490	a	Series Statement Subfield a = Series statement	Title
600	d	Subject added entry Personal Name Subfield d = dates associated with a name	Subject
653	a	Index Term Uncontrolled Subfield a = the term	Subject

Table 1. The 19 Most Frequently Used Author/Title/Subject MARC Indexable Fields/subfields

We identified several possible sets of RadMARC records based on the frequency count data from the Z-Interop Project and completed the creation of two sets if records. Record Set 1 uses the 19 most commonly occurring indexable fields for author, title, and subject-related. The RadMARC record in Figure 3 shows that these content designation structures have been populated.

Record Set 2 uses all author, title, and subject content designation that had frequency counts of 1,000 or more as based on the Z-Interop analyses discussed above.

However, we can extend the RadMARC records to include all possible content designation as listed in the Z-Interop *Indexing Guidelines* document. Before doing that, though, we are planning to create a third set of records based on two sources of information:

- The Network Development and MARC Standards Office (n.d.) recommendations for national level records
- The Program for Cooperative Cataloging (2003) core record standards.

For example, the recommendations for national level records identify “mandatory” and “mandatory if applicable” content designation. A comparison of those recommended content designation structures with the Z-Interop indexing guidelines indicate 131 fields/subfields that are author, title, and subject related. We can create RadMARC records using these 131 fields/subfields.

MARC records can describe different types and formats of bibliographic materials. MARC Leader/06 indicates the type of record (i.e., the information object type being described by the MARC record). The *Anglo-American Cataloguing Rules* (AACR), the standard for descriptive cataloging, specifies rules for describing different types of materials. MARC Leader/07 indicates the bibliographic level of the record. Table 2 shows the values of the MARC Leader/06, Leader/07, and the AACR categories of materials to show the complexity of coding and labeling for what the MARC records describe.

Leader/06 Code	Semantics	AACR Categories of Materials
a	Language material	Books, Pamphlets, and Printed Sheets
c	Notated music	Music (Notated and manuscript music)
d	Manuscript notated music	Music (Notated and manuscript music)
e	Cartographic material	Cartographic Materials
f	Manuscript cartographic material	Cartographic Materials
g	Projected medium	Motion pictures and video-recordings (including digital and non-digital)
i	Nonmusical sound recording	Sound Recordings (musical and non-musical)
j	Musical sound recording	Sound Recordings (musical and non-musical)

Leader/06 Code	Semantics	AACR Categories of Materials
k	Two-dimensional nonprojectable graphic	
m	Computer file	Electronic Resources
o	Kit	
p	Mixed material	Graphic materials (includes mixed materials, with or without archival control)
r	Three-dimensional artifact or naturally occurring object	Three Dimensional Artifacts and Realia
t	Manuscript language material	Manuscripts (including manuscript collections)
Leader/07 Code	**Semantics**	**AACR Categories of Materials**
a	Monographic component part	
b	Serial component part	
c	Collection	
d	Subunit	
i	Integrating resource	
m	Monograph/item	
s	Serial	Continuing Resources

Table 2. Formats and Types of Materials Described in MARC Records

The coding of Type of Record in MARC Leader/06 is not aligned directly with the 10 format types of information objects as addressed by AACR. In the proposal for extension to the Z-Interop Project, we proposed to deal with the formats of material as addressed by AACR. These are found in the third column of the table above. In some cases, two code values in the Leader/06 are addressed by the same AACR format of material. Additionally, the Leader/06 doesn't indicate if the material is a serial publication (or Continuing Resource). For this, the Leader/07 – Bibliographic Level indicates serial with a value of s.

To summarize, the types of records or materials for which RadMARC records were be created address those in Table 3, which also provides the values for the Leader/06 and Leader/07 as appropriate.

Category of Material Described by RadMARC record	Leader/06 Value	Leader/07 Value
Books, Pamphlets, and Printed Sheets	a	
Continuing Resources	a	S
Music (Notated and manuscript music)	c	
Cartographic Materials	e	
Motion pictures and video-recordings (including digital and non-digital)	g	
Sound Recordings (musical and non-musical)	j	
Electronic Resources	m	
Graphic materials (includes mixed materials, with or without archival control)	p	
Three Dimensional Artifacts and Realia	r	
Manuscripts (including manuscript collections)	t	

Table 3. Formats and Types of Materials Addressed in RadMARC Records

Record Set 1 comprises ten RadMARC records, with appropriate tokens representing the categories of materials listed in Table 3. Appendix D contains all ten of these RadMARC records in human-readable (as opposed to machine-readable) form.

For Record Set 2, we faced a new challenge. In a MARC record, one can have only one occurrence of a 100, 110, 111, and 130. Each of these fields occurred 1,000 or more times in our frequency count. Therefore, we made the decision for Record Set 2, to create just four RadMARC records, where each of the records had a 100 or 110 or 111 or 130 field. We then populated all other indexable author, title, and subject fields/subfields that occurred 1,000 or more times with appropriate tokens. The content designation that occurred 1,000 or more times totaled 112 fields/subfields. However, because of the constraints on using MARC fields 100, 110, 111, and 130, each record did not have all 112 fields/subfields populated. The following shows the total number of fields/subfields populated in each of these four RadMARC records:

UNTRadMARC201 (For 100 (personal name) & 700 fields): 96 fields/subfields populated
UNTRadMARC202 (For 110 (corporate name) & 710 fields): 86 fields/subfields populated
UNTRadMARC203 (For 111 (meeting name) & 711 fields): 87 fields/subfields populated
UNTRadMARC204 (For 130 (uniform title) & 730 fields): 83 fields/subfields populated.

Appendix E contains these four RadMARC records in human-readable (as opposed to machine-readable) form. Both sets of RadMARC records are available from the project website.

The creation of the RadMARC records was a manual process using MARC Magician, a MARC record software program, and a text editor. Appendix F contains a description of the RadMARC record creation procedures.

Based on our experience to date, we believe that the RadMARC approach can be used to develop any set of RadMARC records as well as custom built diagnostic records libraries can use to interrogate their systems' behavior. With custom built RadMARC records, libraries will be able to diagnose the indexing policies actually in effect on their systems to verify vendor configurations. Individual RadMARC records can be created to exercise specific indexing policies.

4.5. Use of the RadMARC Records and Test Searches

The RadMARC records are loaded into the bibliographic database of an online library catalog. Since these are legitimate instances of MARC records, they will be loaded through whatever normal record loading process is used by the online catalog.

To verify this process, we loaded Record Set 1 records into the original Z-Interop reference implementation of a Z39.50 server and online catalog system provided to the project by Sirsi Corporation. The records loaded properly. Then the records were indexed for searching using the indexing policies set up for the Z-Interop testbed. Once loaded and indexed, the records could now be searched.

We carried out a set of test searches against the records loaded on the Z-Interop reference implementation to check the validity of our approach. To illustrate how this works, we use the RadMARC record in Figure 4.

100	1 $a rm1001a11r, rm1001a21r, [Demo Record] $d rm1001d11r. [Demo Record]
245	10$a rm2451a11r rm2451a21r rm2451a31r : [Demo Record]$b rm2451b11r rm2451b21r rm2451b31r / [Demo Record]$c rm2451c11r rm2451c21r rm2451c31r. [Demo Record]
440	0$a rm4401a11r rm4401a21r rm4401a31r [Demo Record]
490	1 $a rm4901a11r rm4901a21r rm4901a31r [Demo Record]
600	10$a rm6001a11r rm6001a21r, [Demo Record] $d rm6001a11r. [Demo Record]
650	0$a rm6501a11r rm6501a21r rm6501a31r [Demo Record] $x rm6501x11r [Demo Record]$v rm6501v11r [Demo Record]$z rm6501z11r. [Demo Record]
651	0$a rm6511a11r rm6511a21r [Demo Record]$x rm6511x11r. [Demo Record]
653	_$a rm6531a11r rm6531a21r rm6531a31r [Demo Record]

700	1 $a rm7001a11r rm7001a21r, [Demo Record] $d rm7001d11r. [Demo Record]
710	2 $a rm7101a11r rm7101a21r. [Demo Record]

Figure 4. RadMARC Record Used in Testing

In this record, the following fields/subfields are associated with Author, Title, and Subject content in the record, based on the indexing guidelines document discussed previously:

Author

100	$a $d
245	$c
700	$a $d
710	$a

Title

245	$a $b
440	$a
490	$a

Subject

600	$a $d
650	$a $x $v $z
651	$a $x
653	$a

We used the Z-Interop reference implementation of a Z39.50 client provided to the project to send the searches to the Z-Interop reference implementation Z39.50 server and online catalog. For this testing purpose, the searches were sent manually.

One of the searches defined by the Bath Profile and the U.S. National Profile is an author keyword search. The Z39.50 search uses the following Bib-1 attribute combinations to denote the search as an author keyword search:

Attribute Type	Attribute Values	Attribute Names
Use (1)	1003	Author
Relation (2)	3	Equal
Position (3)	3	any position in field
Structure (4)	2	Word
Truncation (5)	100	do not truncate
Completeness (6)	1	incomplete subfield

Since it is an author search, we send the tokens that appear in the six author related fields/subfields, 100$a, 100$d, 245$c, 700$a, 700$d, and the 710$d. Each search contains the attribute combination and a single token. The following table shows the server response in the third column

Search Type	Query Term	Server Response
BP0.1	rm1001a11r	Success
BP0.1	rm1001a21r	Success
BP0.1	rm1001d11r	Success
BP0.1	rm2451c11r	Success

BP0.1	rm2451c21r	Success
BP0.1	rm2451c31r	Success
BP0.1	rm7001a11r	Success
BP0.1	rm7001a21r	Success
BP0.1	rm7001d11r	Success
BP0.1	rm7101a11r	Success
BP0.1	rm7101a21r	Success

For all author keyword test searches, the Z-Interop server responded correctly and retrieved the single RadMARC record. Therefore, we can state the following:

- The Z39.50 server properly processed the author keyword search
- All six author related fields/subfields (i.e., 100$a, 100$d, 245$c, 700$a, 700$d, and the 710$d) were indexed to support an author keyword search
- The level of Interoperability between the Z39.50 client and Z39.50 server using a Record Set 1 RadMARC record for this particular search was 100%.

More importantly for the project, we were able to confirm that the use of the tokens as structured, and the RadMARC records, provide us a meaningful approach for interoperability testing and allow us to address the following levels from the Z-Interop2 interoperability testing question space:

- Profile conformance level
- Information retrieval system level
- Bibliographic record level

The next step in the project was to develop the automatic testing software and processes to improve the efficiency of interoperability testing.

4.6. Automatic Testing Software and Processes

Another key objective in our exploration of an alternative to the original Z-Interop testbed was to develop ways to automate the testing and analysis activities that had been very labor intensive in the original approach. This section describes the development of testing software.

Once a database has been loaded with one or more RadMARC records, a client can test that system's indexing and searching functionality by issuing searches that expect to return specific records. For example, a database that contains a record with a particular token in its 245$a should return the record when queried with a search for that token against a title index, and using the appropriate Z39.50 query to express the query (e.g., as defined by Bath Profile "title keyword" query). Conversely, so long as the same token does not appear elsewhere in the record, a search for that token in a subject index should *not* find the record.

Test searches such as these may be sent by any conforming Z39.50 client, but it is more efficient to automate testing using one or more scripts. We took a two-level approach to building such scripts: at the low level, we created a domain-specific "little language" specialized for such scripts; and at the higher level, we created an initial set of scripts in that language, both as a useful partial test-suite for servers claiming Z39.50 profile conformance, and as proof of concept of the language/script division.

Although initial designs for the scripting language consisted of only a few domain-specific primitives, it quickly became apparent that scripts may in general need to make use of logical and looping constructs, and perhaps variable assignments and procedure definition/invocation, such as are provided by mainstream programming languages. Accordingly, we decided that the most efficient approach would be to build our language on top of a well-supported, expressive, existing language. Practical considerations indicated that Perl was the most appropriate choice.

Index Data, a software firm in Denmark, served as a contractor for the Z-Interop2 Project, and programmers with Index Data developed the Perl scripts used for automating the interoperability testing. There are three primary components that are involved, and each of these will be discussed briefly here, with supporting materials provided in appendixes. Since some of the components are extensions to Perl there are other Perl modules, programs, and files that the test harness, referred to as the RadioMARC Perl module depends on in order to work. More detailed information is provided below.

4.6.1. A Perl Module for Testing Z39.50 Servers

One component module is called **Net::Z3950::RadioMARC**, which is a Perl extension for testing Z39.50 servers known to contain copies of specific RadMARC records and to produce reports dependent on whether or not the expected records are present in the result set. The specific test scripts used in conjunction with Net::Z3950::RadioMARC is discussed in section 4.6.3.

This module provides the “harness” in which test scripts can be written for detecting the presence of a RadMARC record in a Z39.50-accessible database, and determining how that database indexes the record. Its key provision is the `test()` method, which runs a search for some well-known term that is known to occur in a RadMARC record, and generates different output dependent on whether the record is found or not.

The module contains a number of methods, three of which are important domain-specific operations::

- **set:** sets the value of named parameters – in this case, the connection details for the server to be tested, and the number of seconds to delay between searches in order to avoid overloading the server.
- **add:** registers a set of RadMARC records, added from the named file, which are believed to exist in the server being tested.
- **test:** carries out the actual testing. First, it creates the Z39.50 connection with the server if no connection has already been made. Then it performs the search specified as its first argument. This argument expresses the query in the widely used Prefix Query Format (PQF), as described in the *YAZ User's Guide and Reference* (Hammer, et al., 2004). The same query is used on the client side to select which of the previously **add**ed records is the target for the query, and the result set returned by the server is inspected for that record's presence. A message is printed depending on whether or the record is found, or whether the search failed completely – for example, because the server does not support the specified access-point.

A typical simple script follows:

```
set host => 'z3950.loc.gov', port => '7090', db => 'voyager';
set delay => 3;
add "filename.marc";
test '@attr 1=4 01245a01', { ok => '245$a is searchable as 1=4',
 notfound => 'This server is broken' };
```

In summary, **Net::Z3950::RadioMARC** has all of the methods/functions that users will need to use for interoperability testing. It allows them to configure the testing environment via the **set** method, specify the list of records that the tests are supposed to find via the **add** method, and then perform tests and obtain results via the **test** method.

The current version of this Perl extension is **Net-Z3950-RadioMARC-0.07** and available on CPAN: Comprehensive Perl Archive Network, <http://search.cpan.org/~mirk/Net-Z3950-RadioMARC-0.07/>. This extension to Perl is free software; that can be redistributed and/or modified under the same terms as Perl itself, either Perl version 5.8.3 or, any later version of Perl 5.

4.6.2. A Perl Module for Indexing MARC Records

One component to the Perl extension discussed above is the indexing of the RadMARC records. Index Data also created the following Perl extension to carry out the indexing: **Net::Z3950::IndexMARC**. This is part of the **Net::Z3950::RadioMARC** distribution available under the same licensing agreement, at: <http://search.cpan.org/~mirk/Net-Z3950-RadioMARC-0.07/>. The **Net::Z3950::RadioMARC** module uses the **Net::Z3950::IndexMARC** module.

This module provides a comprehensive inverted index across a set of MARC records, allowing simple keyword retrieval down to the level of individual field and subfields. However, it does this by building a big Perl data-structure (hash of hashes of arrays) in memory, and makes no efforts whatsoever towards optimization. It is quite suitable to the RadMARC interoperability testing since only a small number of RadMARC records are used.

Net::Z3950::IndexMARC is mainly used by the **Net::Z3950::RadioMARC add** method to create an inverted index of the records specified in the test set (i.e., the RadMARC records created for purposes of interoperability testing). **Net::Z3950::IndexMARC** is also used by the **Net::Z3950::RadioMARC test** method to retrieve the appropriate records for purposes of comparison. This is necessary because there can be more than one record in the test set, and the RadMARC test environment needs a method to search/retrieve the "correct" record from the test set so that it can compare the expected record to the record actually retrieved from the server which is being tested for interoperability. It uses the inverted index created by the IndexMARC module to search through its test set independently of the Z39.50 server. If it doesn't find the record in its own index, then the test will fail regardless of whether or not it found a record on the Z39.50 server.

4.6.3. Test Scripts

The **Net::Z3950::RadioMARC** and the **Net::Z3950::IndexMARC** can be considered a set of tools to be called when needed by a testing script. The type of testing script developed for the Z-Interop2 Project was aligned with the searching requirements and the question space for the project. The Bath Profile and the U.S. National Profile were developed to assist implementers of Z39.50 clients and servers to configure their implementations in specific ways to improve interoperability. We relied upon the specifications defined in the profiles to develop the set of test searches to be tested. Once the search requirements were defined, these needed to be incorporated into a script that would automate the sending of appropriate searches using the **Net::Z3950::RadioMARC test** method.

Index Data produced a Perl script, called **bathtest.pl** (which is contained in Appendix H) that uses the functions of **Net::Z3950::RadioMARC** and provides the instructions to issue specific types of searches, to send a specific token from the RadMARC record, and to generate output from the testing.

In the following, we provide an overview of some of the functions of the test script, and Appendix G contains a more complete discussion of using the **Net::Z3950::RadioMARC** and the **bathtest.pl** script.

First, the test script calls the **Net::Z3950::RadioMARC** module:

```
use Net::Z3950::RadioMARC;
```

Next, a set of variables are initialized and default values established:

```
my $pattern = 'rmFFF1S1r';
```

`my $pattern` represents a regular expression that describes the type of radioactive MARC tokens that are used as the basis for retrieving our radioactive MARC records. Refer back to 4.3. The Radioactive MARC Record Structure, for the discussion of the structure of the tokens.

```
my $combo;
```

my $combo stores the Z39.50 Bib-1 attribute combinations for expressing the searches.

```
my $hosturl = 'research.lis.unt.edu:2200/zinterop';
```

my $hosturl stores the IP address, port, and database name of the Z-server to be tested in the form "IP:PORT/DB".

Next we need to set some of the defaults for the **Net::Z3950::RadioMARC** module for this particular script using that module's **set** method.

```
set host => $host, port => $port, db => $dbname;
set delay => 1;
set identityField => '001,035$a';
set verbosity => 1;
```

We are now ready to establish variables that represent the attribute combinations for specific searches. In this sample we define the attribute combination for keyword and keyword with right truncation searches in the variables **$attributes_kw** and **$attributes_kwt** variables respectively.

```
my $attributes_kw = '@attr 2=3 @attr 3=3 @attr 4=2 @attr 5=100 @attr 6=1';
my $attributes_kwt = '@attr 2=3 @attr 3=3 @attr 4=2 @attr 5=1 @attr 6=1';
```

Using the **Net::Z3950::RadioMARC add** method, we can add a RadMARC record to the test set, and when we carry out the testing and receive records from the server being tested, we expect our queries to return this record for specific searches.

```
add 'RadMARCATS1';
```

Finally, we set up for the output of the testing by labeling the output with a brief header that describes the test.

```
print "Bath compliance test script\n\n";
print "Test date: " . localtime() . "\n";
print "Test target: $hosturl\n";
```

The next key part of the script called **@types** sets up the MARC fields/subfields that will be tested, and how those fields are to be tested (i.e., what kinds of searches should be directed at those fields based on the definitions provided in the Bath Profile. The array that is created contains a list of hash references, each of which uses the following indexes: **name** holds the name of the Bath Profile search (in "human readable" form); **use** holds the appropriate use attribute for the given search type; and **fields** holds a reference to a list of strings that represent the subfields to be tested under that search type (in the format **FFF$s**).

```
my @types = (
  {
    'name' => 'Author search (BP0.1).',
    'use' => 1003,
    'fields' => [
      '100$a', '100$d',
      '245$c',
      '700$a', '700$d',
      '710$a'
```

One of the key objectives in exploring an alternative approach to interoperability testing was to automate the testing activities as much as possible. In a RadMARC record, there may be dozens or more tokens, and the script has a procedure for automatically generating appropriate tokens as search terms for the queries that will be issued. The script has a subroutine called **radtoken** that returns a valid RadMARC

token based on the **$pattern** (defined earlier) and when given a **FFF$s** (e.g., 245$a), the subroutine would return a well-structured token such as **rm2451a1r**.

```
sub radtoken {
  $_ = shift;
  my $ret = $pattern;

  my ($field, $subfield) = /(...)\$(.)/;
  $ret =~ s/FFF/$field/;
  $ret =~ s/S/$subfield/;
  return $ret;
```

This brings us to actual testing. The searches are those defined in the Bath Profile, Level 0:

- Author Keyword Search (BP0.1)
- Title Keyword Search (BP0.2)
- Subject Keyword Search (BP0.3)
- Any Keyword Search (BP0.4)

After the report heading is produced:

```
print "Testing Level 0 keyword searching (BP0.1 to 0.4).\n\n";
```

a **foreach** structure loops through each **@type** element. For each element (i.e., for each type of Bath Profile search), we take two steps:

1. The complete attribute combination is built by taking the "use" attribute specified in the **@type** structure and combining it with the other attributes necessary for a keyword search, which are held in the variable **$attributes_kw**.
2. The **runtest** subroutine is used to test each field belonging to a particular Bath Profile search type. As shown below, **$combo** stores the complete attribute combination needed for the test and **$_->{fields}** contains a list of fields that should be tested for each search type.

```
foreach (@types) {
  my $combo = "\@attr 1=" . $_->{'use'} . " $attributes_kw";
  print "Testing: " . $_->{name} . "\n\n";
  runtest $combo, $_->{fields};
  print "\n";
```

More details about the test script can be found in Appendixes G and I, but the above provides a description of the logic and processes that the script carries out.

Figure 4 provides an illustration that shows the relationships and interactions of the **bathtest.pl** script, the **Net::Z3950::RadioMARC**, and the **Net::Z3950::IndexMARC** modules.

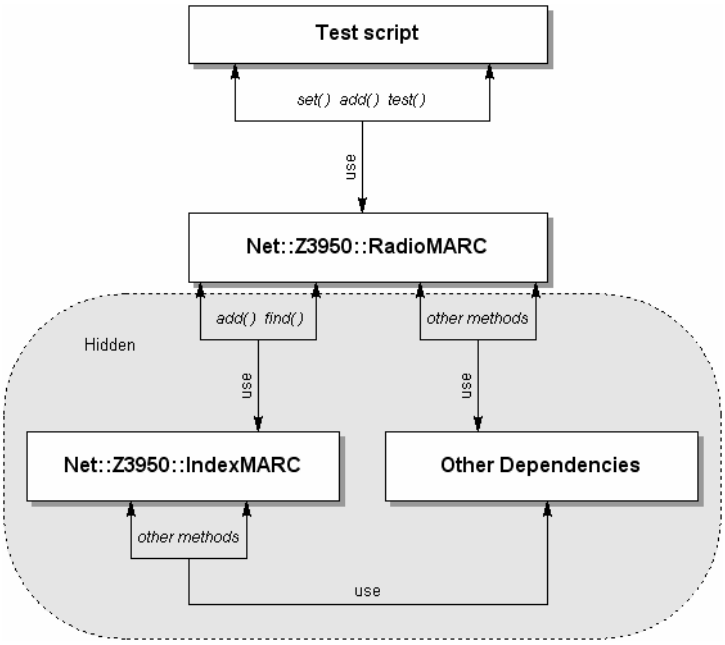


Figure 4. Interaction Among Components for Automated Testing

Figure 5 shows the complete picture of the interoperability testing framework with emphasis on the automated testing components.

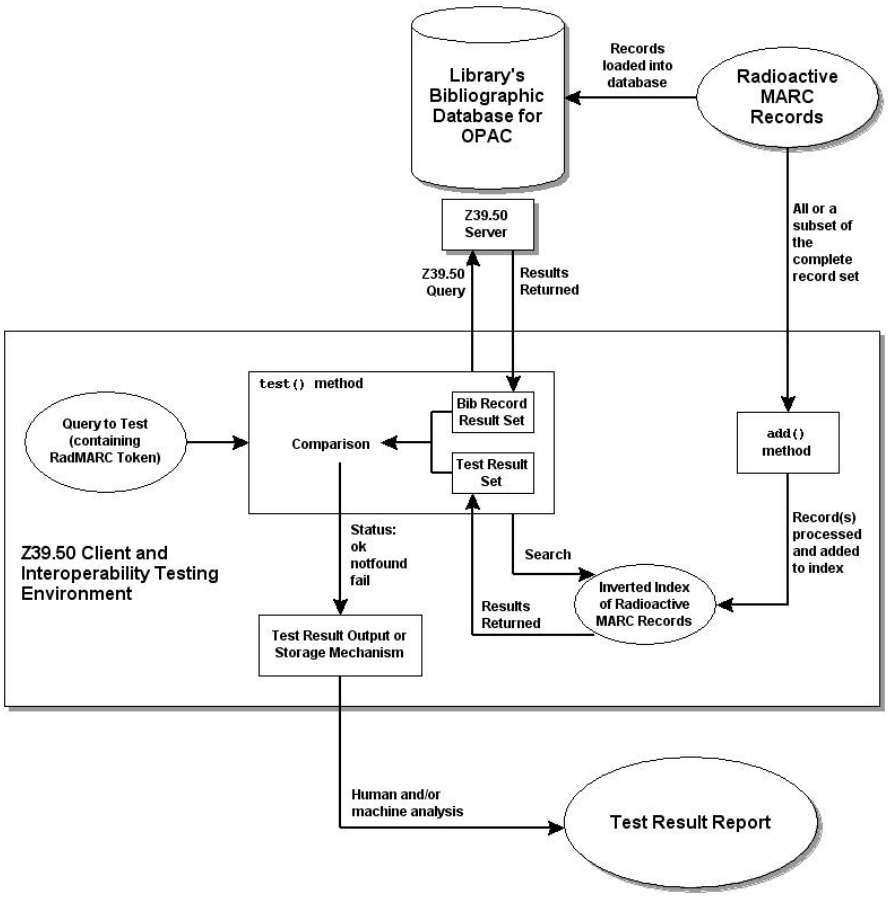


Figure 5. Testing Environment and Processes

4.6.4. Output from bathtest.pl Script

It was also important to generate a usable report for the results of the interoperability testing using the test script. Appendix J contains sample output from our testing against the Z-Interop reference implementation Z39.50 server and bibliographic database. The following extract from that output is explained to show what we can learn from the interoperability testing.

First, there is a heading for the report that indicates what test script was used, the date, and the database/server that was being tested:

```
Bath compliance test script

Test date: Thu Feb 17 12:03:50 2005
Test target: research.lis.unt.edu:2200/zinterop
```

Next, the report indicates which Bath Profile searches were tested:

```
Testing Level 0 keyword searching (BP0.1 to 0.4).
```

Then, the report summarizes the results for each of these Bath Profile Level 0 searches. The one below is for an Author Keyword Search (BP0.1).

```
Testing: Author search (BP0.1).
```

```
Search finds 100$a
Search does NOT find 100$d
Search finds 245$c
Search finds 700$a
Search finds 700$d
Search finds 710$a
```

What can we tell from these results?

First, the server successful executed the search, providing answers to the question space levels related to Profile Conformance and Information Retrieval System. The target server received and processed the Z39.50 query successfully and the underlying information retrieval system was capable of carrying out a keyword search.

Second, we can answer the question space level related to the Bibliographic Record, specifically the indexing policies. The appropriate tokens sent as search terms matched the tokens in the 100$a, 245$c, 700$a, 700$d, and 710$a. This means that each of those fields/subfields was indexed to support an Author Keyword Search. We also see that the appropriate token was not found in the 100$d, which means that the server does not index the 100$d to support an Author Keyword Search. The 100$d contains dates associated with a name as in "**$d**1088-1143". While this information is related to author data, and could be considered an indexable field to support author searches, a typical author keyword search is likely not to be for a date, and so the decision not to index this field is understandable.

At this point in the interoperability testing procedures, human analysis of the output from the test script is necessary to interpret the results.

4.7. The MARCdocs Database

Another development effort undertaken during the Z-Interop2 project, which was not anticipated at the time of the request for extension, was to create a source of MARC 21 Bibliographic Format documentation in a machine-readable and manipulable form. The MARC 21 Documentation Database, referred to as the MARCdocs database, served two key functions within the interoperability testing environment:

- To assist in report generation by using field and subfield names associated with their tag and code values
- To identify selected content designation to use in RadMARC records based on frequency of occurrences determined in a previous analysis.

MARCdocs was a pilot effort aimed at structuring the textual documentation from the MARC 21 Format for Bibliographic Data into a relational database. The project team investigated whether such a database application for the MARC documentation existed and none were found. Therefore, we began the development of such an application for the Z-Interop2 Project. Using a database approach for the authoritative MARC documentation provides new opportunities for various applications, including more efficient maintenance of the documentation, easier and quicker updates and changes, exporting selected data in XML, as well as an aid to research into interoperability, the development and evolution of MARC, a learning/reference tool for those seeking to understand the MARC format, and a tool to help those devising new bibliographic structures.

The application uses open source software tools including Linux, MySQL, and PhP. Project team member, Jason Thomale, designed and implemented the database, developed procedures for data loading, and designed and programmed the web interface for the MySQL database. A publicly web-accessible version of the database is available at: <http://meta.lis.unt.edu/MARCdocs2/>. Appendix K contains the documentation describing the database structure.
The first phase of development was to produce the database application containing the MARC documentation. The second phase of development was to extend the database application to support

certain procedures for Z-Interop2 interoperability testing. Initially, we envisioned that extending the database application would allow us to store appropriate information to assist in the creation of RadMARC records. In addition, the information contained in the database would also support the generation of the Z queries used in the interoperability testing.

The project staff discussed how the MARCdocs application could be extended to support higher levels of automated MARC record creation, interoperability testing, and report generation, as indicated in Figure 6.

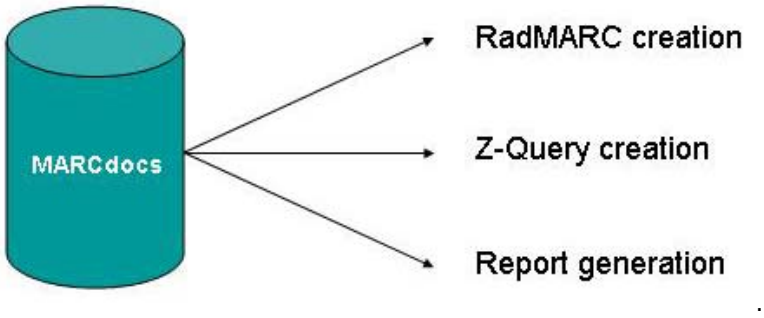


Figure 6. Potential Uses of MARCdocs Database for Interoperability Testing

4.7.1. Extensions to the MARCdocs Database

We began work on extending the MARCdocs database by drafting an project document called *Creating Radioactive MARC Records and Z Queries Using the MARCdocs Database*, which is contained in Appendix L. This provided a basis for discussion among the project team and the Index Data contractors.

Two primary pieces of information need to be stored in the MARCdocs database to support RadMARC and Z-Query creation:

- Indication of which subfields could be indexed to support U.S. National Profile searches, and which specific searches are supported
- One or more tokens for each of these subfields

As discussed in Section 4.4, we populated the RadMARC content designation based on the frequency count analyses carried out as part of the original Z-Interop Project. We also identified various sets of subfields that could be indexed to support the Bath Profile and U.S. National Profile Level 0 searches. We used the frequency count of the indexable fields to select sets of subfields to populate in the RadMARC records. Thus, a third piece of information that needed to be stored in the MARCdocs database was the frequency count for all content designation. The frequency count number was taken from our previous analysis.

Based on the above requirements, we began extending the database structure to store the following information:

- **Frequency count of all MARC content designation:** These data were stored in a new database element and the data value would be an integer. Data for this element were batch loaded from the frequency analysis spreadsheet reports

- **Indexable fields to support U.S. National Profile searches:** A field/subfield can be indexed to support one or more of the searched defined in the Bath Profile and U.S. National Profile. For example, the 245$a can be indexed to support the following searches:

    - Title Search – Keyword
    - Title Search – Keyword with Right Truncation
    - Title Search – Exact Match

- Title Search – First Words in Field
- Title Search – First Characters in Field

Therefore, the database element related to the searches supported by the 245$a would contain the profile label for each of these searches:

- BP0.2
- BP1.5
- BP1.6
- BP1.7
- BP1.8

Initial data values for this new database element were batch loaded from the indexing guidelines document which identifies all candidate subfields that could be indexed for Bath Profile and U.S. National Profile searches.

In the course of the parallel development of the automatic test scripts discussed in Section 4.6 above, we concluded that there was no need to store tokens in the database application, since they could be created for the queries on the fly by the test script. We also realized that the level of effort to use this database application to automatically create the RadMARC records was not cost-effective compared to the manual creation of the RadMARC records.

The extended version of the MARCdocs database, however, was used for the following purposes in the Z-Interop2 Project:

- Identify a set of fields/subfields that occurred at a specified occurrence level based on the frequency count data stored for each of the content designation structures, and these were used in the creation of the RadMARC records
- Identify a set of fields/subfields associated with a specific Bath Profile and U.S. National Profile search for use in the test script that would create and issue the queries for interoperability testing.

In addition, the original MARCdocs database has potential for use in other ongoing and future research projects.

4.8. Concluding Thoughts on the Research and Development Activities

In the preceding sections, we have described the research and development activities carried out in exploring an alternative approach to interoperability testing. We were able to demonstrate the validity of this conceptual approach by carrying out testing with the Z-Interop reference Z39.50 server and online library catalog implementation.

The specific deliverables produced through these activities are:

- A conceptual framework and methodology for interoperability testing using radioactive metadata records
- Two set of RadMARC records usable for interoperability testing a set of searches defined the Bath Profile and the U.S. National Profile, Levels 0 and 1
- Automatic testing modules and scripts to automate the sending of searches, the analysis of retrieved records, and the reporting of the results
- A database application (i.e., MARCdocs) used to support the creation of RadMARC records and to support the automatic testing modules and scripts for issuing queries.

These deliverables were documented, and the documentation is available on the project website.

It is best to consider this project as a proof-of-concept of a very innovative approach to a difficult problem, namely, testing and assessing interoperability of two systems. Responses to presentations made by the

Principal Investigator in recent months indicated a high-level of interest in the methodology and approach of radioactive metadata records for interoperability testing and other applications.

5. The Z-Interop2 Project: Products, Participation, and Dissemination

The preceding section described the research and development activities carried out in our exploration of an alternative approach for interoperability testing. This section identifies the work accomplished in terms of products, participation, and dissemination.

5.1 Interoperability Testbed Products

The goal of Z-Interop2 was to develop an approach to interoperability testing using specially created diagnostic metadata records and automatic testing scripts to conduct interoperability testing in the context of Z39.50 servers and online library catalogs. Section 4 described the research and development activities, and we summarize the products currently available as a result of the project:

- **Conceptual framework and methodology:** At the outset of the project, the Principal Investigator, project staff, and contractors on the project developed a framework for interoperability testing. This document, contained in Appendix B, provided the general ideas for this testing approach. As the project evolved, most of the general concepts and ideas contained in the framework were valid. Some specific details (e.g., the nature of the tokens contained in the RadMARC records) were changed as our understanding increased.

- **RadMARC records:** Two sets of RadMARC records were created. These are legitimate instances of MARC records, with selected indexable fields/subfields populated with special tokens. Determining the structure of the tokens was one of the major developments during the project. Our goal was to have a generalized approach to the tokens that could be customized for MARC records, but also an approach that could be adapted to other metadata systems beyond MARC. A total of 14 RadMARC records were created, confirming our initial assumption that only a small number of such specialized diagnostic records would be sufficient to carry out a wide range of interoperability tests. Specifically, these records enable interoperability testing of a range of author, title, subject, and general keyword searches defined the Bath Profile and the U.S. National Profile, Levels 0 and 1.

- **Automatic testing modules and scripts:** A key challenge for an effective approach to interoperability testing is to automate as many procedures and operations automated as possible. In the approach explored in this project, interoperability testing requires sending a large number of searches and automating the sending of appropriate searches was essential. In addition, having software do the initial analysis of the retrieved records for those searches makes it more cost-effective and less labor intensive.

  Index Data, a contractor on this project, was responsible for developing the automatic testing modules and scripts to automate the sending of searches, the analysis of retrieved records, and the reporting of the results. One primary component in that suite of modules and scripts is **Net::Z3950::RadioMARC**, which is a Perl extension. This is currently available as free software on CPAN. In addition, Index Data developed the initial ***bathtest.pl*** testing script, which uses the **Net::Z3950::RadioMARC** module. While the ***bathtest.pl*** script was used for interoperability testing for Z-Interop2, the script included in Appendix H serves as an example of the type of testing script that can be developed by others who want to conduct specific tests on their library catalog systems.

- **A database of MARC 21 documentation:** The MARCdocs database, an application that stores information from the documentation for the MARC 21 Bibliographic Format, was created. To our knowledge, this is the first publicly available database version of the MARC 21 documentation. For the Z-Interop2 Project, MARCdocs served as a source of information for the creation of RadMARC records and for data for creating the test searches issued by the ***bathtest.pl*** script.

The MARCdocs database is publicly available. Part of the development of this application was done with resources of the Principal Investigator.

Documents related to each of these products of the Z-Interop2 are publicly available, and they are listed in Appendix M of this report.

5.2 Participation

Interoperability testing was carried out with the University of North Texas (UNT) Libraries. The UNT Libraries have an implementation of an Innovative Interfaces integrated library system. Testing was generally successful and this also presented some issues that had not been evident when testing the procedures, records, and automatic testing scripts with the Z-Interop reference implementation. One key challenge was that the UNT Libraries' server would stop accepting searches at arbitrary points. This problem was resolved by segmenting the issuing of test searches in different sessions with the server. Please see Section 6 Outstanding Issues.

5.3 Dissemination

The project website <http://www.unt.edu/zinterop> was the primary mechanism to publish publicly available technical reports, documents, and other relevant information about the project during the research and development activities. Various documents, both draft and final, were posted on the site.

In the past six months, the Principal Investigator has made presentations at three conferences which discussed the Z-Interop2 Project:

- ASIS&T Annual Meeting, October 2005, Raleigh, NC. ***An Alternative Approach to Interoperability Testing: The Use of Special Diagnostic Records in the Context of Z39.50 and Online Library Catalogs***
- Access 2005, October 2005, Edmonton, Alberta. ***Radioactive Metadata Records: An Interoperability Testing Approach Based on Metadata Utilization***
- Coalition for Networked Information 2005 Spring Task Force Meeting, April 2005, Washington, DC. ***A Radioactive Metadata Record Approach for Interoperability Testing Based on Analysis of Metadata Utilization***

To date, one paper has been published in **Proceedings of the 68th ASIS&T Annual Meeting Volume 42, 2005.** This paper entitled, "An Extensible Approach to Interoperability Testing: The Use of Special Diagnostic Records in the Context of Z39.50 and Online Library Catalogs," is available in the draft pre-print format at: <http://www.unt.edu/wmoen/publications/AsistPaperPreprinMoenJune2005.pdf> and also contained in Appendix N.

The Principal Investigator is committed to continuing disseminating information about the Z-Interop2 Project in conference presentations, papers, and articles.

6. Outstanding Issues

There are two areas in which we did not fulfill the objectives we outlined for the Z-Interop2 Project. This was in part due to two factors: 1) the extent of work required in our exploration of this alternative approach to interoperability testing, and 2) the loss of the original Z-Interop reference implementation of a Z39.50 server and online catalog implementation contributed by Sirsi Corporation for the original project.

As an exploratory project, the Principal Investigator tried to estimate the level of resources required to carry out the work. Several of the key activities took more time and effort than had been estimated. Specifically, the work related to automating the interoperability testing through the use of the perl modules and scripts took more time than we had anticipated. In addition, we found that to create the RadMARC records, we would have to use a manual record creation process. Although the MARCdocs database was critical for both the RadMARC record creation (i.e., to know which fields/subfields to population) and for

the query generation by the test script (i.e., to know which searches to send based on the searches defined in the Bath Profile and the U.S. National Profile), it's development and enhancement was also a time-consuming process.

Second, in May 2005, our primary Sun server that hosted the reference implementation was comprised by an malicious hacker. UNT Computer Security required that the machine be pulled off the network for investigation. The loss of this machine and the reference implementation precluded us from continuing testing of the Z-Interop2 procedures, RadMARC records, and automatic testing software. Over the summer, one of our Ph.D. students worked to set up a new application, Cheshire, which we hoped would serve as a testing environment, but this was not completed prior to the end of the Z-Interop2 Project at the end of August 2005.

The two objectives which we did not complete fully were:

- Testing with 3-5 libraries
- Updating the indexing policies

We successfully proved the concept of interoperability testing using radioactive MARC records in the initial testing with the Z-Interop reference implementation, and in testing with the UNT Libraries. However, we were not able to carry out interoperability testing with other libraries.

We focused on interoperability testing for author, title, and subject searches as defined in the Bath Profile and the U.S. National Profile, Level 0 and Level 1. We were not able to carry out testing of other searches defined in those profiles for Level 1, such as Standard Identifier Search, ISBN Search, ISSN Search, Remote System Record Number Search, Date of Publication Search, Language Search , and Format of Material Search. With the focus on the author, title, and subject searches, we were able to use the indexing guidelines originally developed during the Z-Interop Project, and did not need to extend those since we were not able to address these other types of Level 1 searches.

While we were not able to complete all objectives originally envisioned for this project, those objectives are still ones which the Principal Investigator plans to address in the coming months. See the following section for future work and explorations based on the outputs of the Z-Interop2 Project.

7. Future Work

There are several areas in which the Principal Investigator will continue the work initiated through the Z-Interop2 Project. Two of these related to unfinished objectives from the project, and several relate to activities that have been identified as useful for extending the concept of radioactive metadata records and disseminating additional information about the project.

7.1. Additional Interoperability Testing

One of the first activities following the submission of this final report will be to conduct interoperability testing with additional libraries. This interoperability testing will continue to focus on the author, title, and subject searches defined in the Bath Profile and the U.S. National Profile, Levels 0 and 1. This interoperability testing will be concluded within the next six months. Following the completion of this testing, we will make enhancements (i.e., extend the content designation populated with tokens) to RadMARC records to enable interoperability testing using other Level 1 searches defined in the U.S. National Profile, specifically standard identification type searches. We will then carry out interoperability testing of these searches to ensure the records and revised testing procedures work properly.

7.2. Indexing Guidelines

In conjunction with the second set of testing discussed in 7.1., we will review the existing indexing guidelines to recommend additional indexing policies appropriate to improve interoperability among Z39.50 clients and servers.

7.3. Revisions to RadMARC Records Based on the Current MARC Analysis Project

The Principal Investigator has a project underway, funded by IMLS, to analyze the utilization of MARC content designation (i.e., the MCDU Project, <http://www.mcdu.unt.edu>. As noted in Section 4.4. above, the decision for populating certain content designation in the RadMARC records was based on an analysis of the Z-Interop test dataset of 400,000 MARC records from the WorldCat database. In the MCDU Project, we are analyzing over 56,000,000 MARC records from the WorldCat database. This current analysis will provide data regarding the most frequently used fields/subfields, and we will revise or create new RadMARC records to reflect the frequently used, indexable fields/subfields identified in the MCDU Project. As noted in Section 4.4., the RadMARC Record Set 1 has records that reflect different types of materials cataloged. The MCDU Project is investigating the content designation per format of material described. This may result in data that allows us to refine the content designation populated in the 10 RADMARC records. In addition, the results from the MCDU Project may yield valuable information in any revisions to the indexing guidelines referenced in 7.2. above.

7.4. Radioactive Metadata Records in Metadata Transformations

One of the most exciting ideas that emerged from the Z-Interop2 Project, and which the Principal Investigator has discussed in presentations during the past six months, is the use of radioactive metadata records to validate metadata record transformation from one metadata scheme to another.

A significant improvement to the usefulness of metadata is the ability to transform a metadata record created according to one scheme to a metadata record conforming to another scheme (e.g., from a MARC record to a Dublin Core record). For example, the Library of Congress through its Search and Retrieve via URL (SRU) allows a user to request a bibliographic record from its catalog database in conventional MARC, in MARCXML, in MODS, and in Dublin Core. Transformations are carried out on the original record and enable the presentation of the data (some or all of the data) in another metadata scheme.

We believe that radioactive metadata records can assist in the validation of the transformation processes. For example, a MARC record might have data in the 245, the main title field. The data in that field should map to the Dublin Core Title element. Using the tokens in the RadMARC records' 245$a from Figure 3:

> $a rm2451a1r rm2451a2r rm2451a3r : $b rm2451b1r rm2451b2r rm2451b3r / $c rm2451c1r rm2451c2r rm2451c3r.

we can analyze where those tokens end up in the Dublin Core, and if some data are not brought across to the Dublin Core record at all.

Reuse of metadata is essential, and different communities will want metadata records in forms they are familiar with. This implies the need for transformations so communities do not have to create metadata for objects where metadata already exists. Another application where this validation of metadata transformation through the use of radioactive metadata records is the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH). Customized radioactive metadata records can be created for any metadata scheme (standard or non-standard), and transformation processes for OAI Dublin Core records can be assessed.

The Principal Investigator, as resources are available, will explore this use of radioactive metadata records within the next 12 months.

7.5. Publications and Dissemination

Based on the feedback at presentations by the Principal Investigator on the Z-Interop2 project to date, there is interest in the radioactive metadata record concept for interoperability testing, metadata transformation assessment, and other diagnostic activities. Additional publications on the original Z-

Interop and the Z-Interop2 Project are needed. The Principal Investigator plans to produce articles for the following journals in the next 12 months:

- Journal of the American Society for Information Science and Technology (JASIST)
- D-Lib Magazine
- Information Technology and Libraries (ITAL).

Each of these articles will present concepts and methodologies related to the Z-Interop2 project as well as reporting results of interoperability testing. In addition, Jason Thomale, a project team member is currently working on a publication related to the test script and the Net::Z3950::RadioMARC module.

In addition, the Principal Investigator will maintain the project website, and will update the site with information as appropriate.

8. Concluding Thoughts

For nearly five years, the Principal Investigator has been able to conduct research into issues of interoperability in the context of Z39.50 and library catalogs. Thanks to the generous funding from IMLS, this research has produced new understandings and procedures for assessing interoperability. As research projects, Z-Interop and Z-Interop2 have broken new ground as well as confirmed previous assumptions about factors related to interoperability.

In our December 2003 interim report, we wrote:

> Acknowledging that interoperability is multi-faceted means that no one project will solve or even address all factors affecting interoperability. The Z-Interop Project focused on Z39.50 implementations providing access to library catalogs containing bibliographic records. Yet many of the issues addressed by the Z-Interop Project will help others – within and outside of the library community – to focus their attention on high-value interoperability problems. Even within the library community, there are other types of data that are exchanged. For example, specifications are now complete for exchanging holdings information and for exchanging authority records. To fully realize the vision of networked access to library resources will require ongoing research, testing, and improvement related to interoperability of systems and data. The Z-Interop Project has laid a solid conceptual, methodological, and technical foundation for such research and testing. In addition, Principal Investigator anticipates future use of the Z-Interop technical and software infrastructure to pursue research problems uncovered through this valuable project.

The Z-Interop2 Project pursued an alternative approach for interoperability testing from originally pursued in the original Z-Interop Project. In this project we delivered important products to move the assessment of interoperability forward. In the process, we validated the utility of a new conceptual approach using radioactive metadata records. While the term “radioactive metadata records” could be seen as problematic, it has been a descriptively effective metaphor for these specialized diagnostic metadata records.

At the recent Access 2005 Conference in Canada where the Principal Investigator was a keynote speaker and had the opportunity to discuss the Z-Interop2 Project, a member of the audience posted the following comment in his weblog:

> I wish I’d thought of “radioactive metadata”. As used in ZInterop2, described by Bill Moen this morning (who attributed the term to Sebastian Hammer), the idea is to craft diagnostic metadata records (in this case MARC but the idea would apply to any format) to test search systems: put unambiguous easy-to-search tokens in specific fields, mount them in your repository, index them, and then use standard search scripts to generate a report on which tokens were found and which ones weren’t. When I think of the time I’ve wasted trying to find OPAC records with specific patterns of terms to test whether a particular field is being indexed and how, I weep.

> (Posted by Peter Binkley (the Digital Initiatives Technology Librarian at the University of Alberta) in his weblog *Quædam cuiusdam* <http://www.wallandbinkley.com/quaedam/> on October 18th, 2005.)

We also identified potential uses for radioactive metadata records, and the Principal Investigator looks forward to exploring these ideas further in the coming year. Specifically, exploring the use of radioactive metadata records for validating metadata transformation processes seems to offer an important way of building on the understanding gained through IMLS funding of Z-Interop2.

The problems of interoperability will be with those of us working in the networked environment for as long as individual communities develop their own systems, their own metadata schemes, and their own specialized applications – yet want to interact with other information communities to share data and information. The Principal Investigator feels assured, however, that the research carried out in the Z39.50 Interoperability Testbed projects will continue to make contributions to our various communities' efforts to share information in ways that serve our users.

Appendix A: Project Work Plan

U.S. Federal Institute of Museum and Library Services
National Leadership Grant, Phase 2

A Radioactive MARC Record Approach for Interoperability Testing

Project Work Plan

William E. Moen, Ph.D.
<wemoen@unt.edu>
Principal Investigator

School of Library and Information Sciences
Texas Center for Digital Knowledge
University of North Texas
Denton, TX 76203

June 2004

Project Work Plan

1. Introduction

This document details a work plan to guide the planning and execution of a new phase of the Z39.50 Interoperability Testbed Project. In this phase, we will use a set of special, diagnostic MARC records to identify interoperability problems between a Z39.50 client and a Z39.50 server providing access to a database of bibliographic records supporting the search and retrieval functions of an online library catalog. We refer to these special, diagnostic records as radioactive MARC records. The project includes a number of separate but related activities. The structure of the work plan groups these activities into work areas, each of which describes the major activities, tasks, deliverables, timelines, and responsibilities. The work areas are:

- Creating MARC documentation database
- Developing Radioactive MARC records
- Identifying test searches
- Enhancing indexing guidelines for searching
- Creating testing procedures and processes
- Validating radioactive MARC record approach for interoperability testing
- Conducting interoperability testing

A number of these work areas will proceed in parallel, and there are interactions between the work areas.

2. Project Work Areas

2.1 Creating MARC documentation database

This work area focuses on creating a MySQL database to store MARC content designation information. The purpose of this database is to be able to quickly find all content designation structures that can be used for various formats of materials. The final implementation of the database will run under Linux; it will be web accessible via PhP scripts for administration as well as data input.

Activities	Tasks	Deliverables	Timeline	Responsibilities
Design and Prototype DB	• Identify requirements • Develop initial design • Prototype design in MySQL • Prototype user interfaces • Determine backup procedures • Develop XML export functionality	Draft database design; DB prototype	June 1-30	Jason Bill
Implement DB	• Finalize design and specifications • Implement DB under Windows for testing • Implement DB under Linux • Implement user interfaces and connect via PhP to database • Test DB application and interfaces	DB application with user interfaces for data input	June 15-July 15	Jason
Populate DB	• Identify sources of data that can be batch loaded • Identify sources of data		July 1-30	Jason Penny

Activities	Tasks	Deliverables	Timeline	Responsibilities
	that will be manually input • Load data • Input data			
Validate DB contents	• Test database with appropriate queries • Verify accuracy of data in database		July 15-30	Jason Penny
Documentation	• Write complete documentation for database application	Complete document describing DB, procedures, scripts, data loading, etc.	July 30	Jason

Work Area	July04	Aug04	Sep04	Oct04	Nov04	Dec04	Jan05	Feb05
MARC DB								

2.2 Creating MARC Records

This work area covers all the activities that are required to create the radioactive MARC records. This includes the identification of the character strings that will serve as data values for each field/subfield in a record, identifying tools for creating the records, and the actual creation and validation of the records.

Activities	Tasks	Deliverables	Timeline	Responsibilities
Requirements Setting	• Identify requirements for data for records • Determine options for data value character strings • Account for various format specific content designation • Determine process for generating character strings • Identify number and type of records required	Draft requirements document for discussion with Index Data	June 15-July 15	Bill Penny Index Data
Create character string patterns	• Develop tool for producing character strings • Produce character strings	Set of character strings available for data input	July 15-30	Index Data
Identify MARC creation tool(s)	• Identify functionality needed to create radioactive records • Identify potential tools for record creation • Install or implement tool		July 1-15	Penny Index Data
Input data into MARC records	• Create MARC records	Set of Radioactive MARC Records	Aug 1-Sept 1	Penny Jason
Validate MARC records	• Identify procedures to validate records		Aug 1-Sept 1	Penny Jason

Work Area	July04	Aug04	Sep04	Oct04	Nov04	Dec04	Jan05	Feb05
MARC Records								

2.3 Developing Test Searches

This work area covers the activities and tasks related to identifying and agreeing on the test searches to be used. The results of this work area inform MARC records needed and addressed in 2.2.

Activities	Tasks	Deliverables	Timeline	Responsibilities
Identify Searching	• Use U.S. National Profile		July 1-15	Penny

Activities	Tasks	Deliverables	Timeline	Responsibilities
Requirements	for Library Applications, Levels 0 and 1 • Determine character string patterns needed in the record for the searches • Determine indexing requirements (see 2.5)			Bill
Document Test Searches	• Prepare document that details each test search, search term, and expected search results	Test Searches document	July 15- Aug 30	Penny Jason

Work Area	July04	Aug04	Sep04	Oct04	Nov04	Dec04	Jan05	Feb05
Test Searches								

2.4 Creating Automated Testing Scripts

This work area covers the activities and tasks related to developing and creating automated mechanisms for issuing test searches and compiling results. Index Data has primary responsibility for this work area.

Activities	Tasks	Deliverables	Timeline	Responsibilities
	•			
	•			
	•			
	•			
	•			

Work Area	July04	Aug04	Sep04	Oct04	Nov04	Dec04	Jan05	Feb05
Testing Scripts								

2.5 Indexing Guidelines

This work area addresses the expansion of the existing indexing guidelines developed for the first phase of interoperability testing to address the interoperability testing using the radioactive MARC records.

Activities	Tasks	Deliverables	Timeline	Responsibilities
Create draft indexing policies	• Identify appropriate content designation to be indexed for each test search • Develop draft indexing policies • Distribute for review and comment	Draft indexing guidelines	Aug 1 – Sept 15	Penny Jason
Revise and finalize indexing policies	• Compile comments from review of draft • Clarify any questions on content designation to be indexed • Create final version of indexing policies	Indexing Guidelines for Interoperability Testing	Sept 15-30	Penny Jason

Work Area	July04	Aug04	Sep04	Oct04	Nov04	Dec04	Jan05	Feb05
Indexing Policies								

2.6 Validating Testing Procedures

This work area uses the existing Z-Interop reference implementation server and information retrieval system (Sirsi Unicorn) to test and validate all interoperability testing procedures using radioactive MARC Records.

Activities	Tasks	Deliverables	Timeline	Responsibilities
Implement indexing guidelines	• Review procedures for setting indexing policies in Sirsi system • Set up enhanced indexing policies on the system		Sept 1-15	Jason JungWon
Load records	• Use record loading procedures to load radioactive MARC records into system		Sept 15-30	Jason JungWon
Test automated scripts for issuing searches	• Issue all test searches using automated scripts • Identify any problems or issues • Revise and finalize automated scripts		Oct 1-15	Index Data Jason
Test programs for compiling results	• Review results of the searches and the reports generated • Identify revisions or enhancements needed • Finalize results reporting mechanisms		Oct 1-15	Index Data Jason
Documentation	• Develop summary documentation for validation and testing	Validation documentation	Oct 15-30	Jason

Work Area	July04	Aug04	Sep04	Oct04	Nov04	Dec04	Jan05	Feb05
Testing Validation								

2.7 Analysis and Reporting

This work area addresses the development of standard documentation for reporting results of interoperability testing. Analysis procedures will be documented.

Activities	Tasks	Deliverables	Timeline	Responsibilities
Analysis procedures	• Identify analysis procedures for results of testing • Test these procedures during 2.6 above • Finalize analysis procedures	Documentation of Analysis Procedures and Reports	Sept 15-Oct 1	Jason Penny
Report template	• Develop standard report template		Sept 15-Oct 1	Jason

Work Area	July04	Aug04	Sep04	Oct04	Nov04	Dec04	Jan05	Feb05
Analysis & Reporting								

2.8 Conduct Interoperability Testing with Libraries

This work area covers activities and tasks related to conducting interoperability testing with actual library online catalog and Z39.50 server implementations.

Activities	Tasks	Deliverables	Timeline	Responsibilities
Create testbed operation procedures	• Develop draft procedures document • Request review • Finalize procedures document	Testbed procedures document	Oct 1-15	Bill Jason
Radioactive MARC Record Availability	• Set up password protected web folder for records • Load records on site • Test access to records		Oct 1-15	Bill Jason
Limited interoperability testing	• Identify 3-5 libraries willing to go through testing procedure • Conduct Interop testing • Revise any procedures as necessary		Oct 15-Nov 30	Bill Jason Penny Index Data
Publicize testbed	• Prepare press release • Prepare call for participation • Disseminate information		Nov 15-Nov 30	Bill Jason Penny
Operation testbed	• Carry out interoperability testing with libraries		Nov 30-Feb 15	Bill Jason Penny Index Data

Work Area	July04	Aug04	Sep04	Oct04	Nov04	Dec04	Jan05	Feb05
Interop Testing								

3. Project Timeline

The project currently has an eight month duration. All work for the project should be completed by February 28, 2005. The letter approving the extension arrived in the second week of June. Substantial work on the project will begin on July 1, 2004 (allowing time for subcontracts to be issued and staffing to be confirmed). The following gives a high-level view of carrying out the project work.

Work Area	July04	Aug04	Sep04	Oct04	Nov04	Dec04	Jan05	Feb05
MARC DB								
MARC Records								
Test Searches								
Testing Scripts								
Indexing Policies								
Testing Validation								
Analysis & Reporting								
Interop Testing								

Appendix B: Z39.50 Interoperability Testing Framework for Online Library Catalogs Using Radioactive MARC Records

U.S. Federal Institute of Museum and Library Services
National Leadership Grant, Phase 2

A Radioactive MARC Record Approach for Interoperability Testing

Z39.50 Interoperability Testing Framework for Online Library Catalogs Using Radioactive MARC Records

William E. Moen, Ph.D.
<wemoen@unt.edu>
Principal Investigator

School of Library and Information Sciences
Texas Center for Digital Knowledge
University of North Texas
Denton, TX 76203

June 2004

Z39.50 Interoperability Testing Framework for Online Library Catalogs Using Radioactive MARC Records

Introduction

In a first phase of the Z39.50 Interoperability Testbed, a large dataset of MARC records was used. In this work, we are exploring how a set of special, diagnostic MARC records can be developed and used to identify interoperability problems between a Z39.50 client and a Z39.50 server providing access to a database of bibliographic records supporting the search and retrieval functions of an online library catalog. We refer to these special, diagnostic records as radioactive MARC records. This document describes a framework for interoperability testing using these radioactive MARC records. It discusses the various components and identifies the tasks related to developing and implementing the components.

Search Requirements To Be Tested

The Z39.50 profiles for library application (Bath and the U.S. National) specify searches that Z39.50 clients and Z39.50 servers (and the underlying information retrieval system) should support if vendors' systems are conformant with the profiles. Each profile specifies levels of conformance, where each level requires the support of a set of searches. For this work, we will base our searches on the U.S. National Profile for Library Applications, approved by NISO as an American National Standard in 2003. Searches defined in conformance levels 0 and 1 will be tested. The following is a list of these searches. For details on the Z39.50 attribute combinations for each search, refer to the profile.

Level 1 Searches
Author Search – Keyword
Title Search – Keyword
Subject Search – Keyword
Any Search – Keyword

Level 2 Searches
Author Search – Keyword with Right Truncation
Author Search – Exact Match
Author Search – First Words in Field
Author Search – First Characters in Field
Title Search – Keyword with Right Truncation
Title Search – Exact Match
Title Search – First Words in Field
Title Search – First Characters in Field
Subject Search – Keyword with Right Truncation
Subject Search – Exact Match
Subject Search – First Words in Field
Subject Search – First Characters in Field
Any Search – Keyword with Right Truncation
Standard Identifier Search
ISBN Search
ISSN Search
Remote System Record Number Search
Date of Publication Search
Language Search
Format of Material Search – Keyword

These searches span a range of access points (e.g., author, title, subject, ISBN, ISSN, etc.), address precision oriented searches (e.g., exact match, first words in fields, first characters in fields) and recall oriented searches (e.g., keyword, keyword with right truncation, phrase with right truncation), and searches that refine results (e.g., limiting by date, format of material, language).

Test Searches

The searching requirements will inform the development of test searches. One or more test searches may be necessary to adequately test the search types listed above. Each test search will define the type of search, the search term, and the expected results to be retrieved from the system being tested. Searches will be sent from the testbed client to ensure that appropriate Z39.50 attribute combinations are used to characterize the search term. The attribute combinations are defined in the U.S. National Profile.

The search term in the test searches will be aligned with specific data from specific fields/subfields in the radioactive MARC record. The retrieved records for each test search should be one or more of the radioactive MARC records, assuming the system being tested is in conformance with the U.S. National Profile.

The Radioactive MARC Records

One of the key research activities in this project is the development of a set of radioactive MARC records. This is particularly challenging because:

- MARC provides for rich content designation (e.g., various types of title information can be recorded in many different areas of the record)
- MARC records represent different formats of materials (e.g., books, computer files, etc.), and different formats may require the use of specific content designation not used in other formats

The number of required radioactive MARC records is not known at this time, but we anticipate creating at least one MARC record for each of the following formats of materials:

- Books, Pamphlets, and Printed Sheets
- Cartographic Materials
- Electronic Resources
- Continuing Resources
- Manuscripts (including manuscript collections)
- Music (Notated and manuscript music)
- Sound Recordings (musical and non-musical)
- Motion pictures and video-recordings (including digital and non-digital)
- Graphic materials (includes mixed materials, with or without archival control)
- Three Dimensional Artifacts and Realia

Multiple records for each format may be necessary to provide adequate coverage for the anticipated interoperability tests.

The MARC records will use the fields/subfields appropriate to each of these formats (since some MARC fields/subfields only apply to specific formats of materials). In each of the records, there will be patterns of alpha/numeric characters encoded in appropriate fields/subfields. The specifications for the character strings and how they will be generated to code into the records will be determined in discussions with project staff. An example of the concept follows:

The 245$a (title information) for two records might look like the following:

245 $a**Xysos sifys asdsai-aduus**

245 $a**Yszaqs sifys asdsai-aduus**

A title keyword search with the search term “sifys” should retrieve both of these records. A title exact match for the term “Xysos sifys asdsai-aduus” should only retrieve the first record.

Indexing Policies

During the first phase of the Z39.50 Interoperability Testbed Project, we developed indexing guidelines for several of the profile-defined searches (i.e., author, title, and subject keyword searches). When developing these guidelines, we identified approximately 500 subfields that could potentially store relevant data.

Since the testing planned for the current work encompasses searches beyond the keyword searches used in the first phase, it will be necessary to produce indexing guidelines for the additional searches. The indexing guidelines will be implemented in the Z-Interop Z39.50 server and underlying information retrieval system reference implementation. This is necessary to ensure baseline information about searching and retrieval behavior prior to conducting interoperability testing with other systems.

Testing Procedures

The intention of this work is to develop a set of automatic procedures to send the test searches and retrieve results. In the first phase of the project, sending the test searches was manually done by project staff. Automatic procedures can increase the efficiency of interoperability testing. These procedures should enable an increased number of test searches to be used. This will be important to assess the indexing policies in operation on the systems that will be tested. For example, we know that title information can be stored in different MARC subfields. The radioactive MARC records will have different patterns of characters in each of those subfields, and we can send individual searches with the specific string of characters that are stored in each of the subfields to see if the record is returned. If it is, we can conclude:

- The Z39.50 server is processing the search properly and sending it to the information retrieval system
- The information retrieval system does index the particular subfield.

The automated testing procedures will also assist in the management and analysis of results to reduce the manual effort that was required in the first phase of the testbed project. For example, the procedures should record any Z39.50 server diagnostic messages to understand if the server does not support particular types of searches; the procedures should record the search sent and the record retrieved; the procedures should enable compilation of results in an easy to use format for generating reports.

Availability of Radioactive MARC Records

For systems that choose to go through interoperability testing, the radioactive MARC records must be available for access and loading into the local system. The records will be stored on a password protected website from which system managers will be able to download the records. The records will be in conformant MARC 21 format, and system managers should be able to put the records into normal batch loading or other input queues.

Analysis and Reporting

Upon completion of interoperability testing, the results will be analyzed using manual and automatic methods. Wherever possible, automatic analysis should be used to streamline the process of analysis. Interoperability testing reports should include the following:

- Assessment of the level of Z39.50 interoperability
- Assessment of search functionality available
- Assessment of indexing policies.

Completed reports should be easy to understand by the managers of the system tested, and should enable them to take specific actions that might improve interoperability.

Tools and Resources Required

A number of tools will be required to set up and operate the testbed using the radioactive MARC records. The section describes anticipated tools and resources.

Database of MARC Content Designation

Because of the rich content designation available in MARC for bibliographic records, and because some of the content designation is intended for use when describing specific formats of materials, we will create a MARC Content Designation Database (MARC DB). This database will store information from the MARC 21 document and include the following:

- Field tags
- Indicator values, names, and definitions associated with field tags
- Subfield codes, names, and definitions associated with field tags
- Format applicable for each subfield code.

The MARC DB will enable queries to identify all appropriate content designation for the 10 formats of materials discussed above. In addition, the MARC DB will be used when generating reports of testing results to efficiently pull content designation names and definitions to make the reports easier to understand.

MARC Record Creation Tool

To create the radioactive MARC records, we will use a basic MARC record creation tool or cataloging tool. The tool must allow the use of all available MARC content designation (either through a template or by editing). It would be best to use a tool that provides basic validation of MARC input. If possible, the tool should allow import from a CSV or other form of text file to get data into the record and then produce a conformant MARC 21 record.

Radioactive MARC Records

A set of specialized, diagnostic MARC records will be created for use in interoperability testing.

Z39.50 Client for Testing

A Z-Interop Z39.50 client will be used to send the test queries. The client must be conformant with Levels 0 and 1 of the U.S. National Profile.

Z39.50 Server and Information Retrieval System Reference Implementation

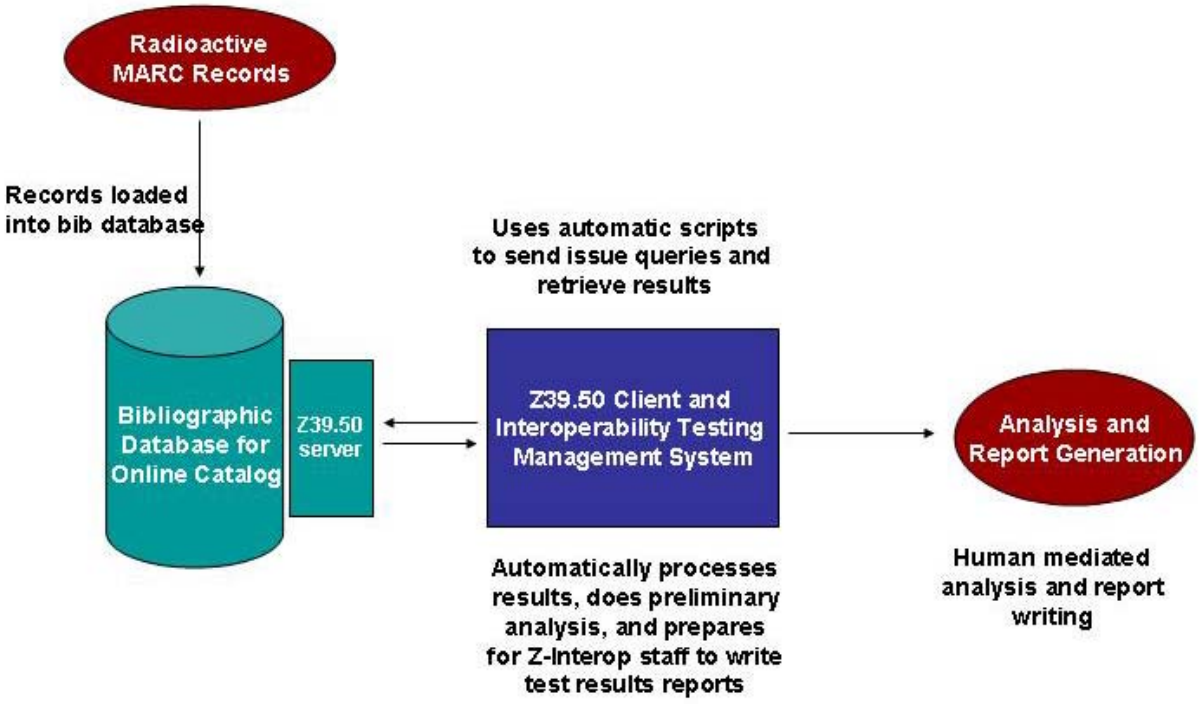
To ensure the reliability of all aspects of the interoperability testing, we will use a reference implementation of a Z39.50 server and information retrieval system. Current plans are to use the existing Z-Interop reference implementation (the SIRSI Unicorn system). Current and new indexing guidelines will inform the indexing policies for the reference implementation. The radioactive MARC records will be loaded into the system and made available to searching. Searching and retrieval procedures will be tested and verified.

Automatic Scripts for Testing and Reporting

Scripts will be used to send test searches from the Z-Interop client to systems.

Visual Overview of Radioactive MARC Record Interoperability Testing

The basic interoperability testing framework can be graphically illustrated to highlight key components and processes involved as in the following.



Summary

This document is intended to initiate and guide the work for developing an approach for interoperability testing using radioactive MARC records. It has identified the components for interoperability testing and some of the tools and resources needed. Each of these components and processes will be described in more detail in separate documents. Additionally, a work plan document will identify the work areas, activities, and tasks to further guide the work and provide project management oversight for the work.

Appendix C: Radioactive MARC Records Specifications

Research, Test & Improve

Zinterop

The Z39.50 Interoperability Testbed

U.S. Federal Institute of Museum and Library Services
National Leadership Grant, Phase 2

A Radioactive MARC Record Approach for Interoperability Testing

Radioactive MARC Records Specifications

William E. Moen, Ph.D.
<wemoen@unt.edu>
Principal Investigator

School of Library and Information Sciences
Texas Center for Digital Knowledge
University of North Texas
Denton, TX 76203

January 2005

Radioactive MARC Records Specifications

1. Introduction

For the Z-Interop2 Project, a set of special diagnostic MARC records (radioactive or RadMARC records) for conducting interoperability testing are required. The document, *Z39.50 Interoperability Testing Framework for Online Library Catalogs Using Radioactive MARC Records* describes an overview of the interoperability testing approach being used in Z-Interop2. Other preliminary documents have identified search requirements and token requirements for the data values of MARC fields and subfields.

Two Z390.50 profiles listed below provide the specifications that are the basis for the test searches used in the interoperability testing:

- ANSO/NISO Z39.89, The U.S. National Z39.50 Profile for Library Applications, <http://www.niso.org/standards/resources/Z39_89final.pdf>
- Bath Profile: An International Z39.50 Specification for Library Applications and Resource Discovery (Release 2.0) <http://www.collectionscanada.ca/bath/tp-bath2-e.htm>

Searches defined at Level 0 and Level 1 require appropriate RadMARC records for the test searches. In addition, different types of records (as indicated in the MARC Leader/06) are needed since systems may index MARC fields/subfields differently depending on the type of record. Finally, the RadMARC records need to be clearly identified as to the type of searching they are intended to assess, and therefore some version information about each record should be included in the record.

This document provides the preliminary specifications for the different RadMARC records to be created for use in the Z-Interop2 interoperability testbed. Experience with these records may result in revisions to the specifications.

2. Types of Records

MARC records can describe different types and formats of bibliographic materials. MARC Leader/06 indicates the type of record (i.e., the information object type being described by the MARC record). The *Anglo-American Cataloguing Rules* (AACR), the standard for descriptive cataloging, specifies rules for describing different types of materials. MARC Leader/07 indicates the bibliographic level of the record. The table below shows the values of the MARC Leader/06, Leader/07, and the AACR categories of materials to show the complexity of coding and labeling for what the MARC records describe.

Leader/06 Code	Semantics	AACR Categories of Materials
a	Language material	Books, Pamphlets, and Printed Sheets
c	Notated music	Music (Notated and manuscript music)
d	Manuscript notated music	Music (Notated and manuscript music)
e	Cartographic material	Cartographic Materials
f	Manuscript cartographic material	Cartographic Materials
g	Projected medium	Motion pictures and video-recordings (including digital and non-digital)
i	Nonmusical sound recording	Sound Recordings (musical and non-musical)
j	Musical sound recording	Sound Recordings (musical and non-musical)
k	Two-dimensional nonprojectable graphic	
m	Computer file	Electronic Resources
o	Kit	
p	Mixed material	Graphic materials (includes mixed materials, with or without archival control)

Leader/06 Code	Semantics	AACR Categories of Materials
r	Three-dimensional artifact or naturally occurring object	Three Dimensional Artifacts and Realia
t	Manuscript language material	Manuscripts (including manuscript collections)
Leader/07 Code	**Semantics**	**AACR Categories of Materials**
a	Monographic component part	
b	Serial component part	
c	Collection	
d	Subunit	
i	Integrating resource	
m	Monograph/item	
s	Serial	Continuing Resources

The coding of Type of Record in MARC Leader/06 is not aligned directly with the 10 format types of information objects as addressed by AACR. In the proposal for extension to the Z-Interop Project, we proposed to deal with the formats of material as addressed by AACR. These are found in the third column of the table above. In some cases, two code values in the Leader/06 are addressed by the same AACR format of material. Additionally, the Leader/06 doesn't indicate if the material is a serial publication (or Continuing Resource). For this, the Leader/07 – Bibliographic Level indicates serial with a value of s.

To summarize, the types of records or materials for which RadMARC records will be created will address those in the table below, which also provides the values for the Leader/06 and Leader/07 as appropriate.

Category of Material Described by RadMARC record	Leader/06 Value	Leader/07 Value
Books, Pamphlets, and Printed Sheets	a	
Continuing Resources	a	s
Music (Notated and manuscript music)	c	
Cartographic Materials	e	
Motion pictures and video-recordings (including digital and non-digital)	g	
Sound Recordings (musical and non-musical)	j	
Electronic Resources	m	
Graphic materials (includes mixed materials, with or without archival control)	p	
Three Dimensional Artifacts and Realia	r	
Manuscripts (including manuscript collections)	t	

While there will likely be commonality of many fields across the types of records, and in the indexable fields to support the profile-defined searches, we will create at least one instance of a RadMARC record for each of the 10 types of records.

3. Record Identification and Version Information

Each RadMARC record will be uniquely identified and each version of the record will be noted. The following methods for identifying and describing each record within the record will be used.

3.1 Use of the 001

MARC 001 is the Control Number field, a non-repeatable field with the following definition:

> The control number assigned by the organization creating, using, or distributing the record. The MARC code for the organization is contained in field 003 (Control Number Identifier)

This is a standard field in MARC records. In some systems, when a record from some other cataloging source is loaded into the bibliographic database, the number in the 001 is moved to the 035. The 035 is the System Control Number field, a repeatable field with the following definition:

> A control number of a system other than the one whose control number is contained in field 001 (Control Number), field 010 (Library of Congress Control Number) or field 016 (National Bibliographic Agency Control Number).

Not all systems move the number from the 001, especially if it is an OCLC number. (Once we get the RadMARC records created, we will contact OCLC to see if they would be interested in allowing us to put these records into the WorldCat database, at which time they would receive a true OCLC number.)

For the initial set of RadMARC records, the Control Number will have a coded structure as follows:

- Characters 1-10: Project signature for the record; value = UNTRadMARC
- Characters 11-13: Sequential number with left padding; value begins at 001 and continues sequentially for subsequent records.

An example of this is: UNTRadMARC001, UNTRadMARC002, etc.

3.2 Use of the 040

MARC 040 is the Cataloging Source field, a non-repeatable field with the following definition:

> The MARC code for or the name of the organization(s) that created the original bibliographic record, assigned MARC content designation and transcribed the record into machine-readable form, or modified (except for the addition of holdings symbols) an existing MARC record. These data and the code in 008/39 (Cataloging source) specify the parties responsible for the bibliographic record.

MARC 040 defines several subfields, and the RadMARC records will use only subfield a, defined as Original Cataloging Agency. For the initial set of RadMARC records, the Cataloging Source will have a code to represent the Z39.50 Interoperability Testbed Project in the form of:

- ZinteropUNT

When loaded into a remote system, this value is not likely to be changed and provides one more place to indicate the source of these special MARC records.

3.3 Use of the 583

To record version and other information about each RADMARC records, we propose to use MARC 583, the Action Note field, a repeatable field with the following definition:

> A copy-specific field that contains information about processing, reference, and preservation actions.

MARC 583 has a sufficient number of subfield structures to enable the recording of the following types of information:

- Signature
- Identifier
- Version
- Description
- Scheme

- Threshold
- Record Creator

This, however, means that we will revise the meaning of the 583 subfield codes as follows:

- a - Action
  - **RadMARC semantics: <signature>** a constant string -- that is always the same for every RadMARC record.
  - Value = RadMARC
- b - Action identification
  - **RadMARC semantics: <identifier>** any string that is "as unique as possible". Will use domain name for project <www.unt.edu/zinterop> and a sequential number (which is the same numeric value as in the 001).
  - Value = www.unt.edu/zinterop/001, www.unt.edu/zinterop/002, etc.
- d - Action interval
  - **RadMARC semantics: <version>** is an integer, beginning at 1 for the initial version of the record and incremented by 1 in each subsequent version.
- e - Contingency for action
  - **RadMARC semantics: <scheme>** identifies uniquely the type of search or other parameters describing the record in the context of interoperability testing.
  - Value will be defined for the RadMARC records produced by the Z-Interop2 Project.
- i - Method of action
  - **RadMARC semantics: <scheme dependent information>** includes scheme dependent information regarding this particular record.
  - Value will be defined for the RadMARC records produced by the Z-Interop2 Project.
- $k - Action agent (R)
  - **RadMARC semantics: <record creator>** contains the name of the person creating the RadMARC record
  - Value = name of person creating record
- x - Nonpublic note
  - **RadMARC semantics: <description>** is arbitrary human-readable text describing the provenance and intent of the record, and perhaps even including things like a version-control log describing its history.
  - Value will be standardized to the extent possible along the lines of: "This is a specially created test record the Z-Interop2 Project under the direction of the Texas Center for Digital Knowledge at the University of North Texas. Contact Dr. William E. Moen via email for information about this project at <wemoen@unt.edu>. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of record that was created to support test searches for author, title, subject, and any profile-defined searches, where the threshold of occurrence of the indexable content designation being populated in the record is 1 for occurrences of more than 100,000 times, and this is the first version of this record."

3.4 Example of Use

The following shows an example of a RadMARC record using the above specifications for identification, version, and other information:

001 UNTRadMARC001
040 **$a** ZinteropUNT
583 **$a** RadMARC **$b** www.unt.edu/zinterop/001 **$d** 1 **$e** ATS **$i** 1 **$k** JungWon Yoon **$x** This is a specially created test record the Z-Interop2 Project under the direction of the Texas Center for Digital Knowledge at the University of North Texas. Contact Dr. William E. Moen via email for information about this project at <wemoen@unt.edu>. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of record that was created to support test searches for author, title, subject, and any profile-defined searches, where the threshold of occurrence of the

indexable content designation being populated in the record is 19, for the 19 most commonly occurring indexable author, title, and subject fields. This is the first version of this record.

4. Format of the Tokens

The core of the RadMARC records is specially designed tokens that populate each indexable field/subfield. A token is a string of characters that has a specific structure and semantics that will serve as “words” or other data values in specific RadMARC field/subfield. A field/subfield may have a sequence of tokens. The following structure results in content-rich token elements.

- A single alpha character for left-hand padding.
  - Value = r
- A single alpha character to indicate the format of the material being described or type of record
  - Value = Selected values as defined in MARC Leader/06 – Type of Record or the Leader/07 – Bibliographic Level to represent the 10 categories of materials discussed in Section 2 as follows:
    - Leader/06 **a** - Language material
      Includes printed, microform, and electronic language material.
    - Leader/06 **c** - Notated music
      Includes microform and electronic notated music.
    - Leader/06 **e** - Cartographic material
      Includes maps, atlases, globes, digital maps, and other cartographic items.
    - Leader/06 **g** - Projected medium
      Examples include: motion pictures, video recordings (including digital video), filmstrips, slides, transparencies, or material specifically designed for projection.
    - Leader/06 **j** - Musical sound recording
      Examples include: phonodiscs, compact discs, or cassette tapes.
    - Leader/06 **m** - Computer file
      Includes the following classes of electronic resources: computer software (including programs, games, fonts), numeric data, computer-oriented multimedia, online systems or services. For these classes of materials, if there is a significant aspect that causes it to fall into another Leader/06 category, the code for that significant aspect is used instead of code m (e.g., vector data that is cartographic is not coded as numeric but cartographic). Other classes of electronic resources are coded for their most significant aspect (e.g., language material, graphic, cartographic material, sound, music, moving image). In case of doubt or if the most significant aspect cannot be determined, consider the item a computer file.
    - Leader/06 **p** - Mixed material
      Indicates that there are significant materials in two or more forms that are usually related by virtue of their having been accumulated by or about a person or body. Includes archival fonds and manuscript collections of mixed forms of materials, such as text, photographs, and sound recordings.
    - Leader/06 **r** - Three-dimensional artifact or naturally occurring object
      Includes man-made objects, such as models, dioramas, games, puzzles, simulations, sculptures and other three-dimensional art works and their reproductions, exhibits, machines, clothing, toys, and stitchery, and naturally occurring objects, such as microscope specimens and other specimens mounted for viewing.
    - Leader/06 **t** - Manuscript language material
    - Leader/07 **s** - Indicating a serial or continuing resource
- Three numbers indicating the Field Tag
  - Value = Defined in MARC 21 specifications
- A single integer to indicate number of occurrence the Field Tag
  - Value = Sequential number starting with 1
- A single alpha character to indicate the Subfield Code

    - Value = Defined in MARC 21 specifications
    - Note: Even though some subfields in specific fields can repeat, the current token format does not allow specification of the occurrence number of the subfield within the field.
- A single integer indicating the offset within subfield
    - Value = Use the following scheme: 1=first token in subfield, 2=second token in subfield; 3= third token in subfield, etc.
- A single alpha character for right-hand padding
    - Value = r

And example token element to show this structure is: `ra2451a1r`. We can parse it as:

- r - Left-hand padding
- a - Type of record -- this is a Monograph-type record
- 245 - Field code
- 1 – First occurrence of field in record
- a - Subfield code
- 1 - Offset within subfield -- 1=first token in subfield
- r - Right-hand padding

If there was a second instance of this field (in this example the 245), the token element for the second occurrence would be `rm2452a11r`. An example of a complete MARC 245 with a sequence of tokens in selected subfields follows:

```
245 $a rm2451a1r rm2451a2r rm2451a3r $b rm2451b1r rm2451b2r rm2451b3r $c
rm2451c1r rm2451c2r
```

In any given subfield, a sequence of tokens will not exceed three tokens. The limitation allows the appropriate interoperability testing of various phrase-oriented searches as defined in the profiles without adding undue complexity to the RadMARC records.

5. RadMARC Records Required to Test Author, Title, Subject, and Any Searches

A number of sets of RadMARC records are required that will support the testing of profiled-defined author, title, subject, and any searches at Levels 0 and 1. In each set, there will be 10 separate records reflecting the different types of records or materials described. Each record in each set will contain the exact same tokens except for the second character of the token element that will reflect the type of record or material. For example, the token element in a 245 field for:

- A Books, Pamphlets, and Printed Sheets Language Material type record will be:
    - ra2451a11r ra2451a21r ra2451a31r (which uses the Leader/06 code “a”)
- A Music (Notated and manuscript music) type record will be:
    - rc2451a11r rc2451a21r rc2451a31r (which used the Leader/06 code “c”)
- A Continuing Resource type record will be:
    - rs2451a11r rs2451a21r rs2451a31r (which used the Leader/07 code “s”)

Our goal is to create several sets of RadMARC records that will have increasing levels of content designation (i.e., the number of fields and subfields) included. The level of content designation for each set of records will be based on the following specifications (which may evolve based on experience):

- The threshold of occurrence of indexable fields that resulted from previous analysis undertaken in the first phase of the Z-Interop Project. The threshold of occurrence will determine which content designation gets populated in each set of records. The threshold of occurrence sets will be defined as follows as indicated by Threshold of Occurrence Code held in the 583 $i. :

    - 1 = The 19 most commonly occurring indexable fields for author, title, and subject-related data as identified in the previous research. See Appendix A for a list of these fields/subfields.
    - 2 =The indexable fields for author, title, and subject-related data occurring at least 1,000 times as identified in the previous research; approximate number of these fields/subfields is: **XXX**. See Appendix B for a list of these fields/subfields.
    - 3 =All indexable fields for author, title, and subject-related data as identified in Z-Interop Project indexing guidelines; approximate number of these fields/subfields is: **XXX**. See the Z-Interop Indexing Guidelines for the complete list of these fields/subfields.
- The indexable fields as listed in the Z-Interop Indexing Guidelines that are recommended by the Program for Cooperative Cataloging Bibliographic Core (BIBCO) Record Elements; approximate number of these fields/subfields is: **XXX**. See Appendix C for a list of these fields/subfields.

With this arrangement, there will be four sets of RadMARC records, with each set containing 10 records.

6. RadMARC Records Required to Test Standard Identifier Number Searches

[to be completed]

7. RadMARC Records Required to Test Qualifying Searches

[to be completed]

8. Record Creation

Project staff will create RadMARC records according to the above specifications using MARC creation software that produces reliable and valid MARC records on export. Initially, the software to be used is MARC Magician, a PC-based application that provides the functionality needed.

Priority of creation should be on the sets of records at Threshold of Occurrence Code 1 and the BIBCO specifications. The time needed to create each set of records should be recorded to get an understanding of the level of effort needed.

9. Testing and Editing the RadMARC Records in the Z-Interop Server

The RadMARC records will be loaded into the Z-Interop Testbed Server. The RadMARC records can be checked by searching the Z-Interop server with the Bookwhere client. The Bookwhere client allows us to set the attribute combination for each of the profile-defined searches.

MARC Magician, unfortunately, places the phrase [Demo Record] in each subfield that is populated. This phrase will need to be edited out of each RadMARC record. The Sirsi Workflows client can be used to easily edit the RadMARC record. Once the records have been edited, the Index Data test harness can be used to run a trial set of searches and generate reports.

10. Creating a File of RadMARC Records for Other Targets

Once the testing has been done on the sets of RadMARC records, and any issues or problems with the records have been resolved or edited away, each set of RadMARC records will be exported from the Z-

Interop testbed server to create the authoritative file of RadMARC records that can be used by other targets to load into their bibliographic database for interoperability testing.

Appendix A: List of Content Designation for Threshold of Occurrence Code 1

Based on previous analysis, the Z-Interop Project, Phase 1, identified the following 19 fields/subfields from the Z-Interop Indexing Guidelines as the most commonly occurring. This set of content designation will be used to create the most minimal versions of the RadMARC records.

# Occurrences	Marc 21 Field	Subfield	Description	Index
602,362	650	a	Subject added entry Topical Term Subfield a = Topical term or geographic name as entry element	Subject
419,641	245	a	Title Statement Subfield a = Title	Title
329,796	245	c	Title Statement Subfield c = statement of responsibility	Author
326,867	650	x	Subject added entry Topical Term Subfield x = General subdivision	Subject
318,692	100	a	Main entry Personal Name Subfield a = personal name	Author
231,459	650	z	Subject added entry Topical Term Subfield z = Geographic subdivision	Subject
176,916	700	a	Added entry Personal Name Subfield a = personal name	Author
169,178	245	b	Title Statement Subfield b = Remainder of title	Title
149,540	100	d	Main entry Personal Name Subfield d = dates associated with a name	Author
118,647	651	x	Subject added entry Geographic Name Subfield x = General subdivision	Subject
113,050	651	a	Subject added entry Geographic Name Subfield a = Geographic name	Subject
83,607	650	v	Subject added entry Topical Term Subfield v = Form subdivision	Subject
74,606	700	d	Added entry Personal Name Subfield d = dates associated with a name	Author
69,636	600	a	Subject added entry Personal Name Subfield a = personal name	Subject
66,375	710	a	Added entry Corporate Name Subfield a = corporate name or jurisdiction name	Author
64,433	440	a	Series Statement Added Entry Title Subfield a = title	Title
62,853	490	a	Series Statement Subfield a = Series statement	Title
56,229	600	d	Subject added entry Personal Name Subfield d = dates associated with a name	Subject
55,311	653	a	Index Term Uncontrolled Subfield a = the term	Subject

Appendix D: RadMARC Record Set 1: Ten Records

001	UNTRadMARC001
040	$a ZinteropUNT
100	$a ra1001a1r, ra1001a2r, $d ra1001d1r.
245	$a ra2451a1r ra2451a2r ra2451a3r : $b ra2451b1r ra2451b2r ra2451b3r / $c ra2451c1r ra2451c2r ra2451c3r.
440	$a ra4401a1r ra4401a2r ra4401a3r
490	$a ra4901a1r ra4901a2r ra4901a3r
583	$a RadMARC $b www.unt.edu/zinterop/001 $d 1 $e ATS $i 1 $k JungWon Yoon $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E. Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 19, for the 19 most commonly occurring indexable author, title, and subject fields discovered in a separate analysis. This is the first version of this record.
600	$a ra6001a1r ra6001a2r, $d ra6001a1r.
650	$a ra6501a1r ra6501a2r ra6501a3r $x ra6501x1r $v ra6501v1r $z ra6501z1r.
651	$a ra6511a1r ra6511a2r $x ra6511x1r.
653	$a ra6531a1r ra6531a2r ra6531a3r
700	$a ra7001a1r ra7001a2r, $d ra7001d1r.
710	$a ra7101a1r ra7101a2r.

001	UNTRadMARC001
040	$a ZinteropUNT
100	$a rc1001a1r, rc1001a2r, $d rc1001d1r.
245	$a rc2451a1r rc2451a2r rc2451a3r : $b rc2451b1r rc2451b2r rc2451b3r / $c rc2451c1r rc2451c2r rc2451c3r.
440	$a rc4401a1r rc4401a2r rc4401a3r
490	$a rc4901a1r rc4901a2r rc4901a3r
583	$a RadMARC $b www.unt.edu/zinterop/001 $d 1 $e ATS $i 1 $k JungWon Yoon $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E. Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 19, for the 19 most commonly occurring indexable author, title, and subject fields discovered in a separate analysis. This is the first version of this record.
600	$a rc6001a1r rc6001a2r, $d rc6001a1r.
650	$a rc6501a1r rc6501a2r rc6501a3r $x rc6501x1r $v rc6501v1r $z rc6501z1r.
651	$a rc6511a1r rc6511a2r $x rc6511x1r.
653	$a rc6531a1r rc6531a2r rc6531a3r
700	$a rc7001a1r rc7001a2r, $d rc7001d1r.
710	$a rc7101a1r rc7101a2r.

001	UNTRadMARC001
040	$a ZinteropUNT
100	$a re1001a1r, re1001a2r, $d re1001d1r.
245	$a re2451a1r re2451a2r re2451a3r : $b re2451b1r re2451b2r re2451b3r / $c re2451c1r re2451c2r re2451c3r.
440	$a re4401a1r re4401a2r re4401a3r
490	$a re4901a1r re4901a2r re4901a3r
583	$a RadMARC $b www.unt.edu/zinterop/001 $d 1 $e ATS $i 1 $k JungWon Yoon $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E. Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 19, for the 19 most commonly occurring indexable author, title, and subject fields discovered in a separate analysis. This is the first version of this record.
600	$a re6001a1r re6001a2r, $d re6001a1r.
650	$a re6501a1r re6501a2r re6501a3r $x re6501x1r $v re6501v1r $z re6501z1r.
651	$a re6511a1r re6511a2r $x re6511x1r.
653	$a re6531a1r re6531a2r re6531a3r
700	$a re7001a1r re7001a2r, $d re7001d1r.
710	$a re7101a1r re7101a2r.

001	UNTRadMARC001
040	$a ZinteropUNT
100	$a rg1001a1r, rg1001a2r, $d rg1001d1r.
245	$a rg2451a1r rg2451a2r rg2451a3r : $b rg2451b1r rg2451b2r rg2451b3r / $c rg2451c1r rg2451c2r rg2451c3r.
440	$a rg4401a1r rg4401a2r rg4401a3r
490	$a rg4901a1r rg4901a2r rg4901a3r
583	$a RadMARC $b www.unt.edu/zinterop/001 $d 1 $e ATS $i 1 $k JungWon Yoon $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E. Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 19, for the 19 most commonly occurring indexable author, title, and subject fields discovered in a separate analysis. This is the first version of this record.
600	$a rg6001a1r rg6001a2r, $d rg6001a1r.
650	$a rg6501a1r rg6501a2r rg6501a3r $x rg6501x1r $v rg6501v1r $z rg6501z1r.
651	$a rg6511a1r rg6511a2r $x rg6511x1r.
653	$a rg6531a1r rg6531a2r rg6531a3r
700	$a rg7001a1r rg7001a2r, $d rg7001d1r.
710	$a rg7101a1r rg7101a2r.

001	UNTRadMARC001
040	$a ZinteropUNT
100	$a rj1001a1r, rj1001a2r, $d rj1001d1r.
245	$a rj2451a1r rj2451a2r rj2451a3r : $b rj2451b1r rj2451b2r rj2451b3r / $c rj2451c1r rj2451c2r rj2451c3r.
440	$a rj4401a1r rj4401a2r rj4401a3r
490	$a rj4901a1r rj4901a2r rj4901a3r
583	$a RadMARC $b www.unt.edu/zinterop/001 $d 1 $e ATS $i 1 $k JungWon Yoon $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E. Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 19, for the 19 most commonly occurring indexable author, title, and subject fields discovered in a separate analysis. This is the first version of this record.
600	$a rj6001a1r rj6001a2r, $d rj6001a1r.
650	$a rj6501a1r rj6501a2r rj6501a3r $x rj6501x1r $v rj6501v1r $z rj6501z1r.
651	$a rj6511a1r rj6511a2r $x rj6511x1r.
653	$a rj6531a1r rj6531a2r rj6531a3r
700	$a rj7001a1r rj7001a2r, $d rj7001d1r.
710	$a rj7101a1r rj7101a2r.

001	UNTRadMARC001
040	$a ZinteropUNT
100	$a rm1001a1r, rm1001a2r, $d rm1001d1r.
245	$a rm2451a1r rm2451a2r rm2451a3r : $b rm2451b1r rm2451b2r rm2451b3r / $c rm2451c1r rm2451c2r rm2451c3r.
440	$a rm4401a1r rm4401a2r rm4401a3r
490	$a rm4901a1r rm4901a2r rm4901a3r
583	$a RadMARC $b www.unt.edu/zinterop/001 $d 1 $e ATS $i 1 $k JungWon Yoon $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E. Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 19, for the 19 most commonly occurring indexable author, title, and subject fields discovered in a separate analysis. This is the first version of this record.
600	$a rm6001a1r rm6001a2r, $d rm6001a1r.
650	$a rm6501a1r rm6501a2r rm6501a3r $x rm6501x1r $v rm6501v1r $z rm6501z1r.
651	$a rm6511a1r rm6511a2r $x rm6511x1r.
653	$a rm6531a1r rm6531a2r rm6531a3r
700	$a rm7001a1r rm7001a2r, $d rm7001d1r.
710	$a rm7101a1r rm7101a2r.

001	UNTRadMARC001
040	$a ZinteropUNT
100	$a rp1001a1r, rp1001a2r, $d rp1001d1r.
245	$a rp2451a1r rp2451a2r rp2451a3r : $b rp2451b1r rp2451b2r rp2451b3r / $c rp2451c1r rp2451c2r rp2451c3r.
440	$a rp4401a1r rp4401a2r rp4401a3r
490	$a rp4901a1r rp4901a2r rp4901a3r
583	$a RadMARC $b www.unt.edu/zinterop/001 $d 1 $e ATS $i 1 $k JungWon Yoon $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E. Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 19, for the 19 most commonly occurring indexable author, title, and subject fields discovered in a separate analysis. This is the first version of this record.
600	$a rp6001a1r rp6001a2r, $d rp6001a1r.
650	$a rp6501a1r rp6501a2r rp6501a3r $x rp6501x1r $v rp6501v1r $z rp6501z1r.
651	$a rp6511a1r rp6511a2r $x rp6511x1r.
653	$a rp6531a1r rp6531a2r rp6531a3r
700	$a rp7001a1r rp7001a2r, $d rp7001d1r.
710	$a rp7101a1r rp7101a2r.

001	UNTRadMARC001
040	$a ZinteropUNT
100	$a rr1001a1r, rr1001a2r, $d rr1001d1r.
245	$a rr2451a1r rr2451a2r rr2451a3r : $b rr2451b1r rr2451b2r rr2451b3r / $c rr2451c1r rr2451c2r rr2451c3r.
440	$a rr4401a1r rr4401a2r rr4401a3r
490	$a rr4901a1r rr4901a2r rr4901a3r
583	$a RadMARC $b www.unt.edu/zinterop/001 $d 1 $e ATS $i 1 $k JungWon Yoon $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E. Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 19, for the 19 most commonly occurring indexable author, title, and subject fields discovered in a separate analysis. This is the first version of this record.
600	$a rr6001a1r rr6001a2r, $d rr6001a1r.
650	$a rr6501a1r rr6501a2r rr6501a3r $x rr6501x1r $v rr6501v1r $z rr6501z1r.
651	$a rr6511a1r rr6511a2r $x rr6511x1r.
653	$a rr6531a1r rr6531a2r rr6531a3r
700	$a rr7001a1r rr7001a2r, $d rr7001d1r.
710	$a rr7101a1r rr7101a2r.

001	UNTRadMARC001
040	$a ZinteropUNT
100	$a rs1001a1r, rs1001a2r, $d rs1001d1r.
245	$a rs2451a1r rs2451a2r rs2451a3r : $b rs2451b1r rs2451b2r rs2451b3r / $c rs2451c1r rs2451c2r rs2451c3r.
440	$a rs4401a1r rs4401a2r rs4401a3r
490	$a rs4901a1r rs4901a2r rs4901a3r
583	$a RadMARC $b www.unt.edu/zinterop/001 $d 1 $e ATS $i 1 $k JungWon Yoon $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E. Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 19, for the 19 most commonly occurring indexable author, title, and subject fields discovered in a separate analysis. This is the first version of this record.
600	$a rs6001a1r rs6001a2r, $d rs6001a1r.
650	$a rs6501a1r rs6501a2r rs6501a3r $x rs6501x1r $v rs6501v1r $z rs6501z1r.
651	$a rs6511a1r rs6511a2r $x rs6511x1r.
653	$a rs6531a1r rs6531a2r rs6531a3r
700	$a rs7001a1r rs7001a2r, $d rs7001d1r.
710	$a rs7101a1r rs7101a2r.

001	UNTRadMARC001
040	$a ZinteropUNT
100	$a rt1001a1r, rt1001a2r, $d rt1001d1r.
245	$a rt2451a1r rt2451a2r rt2451a3r : $b rt2451b1r rt2451b2r rt2451b3r / $c rt2451c1r rt2451c2r rt2451c3r.
440	$a rt4401a1r rt4401a2r rt4401a3r
490	$a rt4901a1r rt4901a2r rt4901a3r
583	$a RadMARC $b www.unt.edu/zinterop/001 $d 1 $e ATS $i 1 $k JungWon Yoon $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E. Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 19, for the 19 most commonly occurring indexable author, title, and subject fields discovered in a separate analysis. This is the first version of this record.
600	$a rt6001a1r rt6001a2r, $d rt6001a1r.
650	$a rt6501a1r rt6501a2r rt6501a3r $x rt6501x1r $v rt6501v1r $z rt6501z1r.
651	$a rt6511a1r rt6511a2r $x rt6511x1r.
653	$a rt6531a1r rt6531a2r rt6531a3r
700	$a rt7001a1r rt7001a2r, $d rt7001d1r.
710	$a rt7101a1r rt7101a2r.

Appendix E: RadMARC Record Set 2: Four Records

100 & 700 (Personal name)

001	UNTRadMARC201
040	$a ZinteropUNT
100	$a ra1001a1r, ra1001a2r, $c ra1001c1r, $d ra1001d1r, $e ra1001e1r $q ra1001q1r ra1001q2r.
210	$a ra2101a1r
222	$a ra2221a1r
240	$a ra2401a1r ra2401a2r ra2401a3r. $f ra2401f1r. $k ra2401k1r. $l ra2401l1r, $m ra2401m1r. $n ra2401n1r, $r ra2401r1r
245	$a ra2451a1r ra2451a2r ra2451a3r : $b ra2451b1r ra2451b2r ra2451b3r / $c ra2451c1r ra2451c2r ra2451c3r. $n ra2451n1r, $p ra2451p1r ra2451p2r.
246	$a ra2461a1r ra2461a2r $f ra2461f1r
440	$a ra4401a1r ra4401a2r ra4401a3r $p ra4401p1r ra4401p2r ra4401p3r
490	$a ra4901a1r ra4901a2r ra4901a3r ; $v ra4901v1r
505	$a ra5051a1r ra5051a2r $g ra5051g1r $r ra5051r1r $t ra5051t1r
508	$a ra5081a1r.
511	$a ra5111a1r.
583	$a RadMARC $b www.unt.edu/zinterop/201 $d 1 $e ATS $i 2 $k JungWon Yoon $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E. Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 112, for the occurring 1,000 or more times in indexable author, title, and subject fields discovered in a separate analysis. (100 (personal name) & 700 fields --> 96 fields) This is the first version of this record.
600	$a ra6001a1r ra6001a2r $b ra6001b1r, $c ra6001c1r $q ra6001q1r ra6001q2r. $t ra6001t1r $v ra6001v1r $x ra6001x1r $z ra6001z1r, $d ra6001d1r.
610	$a ra6101a1r ra6101a2r. $b ra6101b1r $v ra6101v1r $x ra6101x1r $z ra6101z1r.
630	$a ra6301a1r. $p ra6301p1r ra6301p2r $x ra6301x1r.
650	$v ra6501v1r $x ra6501x1r $y ra6501y1r $z ra6501z1r. $2 ra650121r
651	$a ra6511a1r ra6511a2r $x ra6511x1r $v ra6511v1r $y ra6511y1r $z ra6511z1r ra6511z2r.
653	$a ra6531a1r ra6531a2r ra6531a3r
655	$a ra6551a1r ra6551a2r. $2 ra655121r
700	$a ra7001a1r ra7001a2r, $d ra7001d1r, $c ra7001c1r, $e ra7001e1r $m ra7001m1r $n ra7001n1r $p ra7001p1r $q ra7001q1r ra7001q2r $r ra7001r1r. $t ra7001t1r ra7001t2r ra7001t3r. $4 ra700141r
740	$a ra7401a1r ra7401a2r ra7401a3r.
752	$a ra7521a1r $b ra7521b1r $d ra7521d1r.
776	$c ra7761c1r $t ra7761t1r
780	$a ra7801a1r $t ra7801t1r
785	$t ra7851t1r
787	$t ra7871t1r
800	$a ra8001a1r, $d ra8001d1r. $t ra8001t1r $v ra8001v1r.
810	$a ra8101a1r. $b ra8101b1r $t ra8101t1r $v ra8101v1r.
830	$a ra8301a1r $v ra8301v1r. $n ra8301n1r, $p ra8301p1r.

110 & 710 (corporate name)

001	UNTRadMARC202
040	$a ZinteropUNT
110	$a ra1101a1r ra1101a2r ra1101a3r. $b ra1101b1r ra1101b2r.
210	$a ra2101a1r
222	$a ra2221a1r
240	$a ra2401a1r ra2401a2r ra2401a3r. $f ra2401f1r. $k ra2401k1r. $l ra2401l1r, $m ra2401m1r. $n ra2401n1r, $r ra2401r1r
245	$a ra2451a1r ra2451a2r ra2451a3r : $b ra2451b1r ra2451b2r ra2451b3r / $c ra2451c1r ra2451c2r ra2451c3r. $n ra2451n1r, $p ra2451p1r ra2451p2r.
246	$a ra2461a1r ra2461a2r $f ra2461f1r
440	$a ra4401a1r ra4401a2r ra4401a3r $p ra4401p1r ra4401p2r ra4401p3r
490	$a ra4901a1r ra4901a2r ra4901a3r ; $v ra4901v1r
505	$a ra5051a1r ra5051a2r $g ra5051g1r $r ra5051r1r $t ra5051t1r
508	$a ra5081a1r.
511	$a ra5111a1r.
583	$a RadMARC $b www.unt.edu/zinterop/202 $d 1 $e ATS $i 2 $k JungWon Yoon $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E. Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 112, for the occurring 1,000 or more times in indexable author, title, and subject fields discovered in a separate analysis. (110 (Corporate name) & 710 fields --> 86 fields) This is the first version of this record.
600	$a ra6001a1r ra6001a2r $b ra6001b1r, $c ra6001c1r $q ra6001q1r ra6001q2r. $t ra6001t1r $v ra6001v1r $x ra6001x1r $z ra6001z1r, $d ra6001d1r.
610	$a ra6101a1r ra6101a2r. $b ra6101b1r $v ra6101v1r $x ra6101x1r $z ra6101z1r.
630	$a ra6301a1r. $p ra6301p1r ra6301p2r $x ra6301x1r.
650	$v ra6501v1r $x ra6501x1r $y ra6501y1r $z ra6501z1r. $2 ra650121r
651	$a ra6511a1r ra6511a2r $x ra6511x1r $v ra6511v1r $y ra6511y1r $z ra6511z1r ra6511z2r.
653	$a ra6531a1r ra6531a2r ra6531a3r
655	$a ra6551a1r ra6551a2r. $2 ra655121r
710	$a ra7101a1r ra7101a2r. $b ra7101b1r ra7101b2r $t ra7101t1r. $4 ra710141r
740	$a ra7401a1r ra7401a2r ra7401a3r.
752	$a ra7521a1r $b ra7521b1r $d ra7521d1r.
776	$c ra7761c1r $t ra7761t1r
780	$a ra7801a1r $t ra7801t1r
785	$t ra7851t1r
787	$t ra7871t1r
800	$a ra8001a1r, $d ra8001d1r. $t ra8001t1r $v ra8001v1r.
810	$a ra8101a1r. $b ra8101b1r $t ra8101t1r $v ra8101v1r.
830	$a ra8301a1r $v ra8301v1r. $n ra8301n1r, $p ra8301p1r.

111 & 711 (meeting name)

001	UNTRadMARC203
040	$a ZinteropUNT
111	$a ra1111a1r ra1111a2r ra1111a3r $n ra1111n1r : $d ra1111d1r : $c ra1111c1r ra1111c2r.
210	$a ra2101a1r
222	$a ra2221a1r
240	$a ra2401a1r ra2401a2r ra2401a3r. $f ra2401f1r. $k ra2401k1r. $l ra2401l1r, $m ra2401m1r. $n ra2401n1r, $r ra2401r1r
245	$a ra2451a1r ra2451a2r ra2451a3r : $b ra2451b1r ra2451b2r ra2451b3r / $c ra2451c1r ra2451c2r ra2451c3r. $n ra2451n1r, $p ra2451p1r ra2451p2r.
246	$a ra2461a1r ra2461a2r $f ra2461f1r
440	$a ra4401a1r ra4401a2r ra4401a3r $p ra4401p1r ra4401p2r ra4401p3r
490	$a ra4901a1r ra4901a2r ra4901a3r ; $v ra4901v1r
505	$a ra5051a1r ra5051a2r $g ra5051g1r $r ra5051r1r $t ra5051t1r
508	$a ra5081a1r.
511	$a ra5111a1r.
583	$a RadMARC $b www.unt.edu/zinterop/203 $d 1 $e ATS $i 2 $k JungWon Yoon $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E. Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 112, for the occurring 1,000 or more times in indexable author, title, and subject fields discovered in a separate analysis. (111 (meeting name) & 711 fields --> 87 fields)This is the first version of this record.
600	$a ra6001a1r ra6001a2r $b ra6001b1r, $c ra6001c1r $q ra6001q1r ra6001q2r. $t ra6001t1r $v ra6001v1r $x ra6001x1r $z ra6001z1r, $d ra6001d1r.
610	$a ra6101a1r ra6101a2r. $b ra6101b1r $v ra6101v1r $x ra6101x1r $z ra6101z1r.
630	$a ra6301a1r. $p ra6301p1r ra6301p2r $x ra6301x1r.
650	$v ra6501v1r $x ra6501x1r $y ra6501y1r $z ra6501z1r. $2 ra650121r
651	$a ra6511a1r ra6511a2r $x ra6511x1r $v ra6511v1r $y ra6511y1r $z ra6511z1r ra6511z2r.
653	$a ra6531a1r ra6531a2r ra6531a3r
655	$a ra6551a1r ra6551a2r. $2 ra655121r
711	$a ra7111a1r ra7111a2r ra7111a3r $d ra7111d1r : $c ra7111c1r ra7111c2r.
740	$a ra7401a1r ra7401a2r ra7401a3r.
752	$a ra7521a1r $b ra7521b1r $d ra7521d1r.
776	$c ra7761c1r $t ra7761t1r
780	$a ra7801a1r $t ra7801t1r
785	$t ra7851t1r
787	$t ra7871t1r
800	$a ra8001a1r, $d ra8001d1r. $t ra8001t1r $v ra8001v1r.
810	$a ra8101a1r. $b ra8101b1r $t ra8101t1r $v ra8101v1r.
830	$a ra8301a1r $v ra8301v1r. $n ra8301n1r, $p ra8301p1r.

130 & 730 (uniform title)

001	UNTRadMARC204
040	$a ZinteropUNT
130	$a ra1301a1r. $l ra1301l1r.
210	$a ra2101a1r
222	$a ra2221a1r
240	$a ra2401a1r ra2401a2r ra2401a3r. $f ra2401f1r. $k ra2401k1r. $l ra2401l1r, $m ra2401m1r. $n ra2401n1r, $r ra2401r1r
245	$a ra2451a1r ra2451a2r ra2451a3r : $b ra2451b1r ra2451b2r ra2451b3r / $c ra2451c1r ra2451c2r ra2451c3r. $n ra2451n1r, $p ra2451p1r ra2451p2r.
246	$a ra2461a1r ra2461a2r $f ra2461f1r
440	$a ra4401a1r ra4401a2r ra4401a3r $p ra4401p1r ra4401p2r ra4401p3r
490	$a ra4901a1r ra4901a2r ra4901a3r ; $v ra4901v1r
505	$a ra5051a1r ra5051a2r $g ra5051g1r $r ra5051r1r $t ra5051t1r
508	$a ra5081a1r.
511	$a ra5111a1r.
583	$a RadMARC $b www.unt.edu/zinterop/204 $d 1 $e ATS $i 2 $k JungWon Yoon $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E. Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 112, for the occurring 1,000 or more times in indexable author, title, and subject fields discovered in a separate analysis. (130 (uniform title) & 730 fields --> 83 fields). This is the first version of this record.
600	$a ra6001a1r ra6001a2r $b ra6001b1r, $c ra6001c1r $q ra6001q1r ra6001q2r. $t ra6001t1r $v ra6001v1r $x ra6001x1r $z ra6001z1r, $d ra6001d1r.
610	$a ra6101a1r ra6101a2r. $b ra6101b1r $v ra6101v1r $x ra6101x1r $z ra6101z1r.
630	$a ra6301a1r. $p ra6301p1r ra6301p2r $x ra6301x1r.
650	$v ra6501v1r $x ra6501x1r $y ra6501y1r $z ra6501z1r. $2 ra650121r
651	$a ra6511a1r ra6511a2r $x ra6511x1r $v ra6511v1r $y ra6511y1r $z ra6511z1r ra6511z2r.
653	$a ra6531a1r ra6531a2r ra6531a3r
655	$a ra6551a1r ra6551a2r. $2 ra655121r
730	$a ra7301a1r ra7301a2r.
740	$a ra7401a1r ra7401a2r ra7401a3r.
752	$a ra7521a1r $b ra7521b1r $d ra7521d1r.
776	$c ra7761c1r $t ra7761t1r
780	$a ra7801a1r $t ra7801t1r
785	$t ra7851t1r
787	$t ra7871t1r
800	$a ra8001a1r, $d ra8001d1r. $t ra8001t1r $v ra8001v1r.
810	$a ra8101a1r. $b ra8101b1r $t ra8101t1r $v ra8101v1r.
830	$a ra8301a1r $v ra8301v1r. $n ra8301n1r, $p ra8301p1r.

Appendix F: RadMARC Record Creation Procedures

RadMARC Record Creation Procedures

RadMARC records were populated using the following steps:

1. Raw RadMARC Record Creation (using MARC Magician)

   Using MARC Magician, a raw record was created. Each token of field/subfield was typed in according to the Token specification. Figure 1 is the data entry screen, and Figure 2 is the sample record exported from MARC Magician. MARC Magician automatically generated [Demo Record] for each subfield of the exported record, because this project used the trial version of MARC Magician. Thus, the raw record needed to be manipulated for deleting [Demo Record].

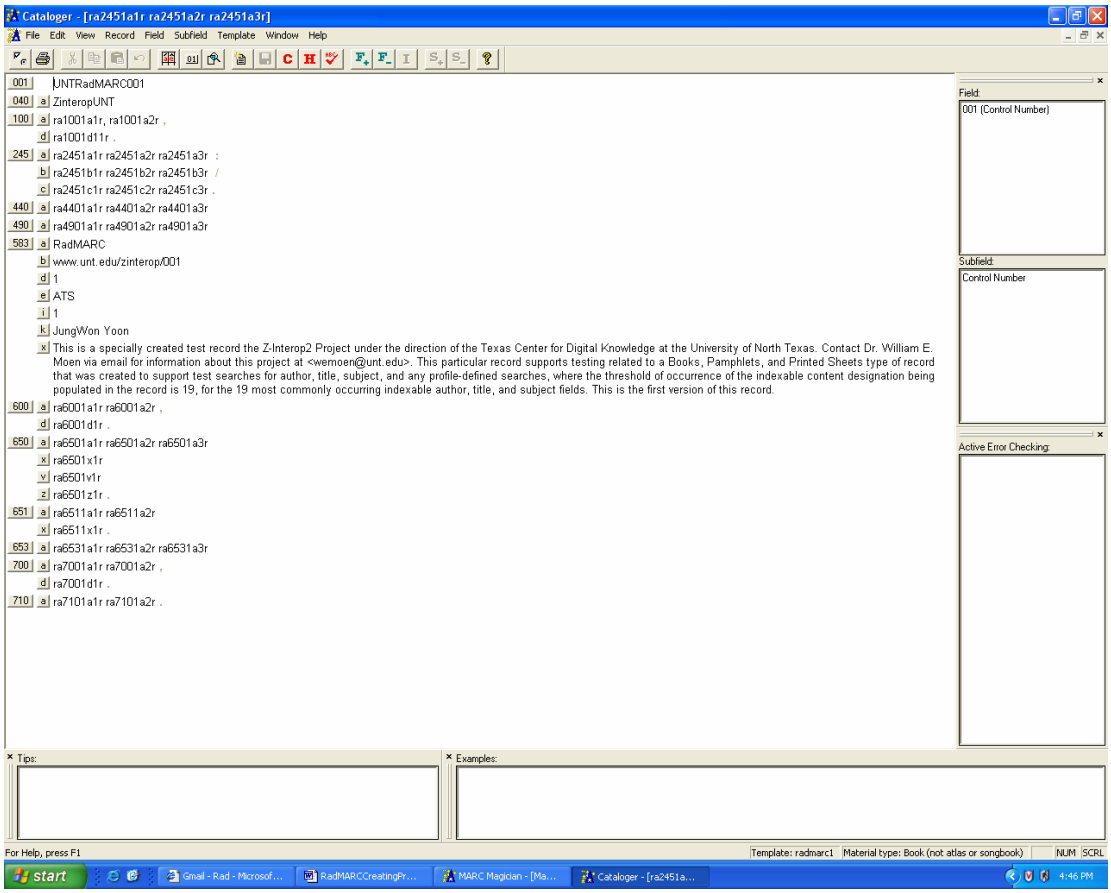


Figure 1. Data Entry Screen

001	UNTRadMARC001 [Demo Record]
040	$a ZinteropUNT [Demo Record]
100	$a ra1001a1r, ra1001a2r, [Demo Record] $d ra1001d1r. [Demo Record]
245	$a ra2451a1r ra2451a2r ra2451a3r : [Demo Record] $b ra2451b1r ra2451b2r ra2451b3r / [Demo Record] $c ra2451c1r ra2451c2r ra2451c3r. [Demo Record]
440	$a ra4401a1r ra4401a2r ra4401a3r [Demo Record]
490	$a ra4901a1r ra4901a2r ra4901a3r [Demo Record]
583	$a RadMARC [Demo Record] $b www.unt.edu/zinterop/001 [Demo Record] $d 1 [Demo Record] $e ATS [Demo Record] $i 1 [Demo Record] $k JungWon Yoon [Demo Record] $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E.

	Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 19, for the 19 most commonly occurring indexable author, title, and subject fields discovered in a separate analysis. This is the first version of this record. [Demo Record]
600	$a ra6001a1r ra6001a2r, [Demo Record] $d ra6001a1r. [Demo Record]
650	$a ra6501a1r ra6501a2r ra6501a3r [Demo Record] $x ra6501x1r [Demo Record] $v ra6501v1r $z ra6501z1r. [Demo Record]
651	$a ra6511a1r ra6511a2r [Demo Record] $x ra6511x1r. [Demo Record]
653	$a ra6531a1r ra6531a2r ra6531a3r [Demo Record]
700	$a ra7001a1r ra7001a2r, [Demo Record] $d ra7001d1r. [Demo Record]
710	$a ra7101a1r ra7101a2r. [Demo Record]

Figure 2 Sample Raw RadMARC Record

2. RadMARC Record Manipulation (using SIRSI Workflow)

To manipulate the raw record, the project staff used the SIRSI Workflow, part of the reference implementation online catalog system of the Z-Interop Project. The raw record was uploaded to the SIRSI system (refer to "*Documentation for the Z-Interop Implementation of the SIRSI Unicorn System*" for uploading procedure), and every occurrence of the "[Demo Record]" was deleted from the raw record through a manual process. The modified record is the final version of RadMARC Record. The populated RadMARC Record was exported using BookWhere.

3. Different Formats of RadMARC Records Creation (using Text Editor)

Once a RadMARC Record was populated, the record was duplicated for populating RadMARC Records supporting different formats. For effectiveness, the 'replace' function of the text editor was used for changing first two elements of each token. For example, once a token for Language Material was populated, the record was duplicated and replaced the first two elements, as 'ra' with 'rm'.

Appendix G: Net::Z3950::RadioMARC: Extension for Testing Z39.50 Servers

Net::Z3950::RadioMARC: Extension for Testing Z39.50 Servers

Mike Taylor
<mike@indexdata.com>
Index Data
Denmark

Note

The current version of **Net::Z3950::RadioMARC** module, along with the current version of this documentation, is available on CPAN: Comprehensive Perl Archive Network, <http://www.cpan.org/>. This extension to Perl is free software; you can redistribute it and/or modify it under the same terms as Perl itself, either Perl version 5.8.3 or, at your option, any later version of Perl 5 you may have available.

Synopsis

```
use Net::Z3950::RadioMARC;

 $t = new Net::Z3950::RadioMARC();
 $t->set(host => 'z3950.loc.gov', port => '7090', db => 'voyager');
 $t->set(delay => 3);
 $t->add("filename.marc");
 $t->test('@attr 1=4 01245a01', { ok => '245$a is searchable as 1=4',
                                  notfound => 'This server is broken' });
 # -- or --
 set host => 'z3950.loc.gov', port => '7090', db => 'voyager';
 set delay => 3;
 add "filename.marc";
 test '@attr 1=4 01245a01', { ok => '245$a is searchable as 1=4',
                              notfound => 'This server is broken' };
```

Description

This module provides the harness in which test-scripts can be written for detecting the presence of a ``radioactive MARC record'' in a Z39.50-compliant database, and determining how that database indexes the record. Its key provision is the **`test()`**method, which runs a search for some well-known term that is known to occur in a ``radioactive'' record, and generates different output dependent on whether the record is found or not.

This module may be used in two different ways: the first approach is to use a rigorous object-oriented syntax in which a test-harness method is explicitly created, and methods are invoked on it. The other is a simpler syntax in which a test-harness object is transparently created behind the scenes when it's first needed, and subsequently referenced by function calls. These two styles are exemplified by the two code-fragments in the synopsis above.

For most purposes, the simple syntax will be preferable. The object-oriented syntax is useful primarily when it is necessary for a single script to run tests against two or more different databases.

Methods

new()

```
 $testHarness = new Net::Z3950::RadioMARC();
```

Creates a new test-harness for checking the searchability of ``radioactive MARC records'' in a database available via Z39.50. There are no arguments; the new object is returned.

Before the new test-harness can be used by the **test()**method, at least its `host` property must be set, and often its `port`, `db` and `format` as well. See the documentation for the **set()** method for details on how this is done and what the argument mean.

It is not necessary to explicitly create a test-harness object in order to use this module. See the description above of the ``simple syntax'' approach, in which a test-harness object is implicitly created.

set()

```
 $t->set(host => 'z3950.loc.gov', port => '7090', db => 'voyager');
 $t->set(delay => 3);
 # -- or --
 set host => 'z3950.loc.gov', port => '7090', db => 'voyager';
 set delay => 3;
```

Sets one or more properties of the specified test-harness, or of the implicit test-harness if none is specified (i.e. when the simple syntax is used instead of the object-oriented syntax).

Each pair of arguments is taken to be a pair consisting of a property name and the corresponding new value. It is an error to provide an odd number of arguments.

The following properties are defined.

`host` [no default]
: The name or IP address of the Internet host where the Z39.50 server to be tested resides. The connection to the host will be forged when the first test-case is run by means of the **test()**method.

`port` [default: 210]
: The port number where the Z39.50 server to be tested resides on the specified host.

db [default: "Default"]
: The name of the database to be searched.

`format` [default: "USMARC"]
: The record syntax to be used when fetching records from result sets to be compared with the known radioactive records.

`delay` [default: 1]
: The delay, in seconds, between issuing one test search and the next. This delay is a courtesy to the server, to prevent it from being overrun by a large test-script.

`messages` [default: an empty hash]
: A reference to a hash which maps test status values to message templates. These are used to generate the reporting output for tests, depending on the status returned by the test, except when overridden by the messages specified for that particular test (see below).

The interpretation of message templates is described in the documentation of the **`test()`** method. Status values for which no message template is provided (i.e. all status value initially) are not reported at all, except for the status **`fail`** which is reported using a simple, explicit default format.

`verbosity` **[default: 1]**

The level at which debugging and other output is emitted. This may be set to any integer; all messages at the nominated value *and less* are emitted. These messages, by level, are as follows:

0 No debugging output at all.

1 Messages are emitted if a query is used that either matches none of the records that have been nominated using the **`add()`** method, or matches more than one of these records (in which case the first one to have been added is used).

Level 1 is the default, since these messages are arguably reporting configuration errors, whereas the higher levels generate chit-chit that is probably only useful for debugging.

2 Messages indicating when the number of hits a query generates in the remote database is not one, as expected.

3 Messages indicating how many hits each query generated in the remote database being tested, however many there are.

4 Messages showing each query before it is tested.

`report` **[default: 1]**

A Boolean indicating whether or not reporting output (as opposed to debugging output) should be emitted for tests. This should nearly always be true: the principal use of this property is as an additional, one-shot property used for a single test, like this:

```
 ($status, $errmsg, $addinfo) = test '@attr 1=4 fruit', { report => 0
};
```

which allows the test-script to explicitly check the status and make whatever choices it deems appropriate without side-effects.

`identityField` **[no default]**

An indication of what MARC field or subfield is taken to convey the identity of a record for the purposes of comparison. If a record in a result-set has the same identity-field value as the radioactive record being tested, then they are regarded as the same record.

It may take the form *tag* for control fields (for example **`001`** to specify the local identifier) or *tag*$*subfield* (for example **`245$a`** to specify the title field).

Multiple candidate identity fields may be specified, separated by commas, like this: **`100,035$a`**. In this case, each such candidate subfield is tried in turn, and the first one that exists in both records being compared is used.

If no identity field is specified, then two records are considered to be the same only if they are byte-for-byte identical.

It is an error to try to set a property other than those described here.

add()

```
$t->add("filename.marc");
# -- or --
add "filename.marc";
```

Adds one or more MARC records to the set that are to be tested for. Records are loaded from the file whose name is specified. Any number of records may be added to the test-set, but using many such records may be self-defeating, since then the radioactive tokens to be searched for are less likely to be unique.

Behind the scenes, this module builds an inverted index of all the words occurring in all the subfields of all the non-control fields in all the records that are added. This is used when **test()** is called to identify which of the **add()**ed records is the one that should be retrieved from the server.

add() returns a list of opaque tokens representing the newly added records. These tokens may be passed as the `record` parameter into the **test()** method to indicate explicitly which of the test-set records a particular query is intended to find.

dumpindex()

```
$t->dumpindex();
```

Dumps to standard output the inverted index generated for the MARC records have been added to the test-set by the **add()** method.

Never call this method.

test()

```
$t->test('@attr 1=4 01245a01', { ok => '245$a is searchable as 1=4',
                                 notfound => 'This server is broken',
                                 record => $token });
$t->test('@attr 1=4 thrickbrutton');
# -- or --
test '@attr 1=4 01245a01', { ok => '245$a is searchable as 1=4',
                             notfound => 'This server is broken',
                             record => $token };
test '@attr 1=4 thrickbrutton';
```

Runs a single test against the server that has been nominated for the specified test-harness. The first argument is a query in PQF (Prefix Query Format) as described in the YAZ manual at http://indexdata.com/yaz/doc/tools.tkl#PQF and the second (optional) is a reference to hash of parameters, some of which are used for mapping status values to message templates.

The query is analyzed to see which of the test-set records it should find. For maximally indicative results, it should match exactly one such record, no more, no less. If it matches more than one, the first one is used for the subsequent matching process: that is, the one that occurred earliest in MARC file that first **add()**ed to the test-set. If the parameter `record` is provided, then its value is use as the opaque token of the test-set record to be used and the query is not used for this purpose.

The query is submitted to the server, and returns some number of hit-set records. Again, the most significant test results are obtained when there is exactly one such record.

Each of the candidate hits is compared with the chosen test-set record to see whether there is match or not - that is, whether the search retrieved the nominated radioactive record.

The result of this process is that a status is generated, being one of the following short strings:

ok The query succeeded, and a record in the hit-set was the same as the record chosen from the test-set.

notfound The query succeeded, but no record in the hit-set was the same as the record chosen from the test-set. This may occur for several different reasons: because the query matched no record in the test-set (which is probably a configuration error); because it matched no record in the database (which means either that the radioactive record is not in the database or that it is not indexed in the way being tested for) or because it found some record in the database, but none of them is the one that was expected (for the same reasons).

fail The query could not be executed, or the records could not be fetched.

The **test()** method returns a triple, (**$status, $errmsg, $addinfo**), with **$errmsg** being the human-readable string corresponding to the BIB-1 diagnostic code returned by the server in the case of an error, and **$addinfo** being any additional information returned by the server along with such a diagnostic.

If the `report` property of the test-harness is true (as it is by default), then a report is emitted describing the outcome of the test. Under some circumstances, it is useful to inhibit this behavior by setting `report` false and testing the explicitly returned values instead.

The reporting output is generated from a template. The template is found by looking up the status of the test in the hash-reference argument, if this is supplied. If it is not supplied, or if the relevant element is missing, it is looked up in the hash that is the value of the **message** property. If the relevant element is not in this hash either, a default template is used for **notfound** and **fail** tests, but NO OUTPUT AT ALL is emitted for **ok** test. This makes it possible to write silent-on-success test scripts. If you want commentary on successful tests, then, you must explicitly specify an **ok** message template, either in the `message` property or in the hash-reference passed into **test()**.

Report-generating templates are strings which may contain the following escape sequences, which are substituted the appropriate values:

%{query}
The query that was run for this test.

%{status}
The status of the test.

%{errmsg}
The human-readable error message returned from the test, if any.

%{addinfo}
The additional information returned from the test, if any.

Appendix H: The *bathtest.pl* Perl Script for Interoperability Testing

Note: The following contains a Perl script used in the context of the Z-Interop2 Project for interoperability testing. Comments (noted by the # sign) throughout the script explain the functions and operations of various lines of the script.

```
#!/usr/bin/perl -w

# $Id: bathtest.pl,v 1.10 2005/03/03 14:59:34 mike Exp $

# This is an example script that queries a Z39.50 server and reports back
# about its behavior based on the results that are returned. It uses the
# RadioMARC perl module written by Mike Taylor (view its documentation at
# [http://www.miketaylor.org.uk/tmp/radiomarc/RadioMARC.html]). The testing
# methods used in this script and the RadioMARC module follow the methods
# outlined by Dr. Bill Moen for a "radioactive" approach to testing Z39.50
# servers (see http://www.unt.edu/zinterop/ZInterop2/Documents/
# InteropTestingFramework20June2004.pdf and http://www.unt.edu/zinterop/
# ZInterop2/Documents/MARCdocsExtensionForRadMARCQueries2Dec2004.pdf for more
# information).
#
# Essentially, this module provides a "test harness" by which one can test
# a given Z server's behavior. To do this, one first builds a test set, which
# is comprised of one or more MARC records that one expects to be returned by
# one's queries. Next, one sends a query to the Z server, and the result of
# this query is compared to the test set. Based on this comparison, the test
# harness generates a status message: "ok" if a retrieved record matched a
# record in the test set; "notfound" if no record in the results matched any
# records in the test set; or "fail" if the query could not be executed. The
# RadioMARC module further allows one to create one's own customized messages
# based on these status messages.
#
# This example script illustrates some of the possibilities for a complete
# test of a Z server. It has been fully commented (read: over-commented) to
# show--without ambiguity--exactly how one uses the RadioMARC module to test
# a server.

# First, we give the typical Perl "use" statements, including
# "use Net::Z3950::RadioMARC", which is required to use the RadioMARC module:

use strict;
use warnings;
use Net::Z3950::RadioMARC;

# Now, we initialize variables and set up default values:

my $pattern = 'rmFFF1S1r'; # This represents a regular expression that
                           # describes the type of tokens we're using as the
                           # basis for retrieving our radioactive MARC
records.
my $combo; # Stores attribute combinations representing queries.

# $hosturl (below) stores the IP address, port, and database name of the Z
# server to be tested in the form "IP:PORT/DB".

my $hosturl = '129.120.92.237:210/INNOPAC';
#my $hosturl = 'research.lis.unt.edu:2200/zinterop';
#my $hosturl = 'localhost:8989/Default';
#my $hosturl = 'test:9999/Default';

# Below, $hosturl is broken up into its constituent parts using a regular
# expression: $host, $port, and $dbname.
```

```
my ($host, $port, $dbname) = $hosturl =~ /(.*):(.*)\/(.*)/;

# Next, we set the test harness defaults for this particular script using the
# "set" method:

set host => $host, port => $port, db => $dbname;
set delay => 1;
set identityField => '001,035$a';
set verbosity => 1;

# The next two variables represent attribute strings that will form the basis
# for keyword ($attributes_kw) and truncated ($attributes_kwt) searches.

my $attributes_kw = '@attr 2=3 @attr 3=3 @attr 4=2 @attr 5=100 @attr 6=1';
my $attributes_kwt = '@attr 2=3 @attr 3=3 @attr 4=2 @attr 5=1 @attr 6=1';

# Now we add a record to the test set using the "add" method. Again, this
# is a record that we expect our queries to the Z server to return.

add 'record3a.mrc';
#add 'RadMARCATS1';
#add 'record.mrc';

# Here begins the output of our test to the user. We output first a brief
# header describing the test.

print "Bath compliance test script\n\n";
print "Test date: " . localtime() . "\n";
print "Test target: $hosturl\n";

# The first test we perform should be the most basic test possible, just to
# ensure that the server is working properly. Below, we use the "test"
# subroutine to perform a search for a token present in the 245$a
(rm2451a1r).
# The 245$a, obviously, should be indexed by the Z server--if it is not, then
# the test should not continue because something is wrong. Thus, if the
# test does not return a status of "ok," then it should not continue. In this
# case, one should ascertain whether the Z server is broken or whether one
has
# neglected to set up the test properly (e.g., if a test-set has not been
# specified, if the correct record has not been loaded onto the server, or if
# the query sent to the server is not correct).

if (test('@attr 1=4 rm2451a1r', {notfound=>'Not Found!', ok=>''}) ne 'ok') {
  print "Test record not found in database -- unable to continue\n";
  exit 1;
}

# @types, defined below, is an important data structure that sets up the MARC
# fields/subfields that are going to be tested and HOW those fields are to be
# tested--i.e., what kind of searches should be directed at those fields,
# according to the Bath Profile. The array itself contains a list of hash
# references, each of which uses the following indexes: "name" holds the name
# of the Bath Profile search (in "human readable" form), "use" holds the
# appropriate use attribute for the given search type, and "fields" holds
# a reference to a list of strings that represent the subfields to be tested
```

```
# under that search type (in the format FFF$s). This data structure is the
key
# to the tests presented in the first half of this example script.

my @types = (
  {
    'name' => 'Author search (BP0.1).',
    'use' => 1003,
    'fields' => [
      '100$a', '100$d',
      '245$c',
      '700$a', '700$d',
      '710$a'
    ]
  },
  {
    'name' => 'Title search (BP0.2)',
    'use' => 4,
    'fields' => [
      '245$a', '245$b',
      '440$a',
      '490$a'
    ],
  },
  {
    'name' => 'Subject search (BP0.3)',
    'use' => 21,
    'fields' => [
      '600$a', '600$d',
      '650$a', '650$x', '650$v', '650$z',
      '653$a'
    ]
  }
);

# Next are a couple of subroutines defined that make life easier throughout
# this testing process.

# radtoken returns a valid RadioMARC token based on the $pattern (defined
# earlier) when given a FFF$s (e.g., 245$a). [print radtoken('245$a');] or
# [print radtoken '245$a';] would thus print "rm2451a1r".

sub radtoken {
  $_ = shift;
  my $ret = $pattern;

  my ($field, $subfield) = /(...)\$(.)/;
  $ret =~ s/FFF/$field/;
  $ret =~ s/S/$subfield/;
  return $ret;
}

# runtest, when given an attribute combination ($combo), a reference to a
list
# of FFF$s's ($list), and (optionally) a 't', which represents "truncation",
# will run through each FFF$s and query the server based on that subfield. If
# the query is successful, the script outputs "Search finds FFF$s";
```

```
# otherwise, it outputs "Search does NOT find FFF$s".

sub runtest {
  my $combo = shift;
  my $list = shift;
  my $trunc = shift;
  foreach (@{$list}) {
    my $tok = radtoken $_;
    if (defined($trunc) && $trunc eq 't') {
      $tok =~ s/.$//; # if this is a truncation search, strip last character
                      # of the given token
    }
    my $search = "$combo " . $tok;
    test $search, {
      ok=>"Search finds $_",
      notfound=>"Search does NOT find $_"
    };
  }
}

# Now we begin the testing.
#
# We start with the bath profile level 0 keyword searches, from 0.1 to 0.4.

if(1) {
print "Testing Level 0 keyword searching (BP0.1 to 0.4).\n\n";

# For each @type (i.e., for each type of BP search), we take two steps:
# first, we build the complete attribute combination by taking the "use"
# attribute specified in the @type structure and combining it with the other
# attributes necessary for a keyword search (held in the variable
# $attributes_kw). Secondly, we use the runtest subroutine to test each field
# belonging to a particular BP search type.

foreach (@types) {
  my $combo = "\@attr 1=" . $_->{'use'} . " $attributes_kw";
  print "Testing: " . $_->{name} . "\n\n";
  runtest $combo, $_->{fields};
  print "\n";
}

# To test the BP 0.4 (an "any" search), we must query based on each field
# given in the @type structure.

$combo = "\@attr 1=1016 $attributes_kw";
print "Testing: Any\n";
foreach (@types) {
  runtest $combo, $_->{fields};
}

# Next, we perform a search based on BP 1.1--author keyword search with right
# truncation. The query is built using the $attributes_kwt variable, which
# holds the appropriate attributes for a right truncation search, and a
series
# of tests are run based on the fields corresponding to an author search
# ($types[0]->{fields}).
```

```
print "\nAuthor search -- keyword with right truncation (BP1.1)\n\n";

runtest "\@attr 1=1003 $attributes_kwt", $types[0]->{fields}, 't';

# Here ends the section in which attribute combinations are pre-specified.
# The rest of the script represents a manual approach to testing, which is
# certainly another way to write test scripts using the RadioMARC module.
#
# Hopefully by now the rest of this script is self-explanatory. For the
# BP 1.2 search, shown below, different token combinations are tested and the
# output is tailored based on what is being tested. Have a look at how the
# rest of the BP searches (up to 1.13) are performed. Note that these tests
# could easily be automated, thus reducing the length of the script.

print "\nAuthor search -- exact match (BP1.2)\n\n";

$combo = '@attr 1=1003 @attr 2=3 @attr 3=1 @attr 4=1 @attr 5=100 @attr 6=3';

test "$combo {rm1001a1r, rm1001a2r}", {
  ok=>     '100$a with comma OK',
  notfound=>   '100$a with comma NOT FOUND'
};

test "$combo {rm1001a1r rm1001a2r}", {
  ok=>     '100$a without comma OK',
  notfound=>   '100$a without comma NOT FOUND'
};

test "$combo {rm1001a1r, rm1001a2r, rm1001d1r}", {
  ok=>     '100$a 100$d with comma OK',
  notfound=>   '100$a 100$d with comma NOT FOUND'
};

test "$combo {rm1001a1r rm1001a2r rm1001d1r}", {
  ok=>     '100$a 100$d without comma OK',
  notfound=>   '100$a 100$d without comma NOT FOUND'
};

test "$combo {rm7001a1r, rm7001a2r}", {
  ok=>     '700$a with comma OK',
  notfound=>   '700$a with comma NOT FOUND'
};

test "$combo {rm7001a1r rm7001a2r}", {
  ok=>     '700$a without comma OK',
  notfound=>   '700$a without comma NOT FOUND'
};

test "$combo {rm7001a1r, rm7001a2r, rm7001d1r}", {
  ok=>     '700$a 700$d with comma OK',
  notfound=>   '700$a 700$d with comma NOT FOUND'
};

test "$combo {rm7001a1r rm7001a2r rm7001d1r}", {
  ok=>     '700$a 700$d without comma OK',
  notfound=>   '700$a 700$d without comma NOT FOUND'
};
```

```
test "$combo {rm7101a1r rm7101a2r}", {
  ok=>     '710$a without comma OK',
  notfound=>   '710$a without comma NOT FOUND'
};

test "$combo {rm2451c1r rm2451c2r rm2451c3r}", {
  ok=>     '245$c without comma OK',
  notfound=>   '245$c without comma NOT FOUND'
};

print "\nAuthor search -- first words in field (BP 1.3).\n\n";

$combo = '@attr 1=1003 @attr 2=3 @attr 3=1 @attr 4=1 @attr 5=100 @attr 6=1';

test "$combo {rm1001a1r rm1001a2r}", {
  ok=>     '100$a without comma OK',
  notfound=>   '100$a without comma NOT FOUND'
};

test "$combo {rm1001a1r}", {
  ok=>     '100$a (partial) OK',
  notfound=>   '100$a (partial) NOT FOUND'
};

test "$combo {rm1001a1r rm1001a2r rm1001d1r}", {
  ok=>     '100$a 100$d without comma OK',
  notfound=>   '100$a 100$d without comma NOT FOUND'
};

test "$combo {rm7001a1r rm7001a2r}", {
  ok=>     '700$a without comma OK',
  notfound=>   '700$a without comma NOT FOUND'
};

test "$combo {rm7001a1r rm7001a2r rm7001d1r}", {
  ok=>     '700$a 700$d without comma OK',
  notfound=>   '700$a 700$d without comma NOT FOUND'
};

test "$combo {rm7101a1r rm7101a2r}", {
  ok=>     '710$a without comma OK',
  notfound=>   '710$a without comma NOT FOUND'
};

test "$combo {rm2451c1r rm2451c2r}", {
  ok=>     '245$c without comma OK',
  notfound=>   '245$c without comma NOT FOUND'
};

print "\nAuthor search -- first characters in field (BP 1.4).\n\n";

$combo = '@attr 1=1003 @attr 2=3 @attr 3=1 @attr 4=1 @attr 5=1 @attr 6=1';

test "$combo {rm1001a1r rm1001a21}", {
  ok=>     '100$a without comma OK',
  notfound=>   '100$a without comma NOT FOUND'
```

```
};

test "$combo {rm1001a11}", {
  ok=>     '100$a (partial) OK',
  notfound=>   '100$a (partial) NOT FOUND'
};

test "$combo {rm1001a1r rm1001a2r rm1001d11}", {
  ok=>     '100$a 100$d without comma OK',
  notfound=>   '100$a 100$d without comma NOT FOUND'
};

test "$combo {rm7001a1r rm7001a21}", {
  ok=>     '700$a without comma OK',
  notfound=>   '700$a without comma NOT FOUND'
};

test "$combo {rm7001a1r rm7001a2r rm7001d11}", {
  ok=>     '700$a 700$d without comma OK',
  notfound=>   '700$a 700$d without comma NOT FOUND'
};

test "$combo {rm7101a1r rm7101a21}", {
  ok=>     '710$a without comma OK',
  notfound=>   '710$a without comma NOT FOUND'
};

test "$combo {rm2451c1r rm2451c21}", {
  ok=>     '245$c without comma OK',
  notfound=>   '245$c without comma NOT FOUND'
};

print "\nTitle search -- keyword with right truncation (BP1.5)\n\n";

runtest "\@attr 1=4 $attributes_kwt", $types[1]->{fields}, 't';

print "\nTitle search -- Exact match (BP1.6).\n\n";

$combo = '@attr 1=4 @attr 2=3 @attr 3=1 @attr 4=1 @attr 5=100 @attr 6=3';

test "$combo {rm2451a1r rm2451a2r rm2451a3r}", {
  ok=>     '245$a OK',
  notfound=>   '245$a NOT FOUND'
};

test "$combo {rm2451a1r rm2451a2r rm2451a3r rm2451b1r rm2451b2r rm2451b3r}",
{
  ok=>     '245$a 245$b OK',
  notfound=>   '245$a 245$b NOT FOUND'
};

test "$combo {rm2451b1r rm2451a1r rm2451a2r rm2451a3r rm2451b1r rm2451b2r
rm2451b3r}", {
  ok=>     '!Server appears insensitive to word order (245$a 245$a)',
  notfound=>    'Reverse order test OK'
};
```

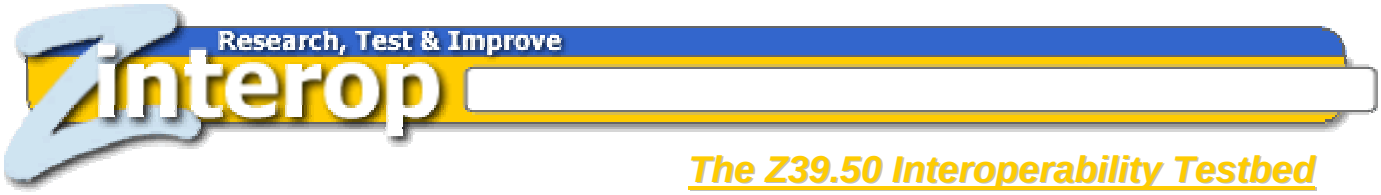
```
test "$combo {rm4401a1r rm4401a2r rm4401a3r}", {
  ok=>     '440$a OK',
  notfound=>   '440$a NOT FOUND'
};

test "$combo {rm4901a1r rm4901a2r rm4901a3r}", {
  ok=>     '490$a OK',
  notfound=>   '490$a NOT FOUND'
};

print "\nTitle search -- First words in field (BP1.7).\n\n";

$combo = '@attr 1=4 @attr 2=3 @attr 3=1 @attr 4=1 @attr 5=100 @attr 6=1';

test "$combo {rm2451a1r rm2451a2r rm2451a3r}", {
  ok=>     '245$a OK',
  notfound=>   '245$a NOT FOUND'
};

test "$combo {rm2451a1r rm2451a2r}", {
  ok=>     '245$a (partial) OK',
  notfound=>   '245$a (partial) NOT FOUND'
};

test "$combo {rm2451a1r rm2451a2r rm2451a3r rm2451b1r rm2451b2r}", {
  ok=>     '245$a 245$b (partial) OK',
  notfound=>   '245$a 245$b (partial) NOT FOUND'
};

test "$combo {rm4401a1r rm4401a2r}", {
  ok=>     '440$a (partial) OK',
  notfound=>   '440$a (partial) NOT FOUND'
};

test "$combo {rm4901a1r rm4901a2r}", {
  ok=>     '490$a (partial) OK',
  notfound=>   '490$a (partial) NOT FOUND'
};

print "\nTitle search -- First characters in field (BP1.8).\n\n";

$combo = '@attr 1=4 @attr 2=3 @attr 3=1 @attr 4=1 @attr 5=1 @attr 6=1';

test "$combo {rm2451a1r rm2451a2r rm2451a31}", {
  ok=>     '245$a OK',
  notfound=>   '245$a NOT FOUND'
};

test "$combo {rm2451a1r rm2451a21}", {
  ok=>     '245$a (partial) OK',
  notfound=>   '245$a (partial) NOT FOUND'
};

test "$combo {rm2451a1r rm2451a2r rm2451a3r rm2451b1r rm2451b21}", {
  ok=>     '245$a 245$b (partial) OK',
  notfound=>   '245$a 245$b (partial) NOT FOUND'
};
```

```
test "$combo {rm4401a1r rm4401a21}", {
  ok=>     '440$a (partial) OK',
  notfound=>  '440$a (partial) NOT FOUND'
};

test "$combo {rm4901a1r rm4901a21}", {
  ok=>     '490$a (partial) OK',
  notfound=>  '490$a (partial) NOT FOUND'
};
}

print "\nSubject search -- keyword with right truncation (BP1.9)\n\n";

runtest "\@attr 1=21 $attributes_kwt", $types[2]->{fields}, 't';

print "\nSubject search -- Exact match (BP1.10).\n\n";

$combo = '@attr 1=21 @attr 2=3 @attr 3=1 @attr 4=1 @attr 5=100 @attr 6=3';

test "$combo {rm6001a1r rm6001a2r}", {
  ok=>     '600$a OK',
  notfound=>  '600$a NOT FOUND'
};

test "$combo {rm6001a1r rm6001a2r rm6001d1r}", {
  ok=>     '600$a 600$d OK',
  notfound=>  '600$a 600$d NOT FOUND'
};

test "$combo {rm6501a1r rm6501a2r rm6501a3r}", {
  ok=>     '650$a OK',
  notfound=>  '650$a NOT FOUND'
};

test "$combo {rm6501a1r rm6501a2r rm6501a3r rm6501x1r}", {
  ok=>     '650$a 650$x OK',
  notfound=>  '650$a 650$x NOT FOUND'
};

test "$combo {rm6511a1r rm6511a2r}", {
  ok=>     '651$a OK',
  notfound=>  '651$a NOT FOUND'
};

test "$combo {rm6511a1r rm6511a2r rm6511x1r}", {
  ok=>     '651$a 651$x OK',
  notfound=>  '651$a 651$x NOT FOUND'
};

test "$combo {rm6531a1r rm6531a2r rm6531a3r}", {
  ok=>     '653$a OK',
  notfound=>  '653$a NOT FOUND'
};

print "\nSubject search -- First words in fields (BP1.11)\n\n";
```

```
$combo = '@attr 1=21 @attr 2=3 @attr 3=1 @attr 4=1 @attr 5=100 @attr 6=1';

test "$combo {rm6001a1r}", {
  ok=>     '600$a (partial) OK',
  notfound=>   '600$a (partial) NOT FOUND'
};

test "$combo {rm6001a1r rm6001a2r}", {
  ok=>     '600$a OK',
  notfound=>   '600$a NOT FOUND'
};

test "$combo {rm6501a1r rm6501a2r}", {
  ok=>     '650$a (partial) OK',
  notfound=>   '650$a (partial) NOT FOUND'
};

test "$combo {rm6501a1r rm6501a2r rm6501a3r rm6501x1r}", {
  ok=>     '650$a 650$x OK',
  notfound=>   '650$a 650$x NOT FOUND'
};

test "$combo {rm6511a1r rm6511a2r}", {
  ok=>     '651$a OK',
  notfound=>   '651$a NOT FOUND'
};

test "$combo {rm6511a1r rm6511a2r rm6511x1r}", {
  ok=>     '651$a 651$x OK',
  notfound=>   '651$a 651$x NOT FOUND'
};

test "$combo {rm6531a1r rm6531a2r rm6531a3r}", {
  ok=>     '653$a OK',
  notfound=>   '653$a NOT FOUND'
};

print "\nSubject search -- First characters in fields (BP1.12)\n\n";

$combo = '@attr 1=21 @attr 2=3 @attr 3=1 @attr 4=1 @attr 5=1 @attr 6=1';

test "$combo {rm6001a11}", {
  ok=>     '600$a (partial) OK',
  notfound=>   '600$a (partial) NOT FOUND'
};

test "$combo {rm6001a1r rm6001a21}", {
  ok=>     '600$a OK',
  notfound=>   '600$a NOT FOUND'
};

test "$combo {rm6501a1r rm6501a21}", {
  ok=>     '650$a (partial) OK',
  notfound=>   '650$a (partial) NOT FOUND'
};

test "$combo {rm6501a1r rm6501a2r rm6501a3r rm6501x11}", {
```

```
  ok=>     '650$a 650$x OK',
  notfound=>   '650$a 650$x NOT FOUND'
};

test "$combo {rm6511a1r rm6511a21}", {
  ok=>     '651$a OK',
  notfound=>   '651$a NOT FOUND'
};

test "$combo {rm6511a1r rm6511a2r rm6511x11}", {
  ok=>     '651$a 651$x OK',
  notfound=>   '651$a 651$x NOT FOUND'
};

test "$combo {rm6531a1r rm6531a2r rm6531a31}", {
  ok=>     '653$a OK',
  notfound=>   '653$a NOT FOUND'
};

print "\nTesting Any Search -- keyword with right truncation (BP1.13)\n\n";

$combo = "\@attr 1=1016 $attributes_kwt";
foreach (@types) {
  runtest $combo, $_->{fields}, 't';
}

print "\n\nInsert explanatory verbiage about RadMarc/Z-Interop  here\n";



# $Log: bathtest.pl,v $
# Revision 1.10  2005/03/03 14:59:34  mike
# Use the identity-field pair 001 and 035$a
#
# Revision 1.9  2005/03/02 22:31:15  mike
# New hosturl, the main UNT server.
# Delay set to 1s, otherwise UNT throws us off quite quickly.
# Identity field set to 035$a, as UNT discards our 001 field.
# Uses new test-set "record3a.mrc", which is the same as "record3.mrc"
#   except for the addition of the 035$a field.
#
# Revision 1.8  2005/02/15 15:30:06  quinn
# Latest rec
#
# Revision 1.7  2005/02/01 22:10:18  quinn
# This version fails on the truncated tests.
#
# Revision 1.6  2005/02/01 21:53:33  quinn
# Smallish
#
# Revision 1.5  2004/12/20 15:23:06  quinn
# Added last truncated level 1 searches
#
# Revision 1.4  2004/12/20 15:04:20  quinn
# Inserted log in tail
#
```

Appendix I: Using the Z3950::RadioMARC Perl Module



U.S. Federal Institute of Museum and Library Services
National Leadership Grant, Phase 2

A Radioactive MARC Record Approach for Interoperability Testing

Using the Net::Z3950::RadioMARC Perl Module Version 1.0

Jason Thomale

School of Library and Information Sciences
Texas Center for Digital Knowledge
University of North Texas
Denton, TX 76203

November 2005

Using the Net::Z3950::RadioMARC Perl Module, Version 1.0

1. Introduction

The Z3950::RadioMARC perl module, written by Mike Taylor of Index Data, provides a “test harness” by which one can deduce a given Z39.50 server’s behavior according to the “radioactive” interoperability testing methods outlined by Dr. William Moen. In general terms, a typical script written for this purpose will query a server based on a small set of carefully constructed records and, based on the results that are returned, create a report from which a user can deduce that server’s behavior.

In writing such a script, one first specifies a *test set*, which is comprised of one or more MARC records that one intends to have one’s tests return. Next, one sends a query to the Z server, and the result of this query is compared to the result of a separate search that the testing environment performs on the specified test set. Based on this comparison, the test harness generates a status message: “ok” if a retrieved record matched a record in the test set; “notfound” if no record in the results matched any records in the test set; or “fail” if the query could not be executed. The RadioMARC module further allows one to create one's own customized reporting messages based on these statuses.

This document, written based on the scripts included in the /examples/ directory of the RadioMARC module, illustrates some of the possibilities for putting together a complete test of a Z server. One should use this in combination with the RadioMARC module’s own documentation (http://search.cpan.org/~mirk/Net-Z3950-RadioMARC-0.07/lib/Net/Z3950/RadioMARC.pm) and the documentation about the Z-Interop Phase 2 project (http://www.unt.edu/zinterop/).

Note: This documentation uses the RadioMARC module’s simple syntax rather than its object-oriented syntax. The object-oriented syntax would be needed only in the case that a single script would test more than one database.

2. A Basic Script: authortest.pl

The authortest.pl script demonstrates a very basic usage of the Radiomarc module. The script itself tests a server only for support of an “author” keyword search, but one may use the techniques illustrated therein to create scripts of greater complexity.

Authortest.pl is divided into three parts—configuration, initial testing, and testing/reporting.

2.1. Configuration

During the configuration section, variables are defined and test harness settings are specified. For example:

```
my $attributes = '@attr 1=1003 @attr 2=3 @attr 3=3 @attr 4=2
     @attr 5=100 @attr 6=1';
my @author_fields = (
  '100$a',
  '100$d',
  '245$c',
  '700$a',
  '700$d',
  '710$a'
);

set host => 'research.lis.unt.edu', port => '2200', db =>
     'zinterop';
set delay => 0;
```

```
set identityField => '001';
set verbosity => 1;
add 'record.mrc';
```

Above, the variable $attributes holds the attributes for a Z39.50 author keyword search, minus the search term (which will be filled in later). Because this is a simple script that performs only one type of test, one set of attributes is sufficient. If one wanted to perform more sophisticated tests, one might define a more complex data structure with multiple sets of attributes. An example of such a data structure appears in section 3.1.

Next, the array @author_fields is a small array containing a sampling of MARC fields/subfields that commonly contain author data. These represent the fields/subfields that the script will test against.

Following this, the `set` statements define properties of the testing environment that dictate how the tests throughout the script will be performed. The *host*, *port*, and *db* properties are self-explanatory. The *delay* property sets the delay, in seconds, between queries being sent to the test server. The *identityField* property sets the MARC fields or subfields that uniquely identify a record. If those fields (in this case, just the 001) match between a record in the result set and the record retrieved from the test set, then they are considered to be the same record. The *verbosity* property describes the level of reporting (i.e., output) that the script will give when run, with the least verbose level being 1 and the most being 5.

Finally, the `add` statement adds one or more MARC records to the test set. The added records should also exist in the database on the server that is being tested, because the script will compare the result set from each query to the expected records that it retrieves from its test set. If the results obtained from the server do not match the expected records in the test set, then the test will fail.

Behind the scenes, the `add` method creates an inverted index of the contents of the fields and subfields present in each test set record. This index is used as a basis for the testing environment's search and retrieval of these records, which occurs when one calls the `test` method.

2.2. Initial Testing

To ensure that the server and script are working properly before beginning a lengthy series of tests, one should perform a basic initial test. For instance:

```
if (test('@attr 1=4 rm2451a1r') ne 'ok') {
     print "Test record not found in database -- unable to
       continue\n";
     exit 1;
}
```

Here the `test` method is used to perform a search for a token present in the 245$a (rm2451a1r). The 245$a, obviously, *should* be indexed by the server—if it is not, then something is wrong and the test should not continue. Thus, if the initial test does not return a status of "ok" then it should end so that one may ascertain whether the server is broken or whether one has neglected to set up the test properly (e.g., if a test-set has not been specified, if the correct record has not been loaded onto the server, or if the query sent to the server is not correct).

2.3. Testing/Reporting

One actually tests a server by sending queries to it via the `test` method according to the parameters set up within the script and according to the ultimate purpose the script is to serve. These tests might be quick and simple, or they might be quite long and complex. Then, within the framework of the test harness' `test` method, report output can be customized.

The script at hand uses the following block of code for testing:

```
foreach (@author_fields) {
  my $search = "$attributes " . radtoken $_;
  test $search, {
    ok=>"1=1003 searches $_",
    notfound=>"1=1003 DOES NOT match $_"
  };
}
```

Because this script is attempting to test for a server's support of author keyword searches, it loops through the array @author_fields, defined in section 2.1, and queries each field according to the RadioMARC methodology. First, it creates a new string containing the complete query to be sent to the server based on the $attributes necessary to carry out an author keyword search and the RadioMARC token for that field (note that *radtoken* is a subroutine that returns a complete RadioMARC token for a string containing a field$subfield pattern, e.g., "100$a"). Thus, the variable $search should now contain a string such as, "@attr 1=1003 @attr 2=3 @attr 3=3 @attr 4=2 @attr 5=100 @attr 6=1 rm1001a1r".

Next, the script queries the database. It sends $search and a very simple reporting template to `test`. If the test's status comes back as "ok" then the script reports that the system does indeed index that field appropriately for an author keyword (attribute 1=1003) search; otherwise, it reports that the system does not.

The `test` method thus performs three basic actions. First, it sends the specified query to the Z39.50 server and gets back the results. Secondly, it uses this same query to search the testing environment's own inverted index—created originally by the `add` method based on the records present in the test set—and gets back those results. Finally, it compares the two sets of results based upon the identity fields or subfields configured using the `set` method, returns a status string based on the results of this comparison, and outputs a string based on the provided reporting template, if applicable.

Output from the authortest.pl script might appear as follows:

```
1=1003 searches 100$a
1=1003 searches 100$d
1=1003 DOES NOT match 245$c
1=1003 searches 700$a
1=1003 searches 700$d
1=1003 searches 710$a
```

3. A Complete RadioMARC Scripting Example: bathtest.pl

The bathtest.pl script contains a more realistic—and more useful—example of a test script that uses the RadioMARC perl module for interoperability testing. Like the simpler authortest.pl, however, bathtest.pl still contains a configuration section, an initial test, and a testing/reporting section.

3.1. Configuration

Such a complex script necessarily contains a more extensive configuration section, which can be further divided into variable initialization, subroutine initialization, and test harness configuration.

3.1.1. Variable Initialization

This script contains the following variable initializations:

```
my $pattern = 'rmFFF1S1r';
my $hosturl = '129.120.92.237:210/INNOPAC';
my ($host, $port, $dbname) = $hosturl =~ /(.*):(.*)\/(.*)/;
```

```
my $attributes_kw = '@attr 2=3 @attr 3=3 @attr 4=2 @attr 5=100
                     @attr 6=1';
my $attributes_kwt = '@attr 2=3 @attr 3=3 @attr 4=2 @attr 5=1
                      @attr 6=1';
my $combo;
my @types = (
  {
    'name' => 'Author search (BP0.1).',
    'use' => 1003,
    'fields' => [
      '100$a', '100$d',
      '245$c',
      '700$a', '700$d',
      '710$a'
    ]
  },
  {
    'name' => 'Title search (BP0.2)',
    'use' => 4,
    'fields' => [
      '245$a', '245$b',
      '440$a',
      '490$a'
    ],
  },
  {
    'name' => 'Subject search (BP0.3)',
    'use' => 21,
    'fields' => [
      '600$a', '600$d',
      '650$a', '650$x', '650$v', '650$z',
      '653$a'
    ]
  }
);
```

Here, $pattern represents a regular expression that describes the type of tokens used as the basis for retrieving our radioactive MARC records. This allows tokens to be generated on the fly based on a given field$subfield string. The next two lines, which define $hosturl, $host, $port, and $dbname represent one method of defining the host, port, and database to be tested. Later these variables will be used to initialize the test harness. $attributes_kw and $attributes_kwt contain strings that represent the necessary attributes for a keyword search and a keyword search with right truncation, respectively. $combo is a temporary holding space for attribute combinations that will be used later in the script.

Finally, @types is a data structure that helps greatly in automating the testing procedure. It contains a list of hash references, each of which represents a different search type (according to the Bath Profile) and contains a “name” element, a “use” element, and a “fields” element. “Name” contains a string that describes the search type and is used for reporting purposes. “Use” contains the value of the use attribute for that search type. “Fields” contains a reference to an array of field$subfield strings that represent the MARC fields that can contain information related to the search type.

3.1.2. Subroutine Initialization

This script uses two subroutines: radtoken() and runtest(). They appear as follows:

```
sub radtoken {
  $_ = shift;
```

```
  my $ret = $pattern;

  my ($field, $subfield) = /(...)\$(.)/;
  $ret =~ s/FFF/$field/;
  $ret =~ s/S/$subfield/;
  return $ret;
}
```

Radtoken() returns a valid RadioMARC token based on the $pattern, defined earlier during the variable initialization stage, when given a field$subfield (e.g., 245$a). It replaces FFF in $pattern with the field and S in $pattern with the subfield. For instance, the statement `print radtoken '245$a'` would thus output “rm2451a1r”.

```
sub runtest {
  my $combo = shift;
  my $list = shift;
  my $trunc = shift;
  foreach (@{$list}) {
    my $tok = radtoken $_;
    if (defined($trunc) && $trunc eq 't') {
      $tok =~ s/.$//;
    }
    my $search = "$combo " . $tok;
    test $search, {
      ok=>"Search finds $_",
      notfound=>"Search does NOT find $_"
    };
  }
}
```

Runtest(), when given an attribute combination ($combo), a reference to a list of field$subfield strings ($list), and—optionally—a 't', which specifies that one is performing a right truncation query, will run through each FFF$S and query the server based on that subfield. If the query is successful, the script outputs "Search finds FFF$S"; otherwise, it outputs "Search does NOT find FFF$S".

3.1.3. Test Harness Configuration

Test harness configuration mainly consists of using the test harness’ `set` and `add` methods, which are discussed briefly in section 2.1. Because the test harness configuration in this script adds nothing to that which has already been discussed, the relevant statements from the bathtest.pl script are included here only for the sake of completeness.

```
set host => $host, port => $port, db => $dbname;
set delay => 1;
set identityField => '001,035$a';
set verbosity => 1;
```

3.2. Initial Testing

As in the test harness configuration for this script, the initial testing is no different than that of the authortest.pl script:

```
if (test('@attr 1=4 rm2451a1r') ne 'ok') {
     print "Test record not found in database -- unable to
       continue\n";
     exit 1;
}
```

Generally, little more than this is necessary for the initial test, although it may depend on the larger purpose of the script. One should make certain that the initial test is sufficient to cover everything for which the rest of the script tests. For instance, the RadioMARC token format of the initial test should match both the token format throughout the script and the token format of the MARC records that were added to the test set.

3.3. Testing/Reporting

After configuration and initial testing, one can begin the *actual* testing and reporting phase. In this case, because the script runs a number of tests based on the Bath Profile (BP), the report is produced as the tests are performed—that is, according to each profile level. Bathtest.pl has two major testing/reporting sections: BP level 0 (keyword) testing, which demonstrates a more automatic testing approach using loops and the @types data structure defined in section 3.1, and BP level 1, which demonstrates a more manual testing approach.

3.3.1. Automatic Testing: BP Level 0

After the report heading is produced:

```
print "Testing Level 0 keyword searching (BP0.1 to 0.4).\n\n";
```

a foreach structure loops through each @type element. For each element (i.e., for each type of BP search), we take two steps: first, the complete attribute combination is built by taking the "use" attribute specified in the @type structure and combining it with the other attributes necessary for a keyword search, which are held in the variable $attributes_kw. Secondly, the runtest subroutine is used to test each field belonging to a particular BP search type. As shown below, $combo stores the complete attribute combination needed for the test and $_->{fields} contains a list of fields that should be tested for each BP search type.

```
foreach (@types) {
  my $combo = "\@attr 1=" . $_->{'use'} . " $attributes_kw";
  print "Testing: " . $_->{name} . "\n\n";
  runtest $combo, $_->{fields};
  print "\n";
}
```

The @types array holds data only for BP levels 0.1 to 0.3. Because BP level 0.4 is a combination of levels 0.1 to 0.3, the @types array must be used differently for this test. Below, the $combo is manually set for a BP 0.4 search, and every single field in the @types array is tested against.

```
$combo = "\@attr 1=1016 $attributes_kw";
print "Testing: Any\n";
foreach (@types) {
  runtest $combo, $_->{fields};
}
```

Finally, a search is performed based on BP 1.1—author keyword search with right truncation. The query is built using the $attributes_kwt variable, which holds the appropriate attributes for a right truncation search, and a series of tests are run based on the fields corresponding to an author search ($types[0]->{fields}).

```
print "\nAuthor search -- keyword with right truncation (BP1.1)\n\n";

runtest "\@attr 1=1003 $attributes_kwt", $types[0]->{fields}, 't';
```

3.3.2. Manual Testing: BP Level 1

As one would expect, manual testing takes more space within a script than automatic testing. More than one half of the bathtest.pl script consists of manual testing (although it does test levels 1.2 through 1.13). Thus, the entire manual test is not included in this document; rather, enough of it is included so that one may get an idea about how manual testing can be done. Note that the testing included here does not make use of the runtest() or radtoken() subroutines.

The heading for the first set of BP level 1 tests (1.2) is printed and the attribute $combo defined:

```
print "\nAuthor search -- exact match (BP1.2)\n\n";

$combo = '@attr 1=1003 @attr 2=3 @attr 3=1 @attr 4=1 @attr 5=100 @attr
6=3';
```

Testing commences using only the `test` method. This way, each applicable field is manually tested to see if the server meets the requirements of the BP level. Testing for BP level 1.2 is as follows:

```
test "$combo {rm1001a1r, rm1001a2r}", {
  ok=>      '100$a with comma OK',
  notfound=>    '100$a with comma NOT FOUND'
};

test "$combo {rm1001a1r rm1001a2r}", {
  ok=>      '100$a without comma OK',
  notfound=>    '100$a without comma NOT FOUND'
};

test "$combo {rm1001a1r, rm1001a2r, rm1001d1r}", {
  ok=>      '100$a 100$d with comma OK',
  notfound=>    '100$a 100$d with comma NOT FOUND'
};

test "$combo {rm1001a1r rm1001a2r rm1001d1r}", {
  ok=>      '100$a 100$d without comma OK',
  notfound=>    '100$a 100$d without comma NOT FOUND'
};

test "$combo {rm7001a1r, rm7001a2r}", {
  ok=>      '700$a with comma OK',
  notfound=>    '700$a with comma NOT FOUND'
};

test "$combo {rm7001a1r rm7001a2r}", {
  ok=>      '700$a without comma OK',
  notfound=>    '700$a without comma NOT FOUND'
};

test "$combo {rm7001a1r, rm7001a2r, rm7001d1r}", {
  ok=>      '700$a 700$d with comma OK',
  notfound=>    '700$a 700$d with comma NOT FOUND'
};

test "$combo {rm7001a1r rm7001a2r rm7001d1r}", {
  ok=>      '700$a 700$d without comma OK',
  notfound=>    '700$a 700$d without comma NOT FOUND'
};
```

```
test "$combo {rm7101a1r rm7101a2r}", {
  ok=>      '710$a without comma OK',
  notfound=>    '710$a without comma NOT FOUND'
};

test "$combo {rm2451c1r rm2451c2r rm2451c3r}", {
  ok=>      '245$c without comma OK',
  notfound=>    '245$c without comma NOT FOUND'
};
```

For more examples of manual tests, please see the bathtest.pl script itself.

3.4. Bathtest.pl Sample Output

The following sample output includes only those tests discussed throughout this document (BP 0.1 through 0.4, 1.1, and 1.2).

```
Bath compliance test script

Test date: Mon Apr 11 11:22:05 2005
Test target: 129.120.92.237:210/INNOPAC

Testing Level 0 keyword searching (BP0.1 to 0.4).

Testing: Author search (BP0.1).

Search finds 100$a
Search finds 100$d
Search does NOT find 245$c
Search finds 700$a
Search finds 700$d
Search finds 710$a

Testing: Title search (BP0.2)

Search finds 245$a
Search finds 245$b
Search finds 440$a
Search does NOT find 490$a

Testing: Subject search (BP0.3)

Search finds 600$a
Search finds 600$d
Search finds 650$a
Search finds 650$x
Search finds 650$v
Search finds 650$z
Search does NOT find 653$a

Testing: Any
Search finds 100$a
Search finds 100$d
Search does NOT find 245$c
Search finds 700$a
Search finds 700$d
```

```
Search finds 710$a
Search finds 245$a
Search finds 245$b
Search finds 440$a
Search does NOT find 490$a
Search finds 600$a
Search finds 600$d
Search finds 650$a
Search finds 650$x
Search finds 650$v
Search finds 650$z
Search finds 653$a

Author search -- keyword with right truncation (BP1.1)

Search finds 100$a
Search finds 100$d
Search does NOT find 245$c
Search finds 700$a
Search finds 700$d
Search finds 710$a

Author search -- exact match (BP1.2)

100$a with comma NOT FOUND
100$a without comma NOT FOUND
100$a 100$d with comma OK
100$a 100$d without comma OK
700$a with comma NOT FOUND
700$a without comma NOT FOUND
700$a 700$d with comma OK
700$a 700$d without comma OK
710$a without comma OK
245$c without comma NOT FOUND
```

4. Discussion

4.1. Advanced Testing/Reporting

The Z3950::RadioMARC perl module is a tremendously useful tool for testing Z39.50 compliant servers using the “radioactive MARC” testing approach. This document and the included example scripts only scratch the surface of what can be accomplished, both in terms of testing and in terms of creating useful reports. One may employ several more advanced testing methods to make one’s test scripts more sophisticated—however, a script’s complexity increases quickly with their implementation.

For example, one might want to test one’s servers for differences in indexing based on record format. The complete set of RadioMARC test records does include one record for each format. Thus, one’s script would need to deal with having multiple records in the test set, running the same sets of tests multiple times on multiple formats, and changing the format of the RadioMARC tokens for each set of tests.

Furthermore, one may automate the more complex search types such as exact match searches, in which case one would need to develop more complicated data structures detailing one’s strategy for carrying out those searches. One might even make some types of searches conditional based on the success or failure of other searches. During an author exact match search, for instance, one might search for terms in the 100$a and 100$d with no punctuation and then again with correct punctuation. If, and only if, both these searches are found, then one could search the same subfields using random punctuation between

search terms. If all three tests are successful, one might conclude that the server ignores punctuation for this type of search. If, however, either of the initial two tests fail, then searching for random punctuation would be pointless because one would already know that the server either keeps punctuation, strips punctuation, or does not include those two fields together at all in its author index.

In automating long sets of tests, one might wish to build separate data files containing the appropriate attribute combinations and MARC subfields that apply for each test. In this case one would need to write subroutines to ingest these files and put them into the appropriate data structures for testing purposes.

Finally, automated analysis of test results presents a further layer of complexity. In implementing it, one must create data structures that can store the results of tests as they run (rather than simply outputting the results) in a meaningful way. Then one must write routines that analyze the data contained in these structures, drawing appropriate conclusions based on the tests, and create preformatted reports.

4.2. Multiple-Record Test Sets

In using multiple records within a test set, one should be cognizant of the RadioMARC module's behavior when it indexes and searches for records in a test set. Ignoring this could cause test results to be widely skewed.

A test will fail if the test harness retrieves no record from the test set index no matter whether or not a record is returned from the Z39.50 server itself. Obviously this behavior might cause problems, as test results could be just as much at the mercy of the RadioMARC module's own indexing policies as they would be the Z39.50 server's indexing policies. Generally this is not a problem for keyword searching, as keyword indexes are rather predictable. Exact match and other phrase-type searches, however, are not straight-forward and difficult, if not impossible, to predict.

Fortunately, the RadioMARC module's `test` and `add` methods provide a simple way for users to specify exactly which record a particular test is meant to return. When this capability is used, the test harness does *not* search within its own inverted index of test set records because it already "knows" which record the user writing the script intends to find. That way, results from tests that utilize phrase indexes will not be affected by the way the RadioMARC module itself indexes the test set records.

First, the `add` method returns a list of tokens that represent each record added to the test set. The following would keep track of these tokens:

```
my @records = add('lib/record3a.mrc');
```

Then the `test` method allows one to specify the token that a test is supposed to find:

```
test ('@attr 1=4 rm245a1r', { ok => '245$a FOUND',
                              notfound => '245$a NOT FOUND',
                              token => $records[0] });
```

Appendix J: Output From *bathtest.pl* Interoperability Testing

Output From The bathtest.pl,v 1.8 2005/02/15

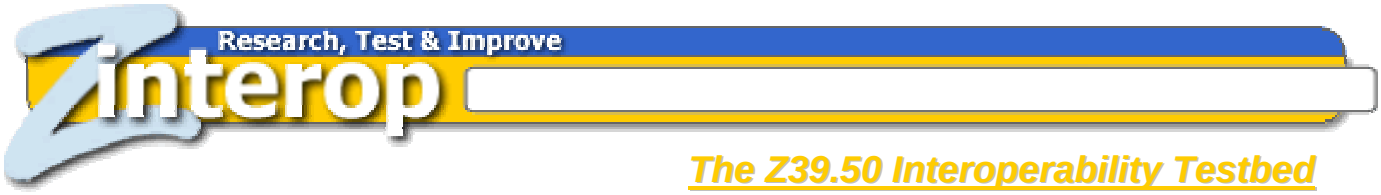
```
Bath compliance test script

Test date: Thu Feb 17 12:03:50 2005
Test target: research.lis.unt.edu:2200/zinterop

Testing Level 0 keyword searching (BP0.1 to 0.4).

Testing: Author search (BP0.1).

Search finds 100$a
Search does NOT find 100$d
Search finds 245$c
Search finds 700$a
Search finds 700$d
Search finds 710$a

Testing: Title search (BP0.2)

Search finds 245$a
Search finds 245$b
Search finds 440$a
Search finds 490$a

Testing: Subject search (BP0.3)

Search finds 600$a
Search finds 600$d
Search finds 650$a
Search finds 650$x
Search finds 650$v
Search finds 650$z
Search finds 653$a

Testing: Any
Search finds 100$a
Search does NOT find 100$d
Search finds 245$c
Search finds 700$a
Search finds 700$d
Search finds 710$a
Search finds 245$a
Search finds 245$b
Search finds 440$a
Search finds 490$a
Search finds 600$a
Search finds 600$d
Search finds 650$a
Search finds 650$x
Search finds 650$v
Search finds 650$z
Search finds 653$a
```

```
Author search -- keyword with right truncation (BP1.1)

Search finds 100$a
Search finds 100$d
Search finds 245$c
Search finds 700$a
Search finds 700$d
Search finds 710$a

Author search -- exact match (BP1.2)

100$a with comma NOT FOUND
100$a without comma NOT FOUND
100$a 100$d with comma NOT FOUND
100$a 100$d without comma NOT FOUND
700$a with comma NOT FOUND
700$a without comma NOT FOUND
700$a 700$d without comma OK
700$a 700$d without comma OK
710$a without comma OK
245$c without comma NOT FOUND

Author search -- first words in field (BP 1.3).

100$a without comma OK
100$a (partial) OK
100$a 100$d without comma NOT FOUND
700$a without comma OK
700$a 700$d without comma OK
710$a without comma OK
245$c without comma NOT FOUND

Author search -- first characters in field (BP 1.4).

100$a without comma NOT FOUND
100$a (partial) NOT FOUND
100$a 100$d without comma OK
700$a without comma NOT FOUND
700$a 700$d without comma NOT FOUND
710$a without comma NOT FOUND
245$c without comma NOT FOUND

Title search -- keyword with right truncation (BP1.5)

Search finds 245$a
Search finds 245$b
Search finds 440$a
Search finds 490$a

Title search -- Exact match (BP1.6).

245$a NOT FOUND
245$a 245$b OK
Reverse order test OK
440$a NOT FOUND
490$a NOT FOUND
```

```
Title search -- First words in field (BP1.7).

245$a OK
245$a (partial) OK
245$a 245$b (partial) OK
440$a (partial) NOT FOUND
490$a (partial) NOT FOUND

Title search -- First characters in field (BP1.8).

245$a NOT FOUND
245$a (partial) NOT FOUND
245$a 245$b (partial) NOT FOUND
440$a (partial) NOT FOUND
490$a (partial) NOT FOUND

Subject search -- keyword with right truncation (BP1.9)

Search finds 600$a
Search finds 600$d
Search finds 650$a
Search finds 650$x
Search finds 650$v
Search finds 650$z
Search finds 653$a

Subject search -- Exact match (BP1.10).

600$a NOT FOUND
600$a 600$d OK
650$a NOT FOUND
650$a 650$x NOT FOUND
651$a NOT FOUND
651$a 651$x OK
653$a NOT FOUND

Subject search -- First words in fields (BP1.11)

600$a (partial) OK
600$a OK
650$a (partial) OK
650$a 650$x OK
651$a OK
651$a 651$x OK
653$a NOT FOUND

Subject search -- First characters in fields (BP1.12)

600$a (partial) NOT FOUND
600$a NOT FOUND
650$a (partial) NOT FOUND
650$a 650$x NOT FOUND
651$a NOT FOUND
651$a 651$x NOT FOUND
653$a NOT FOUND

Testing Any Search -- keyword with right truncation (BP1.13)
```

```
Search finds 100$a
Search finds 100$d
Search finds 245$c
Search finds 700$a
Search finds 700$d
Search finds 710$a
Search finds 245$a
Search finds 245$b
Search finds 440$a
Search finds 490$a
Search finds 600$a
Search finds 600$d
Search finds 650$a
Search finds 650$x
Search finds 650$v
Search finds 650$z
Search finds 653$a


Insert explanatory verbiage about RadMarc/Z-Interop  here
```

Appendix K: MARCdocs: The MARC 21 Bibliographic Format Database



U.S. Federal Institute of Museum and Library Services
National Leadership Grant, Phase 2

A Radioactive MARC Record Approach for Interoperability Testing

MARCdocs
The MARC 21 Bibliographic Format Database
Version 1.2

Jason Thomale
William E. Moen

School of Library and Information Sciences
Texas Center for Digital Knowledge
University of North Texas
Denton, TX 76203

October 2004

MARCdocs
The MARC 21 Bibliographic Format Database
Version 1.2

1. Introduction and Purpose

MARCdocs, the MARC 21 Documentation Database, is a pilot effort aimed at structuring the textual documentation from the MARC 21 Format for Bibliographic Data into a relational database. Using a database approach for the authoritative MARC documentation provides new opportunities for various applications, including more efficient maintenance of the documentation, easier and quicker updates and changes, exporting selected data in XML, as well as an aid to research into the development and evolution of MARC, a learning/reference tool for those seeking to understand the MARC format, and a tool to help those devising new bibliographic structures.

The database was first envisioned by Dr. William Moen of the University of North Texas School of Library and Information Sciences as a tool for his ongoing research into metadata utilization and interoperability. Dr. Shawne Miksa, Penelope Bernardino, JungWon Yoon, and Jason Thomale have carried this idea forward to implementation.

This application uses open source software tools including Linux, MySQL, and PhP. Jason Thomale, a masters student in the School of Library and Information Sciences, designed and implemented the database, developed procedures for data loading, and designed and programmed the web interface for the MySQL database.

2. Database Organization Overview

The MARCdocs data model mirrors the basic structure of most MARC fields as they are presented in the *MARC 21 Bibliographic Format* documentation. Six distinct entities relating to each field are represented:

1. Fields
2. Subfields
3. Indicators
4. Indicator Values
5. Character Position (for fields 006, 007, and 008)
6. Character Position Values (for fields 006, 007, and 008)

The Entity Relationship Diagram (ERD) in Figure 1 illustrates how these entities are related to one another:

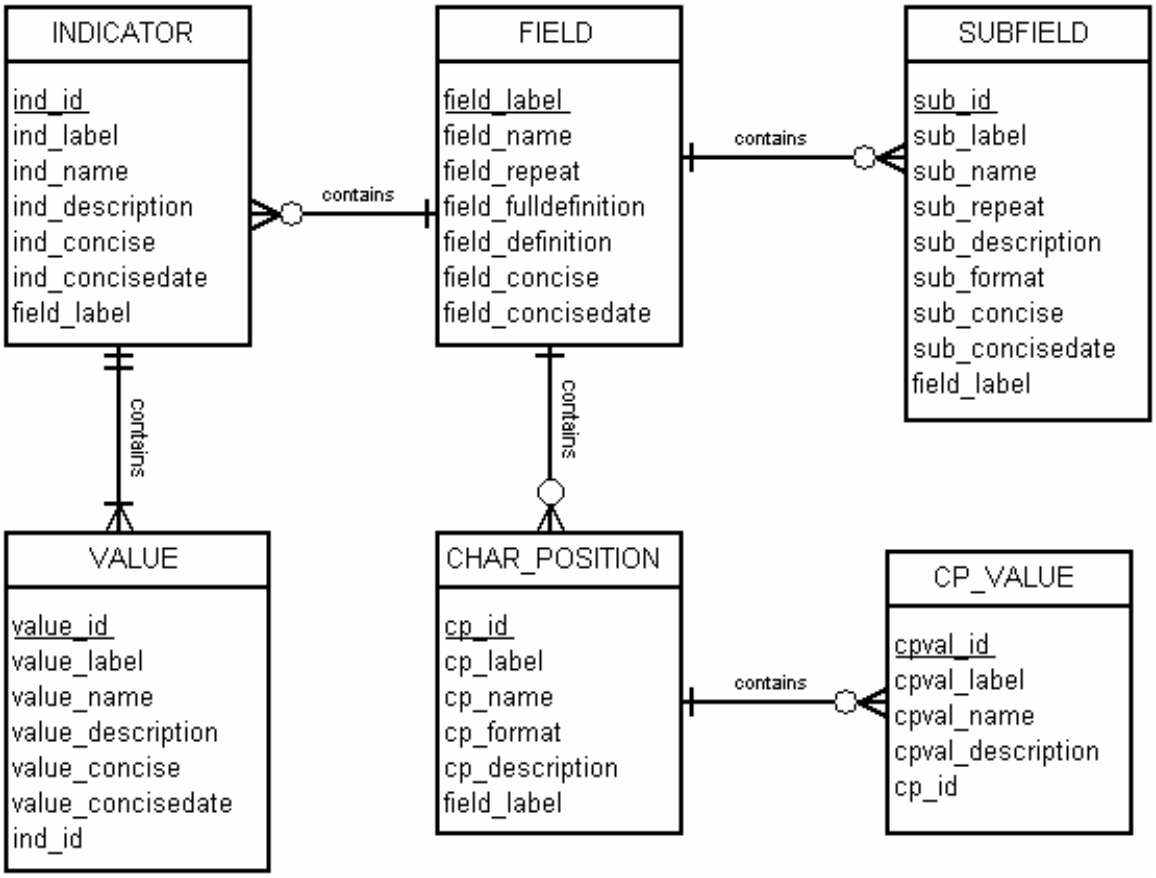


Figure 1. Entity Relationship Diagram for MARCdocs Database

3. Field

The Field entity stores general information about each field documented in the *MARC 21 Bibliographic Format* that can appear within a MARC record.

3.1. Rules

Each field may have zero or more Subfields.

Each field may have zero or more Indicators.

Each field may have zero or more Character Positions.

3.2. Data Structure

Data Element Name	Data Type	Description
FIELD_LABEL Primary Key	Char (3)	The 3-digit field code that uniquely identifies the field.
FIELD_NAME	Varchar (75)	The name of the field.
FIELD_REPEAT	Enum	Describes whether the field is repeatable or not. R=Repeatable, NR=Not Repeatable.
FIELD_FULLDEFINITION	Text	The field's definition as listed under "Field Definition and Scope" in the *MARC 21 Bibliographic Format*.
FIELD_DEFINITION	Text	The field's definition as listed in the Concise version of the *MARC 21 Bibliographic Format*.
FIELD_CONCISE	Enum	Describes whether the field appears in MARC 21 Concise—can be "Yes" or "No."
FIELD_CONCISEDATE	Varchar (4)	Describes the version of MARC 21 Concise from which data for this field was taken.

4. Subfield

The Subfield entity contains a listing of all possible subfields for each field.

4.1. Rules

Each subfield must belong to one and only one field.

4.2. Data Structure

Data Element Name	Data Type	Description
SUB_ID Primary Key	Varchar (7)	A 5-7 digit code that uniquely identifies the Subfield. Construct the SUB_ID as follows: parent field's FIELD_LABEL + S + SUB_LABEL. Subfield 'a' of field 010 would thus have a SUB_ID of 010Sa.
SUB_LABEL	Char (3)	The 3-digit subfield code.
SUB_NAME	Varchar (75)	The name of the subfield.
SUB_REPEAT	Enum	R=Repeatable, NR=Not Repeatable.
SUB_DESCRIPTION	Text	The subfield's description as listed under "Guidelines for Applying Content Designators" in the *MARC 21 Bibliographic Format*.
SUB_FORMAT	Set	The type of material to which this subfield applies. Can be one or more of the following: books, computer files, maps, music, continuing resources, visual materials, mixed materials, or sound recordings. If a subfield applies to all materials, use the word 'All.'
SUB_CONCISE	Enum	Describes whether the subfield appears in MARC 21 Concise—can be "Yes" or "No."
SUB_CONCISEDATE	Varchar (4)	Describes the version of MARC 21 Concise from which data for this subfield was taken.
FIELD_LABEL Foreign Key	Char (3)	A 3-digit code that uniquely identifies the parent field of the subfield.

5. Indicator

The Indicator entity lists all indicators belonging to a particular field.

5.2. Rules

Each indicator must belong to one and only one field.

Indicators may have one or more values.

5.2. Data Structure

Data Element Name	Data Type	Description
IND_ID Primary Key	Varchar (5)	A 5-digit code that uniquely identifies the Indicator. Construct the IND_ID as follows: parent field's FIELD_LABEL + I + IND_LABEL. The first indicator of field 022 would thus have an IND_ID of 022I1.
IND_LABEL	Enum	1 = First Indicator; 2 = Second Indicator
IND_NAME	Varchar (60)	The name of the indicator.
IND_DESCRIPTION	Text	The indicator's description as listed under "Guidelines for Applying Content Designators" in the *MARC 21 Bibliographic Format*.
IND_CONCISE	Enum	Describes whether the indicator appears in MARC 21 Concise—can be "Yes" or "No."
IND_CONCISEDATE	Varchar (4)	Describes the version of MARC 21 Concise from which data for this indicator was taken.
FIELD_LABEL Foreign Key	Char (3)	A 3-digit code that uniquely identifies the parent field of the indicator.

6. Value

The Value entity contains all possible values for each indicator.

6.1. Rules

Each value must belong to one and only one indicator.

'Blank' is an acceptable value. Always use '#' to notate a blank.

6.2. Data Structure

Data Element Name	Data Type	Description
VALUE_ID Primary Key	Varchar (7)	A 7-digit code that uniquely identifies the value. Construct the VALUE_ID as follows: parent indicator's parent field's FIELD_LABEL + I + parent indicator's IND_LABEL + V + VALUE_LABEL. Use '#' for a blank value. The blank value of field 022's first indicator would thus have a VALUE_ID of 022I1V#.
VALUE_LABEL	Char (1)	The 1-digit value code. Again, use '#' for blank.
VALUE_NAME	Varchar (100)	The name of the value.
VALUE_DESCRIPTION	Text	The value's description as listed under "Guidelines for Applying Content Designators" in the *MARC 21 Bibliographic Format*.
VALUE_CONCISE	Enum	Describes whether the indicator value appears in MARC 21 Concise—can be "Yes" or "No."
VALUE_CONCISEDATE	Varchar (4)	Describes the version of MARC 21 Concise from which data for this indicator value was taken.
IND_ID Foreign Key	Varchar (5)	A 5-digit code that uniquely identifies the parent indicator of the value.

7. Char_Position

For fields that contain lists of characters where the positions of the characters have significance (such as the 006, 007, and 008 fields), the Char_Position entity keeps record of those characters' positions' meanings.

7.1. Rules

Each character position must belong to one and only one field.

The "2-letter format code" for building the CP_ID varies depending on the character position's parent field. For fields 006 and 008, use the following (taken from the *MARC 21 Bibliographic Format*): Books = BK, Computer Files = CF, Maps = MP, Music = MU, Serials = SE, Visual Materials = VM, and Mixed Materials = MX; for All Materials, use AM. For the 007 field, use the following: MP = Map, CF = Computer File, GL = Globe, TA = Tactile Material, PG = Projected graphic, MI = Microform, NG = Nonprojected Graphic, MO = Motion Picture, KI = Kit, NO = Notated music, RE = Remote-sensing image, SO = Sound recording, TE = Text, VI = Videorecording, and UN = Unspecified; use AM for All Materials.

7.2. Data Structure

Data Element Name	Data Type	Description
CP_ID Primary Key	Varchar (9)	A 9-digit code that uniquely identifies the character position. Construct the CP_ID as follows: parent field's FIELD_LABEL + CP + 1st applicable character position (CP_LABEL) + 2-letter format code.
CP_LABEL	Varchar (60)	The actual numerical character position(s). If listing more than one, separate by commas (i.e., 18,19,20,21).
CP_NAME	Varchar (60)	The name of the character position.
CP_FORMAT	Set	The format(s) to which the character position applies.
CP_DESCRIPTION	Text	The character position's description as listed under "Guidelines for Applying Content Designators" in the *MARC 21 Bibliographic Format*.
FIELD_LABEL Foreign Key	Char (3)	A 3-digit code that uniquely identifies the parent field of the character position.

8. CP_Value

For fields that contain lists of characters where the positions of the characters have significance (such as the 006, 007, and 008 fields), the CP_Value entity keeps record of those characters' positions' values' meanings.

8.1. Rules

Each character position value must belong to one and only one character position.

If a value is blank, use '#' in the CPVAL_ID and the CPVAL_LABEL.

8.2. Data Structure

Data Element Name	Data Type	Description
CPVAL_ID Primary Key	Varchar (16)	A 10-digit code that uniquely identifies the character position value. Construct the CPVAL_ID as follows: CP_ID + CPVAL_LABEL.
CPVAL_LABEL	Char (7)	The actual value.
CPVAL_NAME	Varchar (150)	The name of the character position value.
CPVAL_DESCRIPTION	Text	The value's description as listed under "Guidelines for Applying Content Designators" in the *MARC 21 Bibliographic Format*.
CP_ID Foreign Key	Varchar (9)	A 9-digit code that uniquely identifies the parent field of the character position.

9. Sample Queries

Note: For these sample queries, only a small portion of the total MARC Bibliographic Format has been entered into the database. Only the contents of fields 022, 045, and 018 exist. For the character position queries (section 9.4), the entirety of the database is used.

9.1. Field/Subfield Queries

QUERY 1

What subfields exist for the 045 field?

```
SQL:    SELECT Field.field_label, Field.field_name, Subfield.sub_id,
          Subfield.sub_label, Subfield.sub_name
FROM Field JOIN Subfield ON Field.field_label =
       Subfield.field_label
          WHERE Field.field_label = 045;
```

field_label	field_name	sub_id	sub_label	sub_name
045	Time Period of Content	045Sa	a	Time period code
045	Time Period of Content	045Sb	b	Formatted 9999 B.C. through C.E. time period
045	Time Period of Content	045Sc	c	Formatted pre-9999 B.C. time period
045	Time Period of Content	045S6	6	Linkage
045	Time Period of Content	045S8	8	Field link and sequence number

QUERY 2

Which fields have Linkage subfields, and what are those subfields?

```
SQL:       SELECT Field.field_label, Subfield.sub_label, Subfield.sub_name
      FROM Field JOIN Subfield ON Field.field_label =
  Subfield.field_label
          WHERE Subfield.sub_name LIKE '%link%';
```

field_label	sub_label	sub_name
022	6	Linkage
022	8	Field link and sequence number
045	6	Linkage
045	8	Field link and sequence number

9.2. Field/Subfield Format Queries

QUERY 1

What formats apply to the 018 field?

```
SQL:       SELECT DISTINCT Subfield.sub_format AS format, Field.field_label,
             Field.field_name
           FROM Field JOIN Subfield ON Field.field_label =
             Subfield.field_label
           WHERE Field.field_label = 018;
```

format	field_label	field_name
Books,Serials	018	Copyright Article-Fee Code

QUERY 2

To what fields (and their subfields) does the “Serials” format apply?

```
SQL:    SELECT Field.field_label, Field.field_name, Subfield.sub_label,
          Subfield.sub_name, Subfield.sub_format
        FROM Field JOIN Subfield ON Field.field_label =
          Subfield.field_label
        WHERE Subfield.sub_format LIKE '%Serials%' OR Subfield.sub_format
          LIKE '%All%';
```

field_label	field_name	sub_label	sub_name	sub_format
022	International Standard Serial Number	a	International Standard Serial Number	Serials
022	International Standard Serial Number	y	Incorrect ISSN	Serials
022	International Standard Serial Number	z	Canceled ISSN	Serials
022	International Standard Serial Number	6	Linkage	Serials
022	International Standard Serial Number	8	Field link and sequence number	Serials
045	Time Period of Content	a	Time period code	All
045	Time Period of Content	b	Formatted 9999 B.C. through C.E. time period	All
045	Time Period of Content	c	Formatted pre-9999 B.C. time period	All
045	Time Period of Content	6	Linkage	All
045	Time Period of Content	8	Field link and sequence number	All
018	Copyright Article-Fee Code	a	Copyright article-fee code	Books,Serials

9.3. Field/Indicator/Value Queries

QUERY 1

What does it mean if the 022 field’s first indicator is blank?

```
SQL:    SELECT Field.field_label, Indicator.ind_label,
          Indicator.ind_name, Value.value_label, Value.value_description
        FROM Field JOIN Indicator ON Field.field_label =
          Indicator.field_label JOIN Value ON Indicator.ind_id =
          Value.ind_id
          WHERE Field.field_label = 022 AND Indicator.ind_label = 1 AND
            Value.value_label ='#';
```

field_label	ind_label	ind_name	value_label	value_description
022	1	Level of international interest	#	Value # indicates that the level of international interest is unknown or not specified. This value is used by all institutions other than the National Serials Data Program (NSDP) and ISSN Canada when recording the ISSN from an issue or from a bibliography.

QUERY 2

What fields have indicators defined, and what are those indicators?

```
SQL:    SELECT Field.field_label, Field.field_name, Indicator.ind_label,
          Indicator.ind_name
        FROM Field JOIN Indicator ON Field.field_label =
          Indicator.field_label;
```

field_label	field_name	ind_label	ind_name
022	International Standard Serial Number	1	Level of international interest
045	Time Period of Content	1	Type of time period in subfield b or c

9.4. Character Position Queries

QUERY 1

What does character position 18 in field 008 signify for the Music format?

```
SQL:    SELECT Field.field_label, Field.field_name,
          Char_Position.cp_label,
          Char_Position.cp_name, Char_Position.cp_format,
          Char_Position.cp_description
        FROM Field LEFT JOIN Char_Position ON Field.field_label =
          Char_Position.field_label
        WHERE Field.field_label = 008 AND Char_Position.cp_label LIKE
          "%18%" AND Char_Position.cp_format = "Music";
```

field_label	field_name	cp_label	cp_name	cp_format	cp_description
008	FIXED-LENGTH DATA ELEMENTS	18,19	Form of composition	Music	A two-character code that indicates the form of composition. The codes are based on Library of Congress subject headings. If more than one code is appropriate, the code /mu/ (Multiple forms) is used in 008/18-19 and all appropriate specific codes are given in field 047 (Form of Composition).

QUERY 2

For the above results, what are the possible values for this character position, and what do they mean?

```
SQL:     SELECT CP_Value.cpval_label, CP_Value.cpval_name
         FROM Field LEFT JOIN Char_Position ON Field.field_label =
           Char_Position.field_label LEFT JOIN CP_Value ON
           Char_Position.cp_id = CP_Value.cp_id
         WHERE Field.field_label = 008 AND Char_Position.cp_label LIKE
           "%18%" AND Char_Position.cp_format = "Music";
```

(PLEASE NOTE: Only the first 30 results are shown.)

pval_label	cpval_name
An	Anthems
Bd	Ballads
Bg	Bluegrass music

pval_label	cpval_name
bl	Blues
bt	Ballets
ca	Chaconnes
cb	Chants, Other religions
cc	Chant, Christian
cg	Concerti grossi
ch	Chorales
cl	Chorale preludes
cn	Canons and rounds
co	Concertos
cp	Chansons, polyphonic
cr	Carols
cs	Chance compositions
ct	Cantatas
cy	Country music
cz	Canzonas
dg	Dance forms
dv	Divertimentos, serenades, cassations, divertisseme...
fg	Fugues
fm	Folk music
ft	Fantasias
gm	Gospel music
hy	Hymns
jz	Jazz
mc	Musical revues and comedies
md	Madrigals
mi	Minuets

Appendix L: Creating Radioactive MARC Records and Z Queries Using the MARCdocs Database

Note: This was an internal discussion paper for exploring the use of the MARCdocs database in the context of the Z-Interop2 interoperability testing framework

Creating Radioactive MARC Records and Z Queries Using the MARCdocs Database

Draft – December 2, 2004

Prepared by
William E. Moen

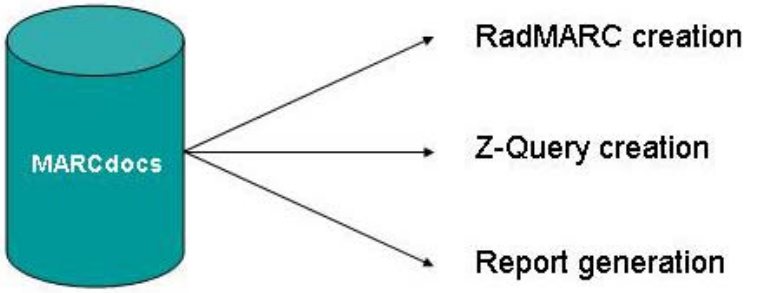
Introduction

This document describes how we can extend a relational database of MARC documentation to store the appropriate information that will support the automatic generation of the special, diagnostic MARC records we call radioactive MARC (RadMARC) records. The information contained in the database will also support the generation of the Z queries used in the interoperability testing.

The MARCdocs Database

Over the summer and early fall, project staff created an application to store information from the *MARC 21 Concise Format for Bibliographic Data*. MARCdocs, the MARC 21 Documentation Database, is a pilot effort aimed at structuring the textual documentation from the MARC 21 Format for Bibliographic Data into a relational database. Using a database approach for the authoritative MARC documentation provides new opportunities for various applications. This application uses open source software tools including Linux, MySQL, and PhP.

Based on a suggestion from Sebastian Hammer, Index Data, the project staff has discussed how the MARCdocs application could be extended to support higher levels of automated MARC record creation, interoperability testing, and report generation, as indicated in the figure below.



Requirements for Extensions to MARCdocs Database

Two primary pieces of information need to be stored in the MARCdocs database to support the RadMARC and Z-Query creation:

- Indication of which subfields could be indexed to support U.S. National Profile searches, and which specific searches are supported
- One or more tokens for each of these subfields

From previous research, we know that the occurrences of MARC content designation vary widely. We did a frequency count of MARC content designation used In the Z-Interop test dataset of more than 400,000

MARC records from OCLC's WorldCat database. The table below shows a sample of the frequency analysis.

MARC 21 Field	MARC Subfield	Occurrence
001		419,657
003		419,657
005		419,657
006		652
007		30,556
008		419,657
010	a	305,407
010	b	2
010	z	6,627
650	2	15,361
650	6	9
650	a	602,362
650	b	28
650	c	4
650	k	2
650	v	83,607
650	x	326,867
650	y	32,728
650	z	231,459

Further, we analyzed the occurrence of content designation that could be indexed to support Author, Title, and Subject searches, and discovered that 19 of the more than 500 subfields that could be indexed accounted for 80% of all occurrences. The table below shows these 19 subfields.

# of Occurrences	Marc 21 Field	Subfield	Description	Index
602,362	650	a	Subject added entry Topical Term Subfield a = Topical term or geographic name as entry element	Subject
419,641	245	a	Title Statement Subfield a = Title	Title
329,796	245	c	Title Statement Subfield c = statement of responsibility	Author
326,867	650	x	Subject added entry Topical Term Subfield x = General subdivision	Subject
318,692	100	a	Main entry Personal Name Subfield a = personal name	Author
231,459	650	z	Subject added entry Topical Term Subfield z = Geographic subdivision	Subject
176,916	700	a	Added entry Personal Name Subfield a = personal name	Author
169,178	245	b	Title Statement Subfield b = Remainder of title	Title
149,540	100	d	Main entry Personal Name Subfield d = dates associated with a name	Author
118,647	651	x	Subject added entry Geographic Name Subfield x = General subdivision	Subject
113,050	651	a	Subject added entry Geographic Name Subfield a = Geographic name	Subject
83,607	650	v	Subject added entry Topical Term Subfield v = Form subdivision	Subject
74,606	700	d	Added entry Personal Name Subfield d = dates associated with a name	Author
69,636	600	a	Subject added entry Personal Name Subfield a = personal name	Subject
66,375	710	a	Added entry Corporate Name Subfield a = corporate name or jurisdiction name	Author
64,433	440	a	Series Statement Added Entry Title Subfield a = title	Title
62,853	490	a	Series Statement Subfield a = Series statement	Title
56,229	600	d	Subject added entry Personal Name	Subject

# of Occurrences	Marc 21 Field	Subfield	Description	Index
			Subfield d = dates associated with a name	
55,311	653	a	Index Term Uncontrolled Subfield a = the term	Subject

We can identify various sets of subfields that could be indexed to support the U.S. National Profile Level 0 searches, and use the frequency count from the previous analysis to select sets of subfields. The selection could be based on a threshold of occurrence (e.g., select all subfields that could be indexed to support the Author Keyword Search that occur more than 100,000 times). This would yield a smaller number of subfields per record that needed to be populated.

A third piece of information that needs to be stored in the MARCdocs database is the frequency count for all content designation. The frequency count number will be taken from our previous analysis.

Proposed Extensions to MARCdocs Database

Based on the above requirements, we propose to extend the database structure to store the following information:

- **Frequency count of all MARC content designation.** This data will be stored in a new database element that will contain an integer. Data for this element will be batch loaded from the frequency analysis spreadsheet reports

- **Indexable fields to support U.S. National Profile searches.** The data to be stored are the search labels in the U.S. National Profile. We know that the same subfield can support more than one search. For example, the 245$a can be indexed to support the following searches:
    - Title Search – Keyword
    - Title Search – Keyword with Right Truncation
    - Title Search – Exact Match
    - Title Search – First Words in Field
    - Title Search – First Characters in Field

  Therefore, for these searches, the following will be stored in a new database element:
    - BP0.2
    - BP1.5
    - BP1.6
    - BP1.7
    - BP1.8

  Initial data for this element will be batch loaded from the indexing guidelines document which identifies all candidate subfields that could be indexed for U.S. National Profile searches:
    - BP0.1 (Author Search – Keyword)
    - BP0.2 (Title Search – Keyword)
    - BP0.3 (Subject Search – Keyword)
    - BP0.4 (Any Search – Keyword)
- **Subfield tokens.** We propose to store tokens in the MARCdocs database. Tokens may consist of one or more strings of characters bounded by blank spaces. The form of the tokens that occur in the RadMARC records will be something like the following:

  [MARCFieldTag][SubfieldCode]xxxxxx yyyyy zzzzz

  for example:

  245axxxxxx yyyyy zzzzz

  However, there is no need to store the [MARCFieldTag][SubfieldCode] as part of the token in the database, since that can be generated based on the fact that each unique token will be

associated with a specific subfield. Therefore, the form of a token stored in the MARCdocs database for a subfield that supports a specific search may look like the following:

xxxxxx yyyyy zzzzz

Token Requirements to Support Profile Level 0 and 1 Searches for Author, Title, Subject

We discussed how a single token for a subfield could support the interoperability testing of more than one profile-defined search. Taking the example of the different title searches defined in the U.S. National Profile, we believe we can create a single token that can be used for testing each of the five title searches. In the following example, we will look at a token for the 245$a.

1. To support testing of **BP0.2 Title Search – Keyword** we would need at least the following token in the database: **xxxxxx.** The token available for the Z-query could look like: **245axxxxxx**

2. To support testing of **BP1.5 Title Search – Keyword with Right Truncation**, the same token would be sufficient, as long as there was some logic in the test script creating the Z-query, for example, to truncate after the seventh character. The token in the database would be: **xxxxxx.** The token used for the Z-query could look like: **245axxx**

3. To support testing of **BP1.6 Title Search – Exact Match**, the same token could be considered sufficient, but not very useful. So, we would have a token with at least two character strings, so the following could be a token in the database: **xxxxxx yyyyy.** The token used for the Z-query could look like: **245axxxxxx yyyyy**

4. To support testing of **BP1.7 Title Search – First Words in Field**, the same token could be used as with **BP1.6**, but that wouldn't necessarily be useful. So, we would add another character string to the token and have: **xxxxxx yyyyy zzzzz** The token used for the Z-query could look like: **245axxxxxx yyyyy**

5. To support testing of **BP1.8 Title Search – First Characters in Field**, we could use the token defined for BP1.7: **xxxxxx yyyyy zzzzz.** The token available for the Z-query could look like: **245axxx**

Therefore, with some deliberation, we might be able to conclude that the token **xxxxxx yyyyy zzzzz** could be stored with the 245$a in the MARCdocs database, and this token will be sufficient to test all five profile-defined title searches. This may be too simplistic, but we propose this for discussion.

For the initial records, we may want to consider hand-crafting the tokens if the number of subfields that need to be populated is relatively small. But at some point, automatic generation of the tokens would be desirable.

Format of the Tokens

Discussions resulting from the first draft of this document yielded agreement among the project team on a structured and semantically rich token. The token elements will have the following component parts:

- A single alpha character for lefthand padding: Agreement to use the letter "r" for this character
- A single alpha character to indicate the format of the material being described or type of record: Agreement to use the code as defined in MARC 21 for Leader/06 – Type of Record as follows:
  - a - Language material
    Includes printed, microform, and electronic language material.
  - c - Notated music
    Includes microform and electronic notated music.
  - d - Manuscript notated music
    Includes microform manuscript music.

- e - Cartographic material
  Includes maps, atlases, globes, digital maps, and other cartographic items.
- f - Manuscript cartographic material
  Includes microform manuscript maps.
- g - Projected medium
  Examples include: motion pictures, videorecordings (including digital video), filmstrips, slides, transparencies, or material specifically designed for projection.
- i - Nonmusical sound recording
  Examples include: speech.
- j - Musical sound recording
  Examples include: phonodiscs, compact discs, or cassette tapes.
- k - Two-dimensional nonprojectable graphic
  Examples include: activity cards, charts, collages, computer graphics, drawings, duplication masters, flash cards, paintings, photonegatives, photoprints, pictures, photo CDs, postcards, posters, prints, spirit masters, study prints, technical drawings, photomechanical reproductions, and reproductions of any of these.
- m - Computer file
  Includes the following classes of electronic resources: computer software (including programs, games, fonts), numeric data, computer-oriented multimedia, online systems or services. For these classes of materials, if there is a significant aspect that causes it to fall into another Leader/06 category, the code for that significant aspect is used instead of code m (e.g., vector data that is cartographic is not coded as numeric but cartographic). Other classes of electronic resources are coded for their most significant aspect (e.g., language material, graphic, cartographic material, sound, music, moving image). In case of doubt or if the most significant aspect cannot be determined, consider the item a computer file.
- o - Kit
  Contains a mixture of components from two or more types of items, none of which is the predominant constituent of the kit.
- p - Mixed material
  Indicates that there are significant materials in two or more forms that are usually related by virtue of their having been accumulated by or about a person or body. Includes archival fonds and manuscript collections of mixed forms of materials, such as text, photographs, and sound recordings.
- r - Three-dimensional artifact or naturally occurring object
  Includes man-made objects, such as models, dioramas, games, puzzles, simulations, sculptures and other three-dimensional art works and their reproductions, exhibits, machines, clothing, toys, and stitchery, and naturally occurring objects, such as microscope specimens and other specimens mounted for viewing.
- t - Manuscript language material

- Three numbers indicating the Field Tag: As defined in MARC 21 specifications
- A single integer to indicate number of occurrence the Field Tag
- A single alpha character to indicate the Subfield Code: As defined in MARC 21 specifications
- A single integer indicating the offset within subfield: Using the following scheme: 1=first word in subfield, 2=second word in subfield; 3= third word in subfield, etc.
- A single integer indicating the version identification of the token: Differences in token versions may be based on Threshold of Occurrences, or some other discriminator
- A single alpha character for righthand padding: Agreement to use the letter “r” for this character.

And example token element to show this structure is: `rm2451a11r`. We can parse it as:

- r - lefthand padding
- m - type of record -- this is a Monograph-type record
- 2 - field code
- 4 –

- 5 –
- 1 – First occurrence of field in record
- a - Subfield code
- 1 - Offset within subfield -- 1=first word in subfield
- 1 - version identification or other discriminator
- r - Righthand padding

If there was a second instance of this field (in this example the 245), the token element for the second occurrence would be `rm2452a11r.` We can parse it as:

- r - lefthand padding
- m - type of record -- this is a Monograph-type record
- 2 - field code
- 4 –
- 5 –
- 2 - Second occurrence of this field in the record
- a - Subfield code
- 1 - Offset within subfield -- 1=first word in subfield
- 1 - version identification or other discriminator
- r - Righthand padding

The following is an example to show what sort of tokens (consisting of one or more token elements) might be created for an existing MARC record. The MARC record elements are listed in normal Courier font and the tokenized elements are in Courier bold:

```
LDR01019cam  2200265   4500^
Note: Leader/O6 has a code of “a” which means the type of record is Language material
001ocm00000003^
Note: The 001 will contain a unique local record number and will not be tokenized.
003OCoLC^
Note: Note: The 003 will contain a code for the number in 001 and will not be tokenized.
00520010925133908.0^
Note: The 005 will contain the date/time stamp and will not be tokenized.
008690414s1963    nyu      b    000 0 eng  ^
Note: The 008 and other fixed fields will not be tokenized.
010  _a   63064323 ^
Note: The 010 contains the Library of Congress Control Number and will not be tokenized (and likely not contained in the RadMARC records).
040  _aDLC _cDLC ^
Note: The 040 contains a coded value for the cataloging source and will not be tokenized (and likely not contained in the RadMARC records).
05004_aHV700.5 _b.N37 ^
Note: The 050 contains a Library of Congress Call Number and will not be tokenized (and likely not contained in the RadMARC records).
0820 _a362.7/3 ^
Note: The 082 contains a Dewey Decimal Classification Number and will not be tokenized (and likely not contained in the RadMARC records).

1102 _aNational Study Service. ^
Token Elements: rm110a11r rm110a21r rm110a31r
```

24510_aIllegitimacy and adoption in Maine : _breport of a study made for the Maine Committee on Children and Youth. ^
Token Elements: rm245a11r rm245a21r rm245a31r rm245a41r rm245a51r : rm245b11r rm245b21r rm245b31r rm245b41r rm245b51r rm245b61r rm245b71r rm245b81r rm245b91r rm245b101r rm245b111r rm245b121r rm245b131r

Note: In this case the offset integer is more than one character since there were 13 words in the actual MARC 245$a; in the RadMARC records, this value will also be 9 or less.

260 _a[New York], _c1963. ^
Token Elements: rm260a11r rm260a21r rm260ca11r

300 _a24 p. ; _c28 cm. ^
Token Elements: rm300a11r rm300ca11r

500 _aCover title. ^
Token Elements: rm500a11r rm500a21r

504 _aBibliographical footnotes. ^
Token Elements: rm504a11r rm504a21r

650 0_aIllegitimacy _zMaine. ^
Token Elements: rm6501a11r rm6501z11r

650 0_aAdoption _zMaine. ^
Token Elements: rm6502a11r rm6502z11r

7101 _aMaine. _bCommittee on Children and Youth. ^
Token Elements: rm710a11r rm710b11r rm710b21r rm710b31r rm710b41r rm710b51r

This is just an example. The RadMARC records' subfields will have a token comprised of no more than three token elements.

Using the Extended MARCdocs Data to Support RadMARC Record Creation

We propose to select data from the MARCdocs database to populate (either manually or in the future in an automatic way) the radioactive MARC records. Based on the discussion so far, the MARCdocs database will have appropriate date to assist in the creation of the RadMARC records.

The logic of this is as follows:

> Query MARCdocs to find all content designation that supports a specific search, where the content designation has a frequency count that meets an identified threshold of occurrence, and produce output that lists the content designation and the token for each specific content designation.

An example query might be: Find all content designation that supports BP02. The output of this query would is reflected in the following table:

Occurrence	Field	Subfield
419641	245	A
169178	245	b
64433	440	a
62853	490	a
32535	505	t

Occurrence	Field	Subfield
30443	830	a
30378	505	a
30173	245	h
29833	740	a
29669	490	v
23558	830	v
20166	240	a
17556	246	a
15417	700	t
10031	240	l
7743	505	g
6662	222	a
5805	780	t
5674	730	a
5529	810	t
5523	210	a
4924	810	v
4471	785	t
4075	800	t
4031	130	a
2784	245	p
2587	830	p
2425	830	n
2405	700	n
2231	776	c
2063	440	p
2026	240	k
1924	800	v
1881	240	n
1846	700	m
1819	700	p
1644	240	m
1492	245	n
1481	710	t
1229	776	t
1137	240	f
1133	700	r
1099	130	l
1082	787	t
1011	246	f
1007	240	r
963	740	h
962	222	b
947	130	p
843	210	b
805	240	h
778	440	n
703	730	p
663	700	o
608	246	p
590	770	t
561	775	t

Occurrence	Field	Subfield
558	730	l
512	240	s
496	730	f
463	130	f
429	240	o

Now, if we refine the query to ask only for the content designation that meets a specific threshold of occurrence, we could state it as: Find all content designation that supports BP02, where the occurrence of the content designation is greater than 100,000. The result would be:

Occurrence	Field	Subfield
419641	245	A
169178	245	b

We could decide to increase the threshold to greater than 50,000 and the result would be the following set of content designation:

Occurrence	Field	Subfield
419641	245	A
169178	245	b
64433	440	a
62853	490	a

This approach is attractive since we could rationally base our RadMARC contents to reflect empirical data from our previous content designation utilization. And it would allow us to work with a smaller number of fields/subfields.

In this case, we would create a MARC record that has only these four fields/subfields to populate. It would be relatively easy to handcraft the tokens for these and input them in the MARCdocs. Let's assume the following tokens associated with each of the fields/subfields (we don't know if these are the appropriate character strings for the tokens, but we will use them for this example):

245 $a: xxxxx yyyy zzzz
245 $b: qqqq rrrrrr sssss
440 $a: aaaa bbbb ccccc
490 $a: dddd eeee ffff

We could now do the query of the MARCdocs database: Find all content designation that supports BP02, where the occurrence of the content designation is greater than 50,000. And the output would be structured as follows:

Occurrence	Field	Subfield	Token in MARCdocs
419641	245	a	xxxxx yyyy zzzz
169178	245	b	qqqq rrrrrr sssss
64433	440	a	aaaa bbbb ccccc
62853	490	a	dddd eeee ffff

With such a structured output, it would be possible to transform the token into what would go into the appropriate RadMARC record field:

245axxxxx yyyy zzzz
245bqqqq rrrrrr sssss
440aaaaa bbbb ccccc
490adddd eeee ffff

In addition to the content designation selected from the MARCdocs that could support the specific profile-defined searches, it will be necessary to populate other MARC fields that are likely to be expected by a system such as the 001, 005, 010, etc. We will also populate a specific field in the RadMARC record that would indicate the version of this record (e.g., putting the following in a note field: RadMARC record for threshold of occurrence greater than 50,000) which allows us to control the versions of the RadMARC records. In addition, we will need to set the indicator values for the fields contained in the specific RadMARC records.

Using the Extended MARCdocs Data to Support Z-Query Creation

The same data could be used for the test script that will create the various Z-queries necessary to test interoperability. For example, we could provide to the test script the following:

Profile Search	Field	Subfield	Token in MARCdocs
BP02	245	a	xxxxx yyyy zzzz
BP02	245	b	qqqq rrrrrr sssss
BP02	440	a	aaaa bbbb ccccc
BP02	490	a	dddd eeee ffff

The test script could read the data structured in this table, recognize that this is to create test searches for the BP02 searches, and then create the appropriate four Z-queries:

(1,4)(2,3)(3,3)(4,2)(5,100)(6,1) 245axxxxx yyyy zzzz
(1,4)(2,3)(3,3)(4,2)(5,100)(6,1) 245bqqqq rrrrrr sssss
(1,4)(2,3)(3,3)(4,2)(5,100)(6,1) 440aaaaa bbbb ccccc
(1,4)(2,3)(3,3)(4,2)(5,100)(6,1) 490adddd eeee ffff

The output of the MARCdocs database query could easily help automate the creation of the Z-queries that would match the tokens in the query that are in the RadMARC record.

Summary and Conclusion

We think that the above approach to extending the MARCdocs database seems promising to help automate the RadMARC record creation and the Z-query creation. We want to make sure we have the tokens in the records that will be the search terms in the Z-query. The approach described seems to offer that.

The main thing will be to determine what the token recorded in the MARCdocs database for each field/subfield looks like. Thoughts on that are welcome.

Appendix M: Z-Interop2 Papers and Documents

The following is a list of papers and other documents that were developed in research and development to explore an alternative approach to interoperability testing through the use of radioactive metadata records. These are available on the project website.

Project Documents

Creating Radioactive MARC Records and Z Queries Using the MARCdocs Database. December 2, 2004. Available URL: <http://www.unt.edu/zinterop/ZInterop2/Documents/MARCdocsExtensionForRadMARCQueries2Dec2004.pdf>.

MARCdocs: The MARC 21 Bibliographic Format Database (Version 1.2) October 28, 2004. Available URL: <http://www.unt.edu/zinterop/ZInterop2/Documents/MARCdocs_docs_v1.2.pdf>.

Radioactive MARC Records Specifications. January 1, 2005. (Draft). Available URL: <http://www.unt.edu/zinterop/ZInterop2/Documents/RadMARCRecordsSpecifications1Jan2005.pdf>.

Z-Interop 2 Project Search & Record Data Requirements for Z39.50 Interoperability Testing Using Radioactive MARC Records. September 15, 2004 (Version 3 Draft). Available URL: <http://.edu/zinterop/ZInterop2/Documents/SearchDataRequirements_v3wem15sep2004.doc>.

Z-Interop2 Project Work Plan. June 2004. Available URL: <http://www.unt.edu/zinterop/ZInterop2/Documents/WorkPlanJune0024.pdf>

Z39.50 Interoperability Testing Framework for Online Library Catalogs Using Radioactive MARC Records. June 2004. Available URL: <http://www.unt.edu/zinterop/ZInterop2/Documents/InteropTestingFramework20June2004.pdf>.

Z-Interop2 Project Work Plan. June 2004. Available URL: <http://www.unt.edu/zinterop/ZInterop2/Documents/WorkPlanJune0024.pdf>.

Published Paper

Moen, William E., Hammer, Sebastian, Taylor, Mike, Thomale, Jason, and Yoon, JungWon. (2005). "An Extensible Approach to Interoperability Testing: The Use of Special Diagnostic Records in the Context of Z39.50 and Online Library Catalogs." In Andrew Grove (Ed.), Proceedings of the 68th ASIS&T Annual Meeting Volume 42. Silver Spring, MD: American Society for Information Science and Technology.

Presentations About Project

An Alternative Approach to Interoperability Testing: The Use of Special Diagnostic Records in the Context of Z39.50 and Online Library Catalogs. ASIS&T Annual Meeting, October 2005, Raleigh, NC.

Radioactive Metadata Records: An Interoperability Testing Approach Based on Metadata Utilization. Access 2005, October 2005, Edmonton, Alberta.

A Radioactive Metadata Record Approach for Interoperability Testing Based on Analysis of Metadata Utilization. Coalition for Networked Information 2005 Spring Task Force Meeting, April 2005, Washington, DC.

Appendix N: An Extensible Approach to Interoperability Testing: The Use of Special Diagnostic Records in the Context of Z39.50 and Online Library Catalogs.

Note: This paper was accepted for presentation at the ASIS&T Annual Meeting, and published in **Proceedings of the 68th ASIS&T Annual Meeting Volume 42, 2005.**

An Extensible Approach to Interoperability Testing: The Use of Special Diagnostic Records in the Context of Z39.50 and Online Library Catalogs

William E. Moen, Ph.D.
Texas Center for Digital Knowledge, University of North Texas
<wemoen@unt.edu>

Sebastian Hammer & Mike Taylor
Index Data, Copenhagen, Denmark

Jason Thomale & JungWon Yoon
Texas Center for Digital Knowledge, University of North Texas

Assessing interoperability in the networked information services and applications environment presents difficult challenges due in part to the multi-level and multi-faceted aspects of interoperability. Recent research to establish an interoperability testbed in the context of Z39.50 protocol clients and servers and online catalog applications identified threats to interoperability and defined a question space for interoperability testing. This paper reports on follow-up research to develop an alternative approach for interoperability testing in the context of networked information retrieval that uses specially designed diagnostic records. These records, referred to as radioactive records, enable interoperability assessment at the protocol and semantic levels. This approach appears to offer an extensible method for interoperability testing for other metadata and protocol application environments. The resulting interoperability testbed incorporates additional components to exploit automatic processes for interoperability testing and assessment, thus improving the efficiency of interoperability testing.

Introduction

Pursuit of interoperability in the networked information environment has been compared to the pursuit for the Holy Grail (Tennant, 1998). We believe it exists, and we believe we can find it (or achieve it). Testing for interoperability in basic networked services and applications such as information retrieval have often resembled the Keystone Kops in the simplicity of some approaches, or Rube Goldberg machines for conformance testing – far from the sublime pursuit of the Grail. Yet the challenges in achieving useful levels of interoperability are problematic in part because of the multi-faceted nature and types of interoperability (Miller, 2000).

One networked information service area that has provided an opportunity to explore the multi-faceted nature of interoperability is the use of the Z39.50 information retrieval protocol to conduct information retrieval tasks on a variety of online databases, including bibliographic databases associated with online catalog applications. We have seen how optimal interoperability must occur not only at the syntactic or functional level provided by the protocol but also at the semantic level. This latter level addresses ability of two systems to present and process user information tasks in a way that meanings of those tasks are retained. Reliability, trustworthiness, and usability of networked resources and services are founded on assumptions about the levels of interoperability occurring when two or more systems interact in service to applications and users.

This paper describes an on-going research project to explore issues related to interoperability in the context of metasearch applications across multiple online library catalogs or bibliographic databases. The immediate goal of this research is to improve interoperability when using the Z39.50 information retrieval protocol. The paper presents a new approach to interoperability testing through the use of specially-designed Machine-Readable Cataloging (MARC) records, which we call radioactive MARC (RadMARC) records.

Background

The literature on the topic of interoperability is both broad and deep, and continues to expand. From

brief overviews describing interoperability (see Miller, 2000) to more technical treatments (see Lynch & Garcia-Molina, 1995) to the implications of interoperability on policy (see Moen, 2001a), the literature treats interoperability from multiple perspectives.

The U.S. Federal Institute of Museum and Library Services (IMLS) awarded a National Leadership Grant to the Texas Center for Digital Knowledge in 2000 for a research project to explore the issues of interoperability among online library catalogs and their bibliographic databases accessible via the Z39.50 protocol. The overall goal of the Z39.50 Interoperability Testbed (Z-Interop) Project was to improve Z39.50 semantic interoperability among libraries for information access and resource sharing. Information about this phase of the research, including the full proposal and various reports, is available at: <http://www.unt.edu/zinterop>.

Several key components of the testbed included:

- **Test dataset:** Approximately 400,000 MARC 21 bibliographic records from OCLC's WorldCat database.
- **Reference implementations:** Reference implementations of a Z39.50 server and an information retrieval system (in the form of an online catalog) using an integrated library system from Sirsi Corporation. Reference implementation for a Z39.50 client using the Bookwhere Z39.50 client. The reference implementations were under the control of the Z-Interop project staff to configure according to published specifications in the form of Z39.50 profiles.
- **Test searches and benchmarks:** A set of test searches as defined in the Z39.50 profiles and benchmark results for each test search established by using the reference implementations.

Project staff assumed that the target audience for interoperability testing would be the vendors of Z39.50 client and server products, and individual libraries that wanted to check their systems interoperability with the reference implementations. Although the project was successful in recruiting participation from vendors of products, we discovered that individual libraries did not have the capability to load the 400K test dataset into their systems. Hardly any libraries have a test environment for their implementations, and this was a basic limitation of the Z-Interop Project's testbed approach. However, vendors of both Z39.50 servers and clients went through interoperability testing. The testbed proved fruitful in better understanding several factors that affect interoperability, and also demonstrated that with some attention by the vendors, their products could be configured to achieve 100% interoperability using the testing procedures provided by the testbed.

However, the true arena for interoperability testing is not just the vendors' products but their actual instantiation in a particular implementation, namely the implementation of the product as a production-level application in a library.

An Alternative Approach for Interoperability Testing

The limitations of the testbed approach described above motivated an investigation for an alternative method for interoperability testing for Z39.50 servers and library bibliographic databases. IMLS provided additional funding to continue the Z-Interop Testbed to explore this alternative approach.

The alternative method uses a small set of very special MARC records (we refer to these as "radioactive MARC records," explained below) that can serve as diagnostic mechanisms for assessing system functionality, performance, and interoperability. This alternative approach has potential for providing interoperability testing services to individual libraries. In addition, this approach may be adaptable to other protocol and metadata contexts beyond Z39.50 and MARC.

The metaphor of a radioactive MARC record is based on current medical diagnostic techniques for people. When a person has a particular medical condition, there may be two approaches for diagnosis. One could be considered invasive, where the person would undergo a surgical technique for physically examination of the problematic area or anomaly. The other approach could be considered less invasive, where the patient is injected with a dye, possibly radioactive, and once it has spread throughout the body, scanning techniques allow a medical professional to identify structural or mechanical problems or anomalies.

A "radioactive" MARC record approach for interoperability testing is less "invasive" for an individual library. It does not require loading a large test dataset such as used in the first Z-Interop testbed. Nor does it require a separate testing environment on the local implementation. Instead, the library loads these special MARC records into its production online catalog system, and the Z-Interop staff conducts a series of tests to assess system functionality, performance, and interoperability. The radioactive MARC records are legitimate instances of MARC records that a library system can import and process, and then remove when the testing is

completed. These records, however, have very special characteristics.

The Threats to Interoperability

In the first phase of the Z-Interop testbed, we anticipated several levels at which interoperability needs to occur, and we identified some of the threats to such interoperability. The research in that project confirmed the reality of these threats.
In a broader context, Moen (2001b) identified a number of diverse factors that can affect interoperability in networked information retrieval applications:

- Multiple and disparate operating and Information retrieval systems
- Multiple protocols
- Multiple metadata schemes
- Multiple data formats
- Multiple languages and character sets
- Multiple vocabularies, ontologies, and disciplines.

In the context of the Z-Interop Project, we identified key factors threatening interoperability:

- Differences in implementation of the standard
- Differences in local information retrieval systems

In the latter case, this includes search functionality available in the system, indexing policies affecting the access points in the database, word extraction and processing choices, and character set and character encoding and normalization. As a way to indicate the scope of our investigations, Moen (2001a) identified the levels of interoperability of concern in the Z39.50 context:

- Low-level protocol (syntactic): Do Z-client and Z-servers interchange protocol messages according to the standard?
- High-level protocol (functional): Do Z-client and Z-servers support appropriate Z39.50 information retrieval services for user tasks?
- Semantic level: Can Z-clients and Z-servers and local information retrieval systems preserve and act on meaning of information retrieval tasks?
- User Task level: Do systems support information retrieval tasks of one or more user groups?

Within the context of our investigations and the maturity of Z39.50, the syntactic level is of little concern. The development of the Bath Profile: An International Z39.50 Specification for Library Applications and Resource Discovery (2004), and the U.S. National Z39.50 Profile for Library Applications (National Information Standards Organization, 2003) addressed many issues related to the functional level. The Z-Interop testbed was successful in part because the profiles defined expected Z-client and Z-server behaviors and interactions. The biggest challenge to reliable interoperability appears at the semantic level. Semantic interoperability here is not addressing the concerns of two words meaning the same thing or other problems related to linguistics and meaning. Instead, semantic interoperability concerns the ability of two systems to present and process user information tasks in a way that meanings of those tasks are retained. For example, if a user does a title search for information resources, the search is actually executed on a search target against words from titles in the record. A common sense idea, yet often search targets do not process searches as the user intended (e.g., processing an exact match search for a title as a set of keywords combined using Boolean operators and matching the words not only in title access points but in other access points as well).

The Question Space for Interoperability Testing

In our alternative approach for interoperability testing we identified a set of questions that could be asked to address the different levels of interoperability. These questions pointed to appropriate test searches and the data needed in the records. The following summarizes the question space for our interoperability testing:

- **Profile conformance level:** Addresses the interoperability between the Z-client and Z-server. Assessing this level of interoperability relies on the use of Z39.50 profiles that identify Z39.50 specifications for search and retrieval. Questions that can be addressed at this level include:
    - Does the Z-server process each query successfully?
    - If the Z-server cannot process the query as sent, does it send the appropriate diagnostic message?
- **Information retrieval (IR) system level:** Addresses the capability of the IR system underlying the online catalog application. Questions that can be addressed at this level include:
    - Does the IR system have the requisite search functionality to support the searches defined in the Z39.50 profiles?

- **Metadata record level:** Also an IR system focus, but concerned with how the IR system indexes fields in the metadata record to provide access points or searchable components of the record. Questions address by this level include:
    - Does the information retrieval system index the appropriate fields in the records for specific access points?
    - Do the system's indexing policies support searches for the searches defined in the Z39.50 profile?
- **Data content level:** Addresses how the IR system processes the data content of the records, such as processes related to normalization of the data, dealing with hyphenated works, and special characters and diacritics.

The question space is also informed by two Z39.50 profiles:

- ANSO/NISO Z39.89, The U.S. National Z39.50 Profile for Library Applications (National Information Standards Organization, 2003) <http://www.niso.org/standards/resources/Z39_89final.pdf>
- Bath Profile: An International Z39.50 Specification for Library Applications and Resource Discovery, Release 2.0 (The Bath Group, 2004)

These specifications provide well-defined searches and expected client and server behaviors at several conformance levels. For initial interoperability testing, we used the profile-defined searches listed Table 1. These searches also pointed to the data necessary in the RadMARC records to support the testing using these searches.

Table 1. Z-Interop Testbed Search Types Approximately Here

Components of the Radioactive MARC Record Interoperability Testbed

In addition to the reference implementations used in the original Z-Interop testbed, the alternative testing approach introduces three new components:

- The specially designed MARC records
- A set of test searches and automatic testing script that issues searches, retrieves records, and develops reports on the search and retrieval results
- A database of MARC documentation that enables the automatic identification of types of searches to issue.

The basic interoperability testing framework is illustrated in Figure 1 to highlight key components and processes.

Figure 1. Interoperability Testing Framework Approximately Here

Radioactive MARC Records

As noted before, two Z390.50 profiles provide the specifications that are the basis for the test searches. Searches defined at Level 0 and Level 1 require appropriate RadMARC records for the test searches. In addition, different types of records (as indicated in the MARC Leader/06) are needed since systems may index MARC fields/subfields differently depending on the type of record. Finally, the RadMARC records need to be clearly identified as to the type of searching they are intended to assess, and therefore some version information about each record is included in the record.

MARC records can describe different types and formats of bibliographic materials. MARC Leader/06 indicates the type of record (i.e., the information object type being described by the MARC record). The *Anglo-American Cataloguing Rules* (AACR), the standard for descriptive cataloging, specifies rules for describing different types of materials. MARC Leader/07 indicates the bibliographic level of the record. Table 2 summarizes the values of the MARC Leader/06, Leader/07, and the AACR categories of materials to show the complexity of coding and labeling for what the MARC records describe.

The coding of Type of Record in MARC Leader/06 is not aligned directly with the 10 format types of information objects as addressed by AACR (in the third column of the table above). In some cases, two code values in the Leader/06 are addressed by the same AACR format of material. Additionally, the Leader/06 doesn't indicate if the material is a serial publication (or Continuing Resource). For this, the Leader/07 – Bibliographic Level indicates serial with a value of s. To summarize, the types of records or materials for which RadMARC records were created address those in the table.

Table 2. Types of Materials Described by MARC Bibliographic Records Approximately Here

The specially constructed MARC records for this approach to interoperability testing are the foundation, and the design of these records was a key intellectual challenge. The fundamental data unit in the RadMARC records is a token. A token is a string of characters that has a specific structure and semantics that will serve as “words” or other data values in specific fields/subfields. A field/subfield may have a sequence of tokens. The specially designed tokens populate selected field/subfields in the RadMARC records. Several sets of RadMARC records are used in interoperability testing. The sets are distinguished by the amount of content designation populated in the records (see discussion below). All selected content designation use the special tokens. The following is the structure of content-rich tokens being used in the RadMARC records:

- A single alpha character for left-hand padding.
  - Value = r
- A single alpha character to indicate the format of the material being described or type of record
  - Value = Selected values as defined in MARC Leader/06 – Type of Record or the Leader/07 – Bibliographic Level
- Three numbers indicating the Field Tag
  - Value = Defined in MARC 21 specifications
- A single integer to indicate number of occurrence the Field Tag
  - Value = Sequential number starting with 1
- A single alpha character to indicate the Subfield Code
  - Value = Defined in MARC 21 specifications
- A single integer indicating the offset within subfield
  - Value = Use the following scheme: 1=first token in subfield, 2=second token in subfield; 3= third token in subfield, etc.
- A single alpha character for right-hand padding
  - Value = r

An example token that shows this structure is **ra2451a1r**, which can be parsed as:

- r - Left-hand padding
- a - Type of record -- this is a books type record
- 245 - Field code
- 1 – First occurrence of field in record
- a - Subfield code
- 1 - Offset within subfield, where 1 = first token in subfield
- r - Right-hand padding

In addition to the field- and subfield-specific tokens, each RadMARC record contains additional information to uniquely identify the record, the version of the record, and other details about the source and purpose of the record. The following is an example of a RadMARC record in human-readable form built according to the specifications.

Figure 2. Sample RadMARC Record Approximately Here

Automated Testing Scripts and Processes

Once a server's database has been injected with one or more radioactive records, a client can test that server's indexing and searching functionality by issuing searches that expect to return specific records. For example, a server that contains a record with a particular token in its 245$a should yield the record when queried with a search for that token against a title index, and using the appropriate Z39.50 query to express the query (e.g., as defined by Bath profile "title keyword” query). Conversely, so long as the same token does not appear elsewhere in the record, a search for that token in a subject index should *not* find the record.

Test searches such as these may be sent by any conforming Z39.50 client, but it is more efficient to automate testing using ready-rolled scripts. We took a two-level approach to building such scripts: at the low level, we created a domain-specific “little language” specialized for such scripts; and at the higher level, we created an initial set of scripts in that language, both as a useful partial test-suite for servers claiming Z39.50 profile conformance, and as proof of concept of the language/script division. Although initial designs for the scripting language consisted of only a few domain-specific primitives, it quickly became apparent that scripts may in general need to make use of logical and looping constructs, and perhaps variable assignments and procedure definition/invocation, such as are provided by mainstream programming languages. Accordingly, we decided that the most efficient approach would be to build our language on top of a well-supported,

expressive, existing language. Practical considerations indicated that Perl was the most appropriate choice, although Python would have been an attractive alternative were it better appreciated and more widely adopted in the Z39.50 community.

Our strategy, then, was to extend Perl with a "RadioMARC" module to allow Z39.50 searching of servers known to contain copies of specific records, and to emit reports dependent on whether or not the expected records are present in the result set. Perl's own language constructs are used in more complex test scripts to determine at run-time which tests to attempt, depending on the results of earlier tests. A typical simple script follows:

```
use Net::Z3950::RadioMARC;
set host => 'z3950.loc.gov', port =>
'7090', db => 'voyager';
set delay => 3;
add "filename.marc";
test '@attr 1=4 01245a01', { ok =>
'245$a is searchable as 1=4',
 notfound => 'This server is broken'
 };
```

This illustrates the three important domain-specific operations, **set**, **add** and **test**. Once the RadioMARC module has been introduced (the "use" statement on the first line), these may be freely used:

- **set** merely sets the value of named parameters – in this case, the connection details for the server to be tested, and the number of seconds to delay between searches in order to avoid overloading the server.
- **add** registers a set of MARC records, added from the named file, which are believed to exist in the server being tested.
- **test** does the real work. First, it creates the Z39.50 connection if no connection has already been forged. Then it performs the search specified as its first argument. This argument expresses the query in the widely used Prefix Query Format (PQF), as described in the *YAZ User's Guide and Reference* (Hammer, et al., 2004), The same query is used on the client side to select which of the previously **add**ed records is the target for the query, and the result set returned by the server is inspected for that record's presence. A message is emitted depending on whether or the record is found, or whether the search failed completely – for example, because the server does not support the specified access-point.

As part of the deliverables from this research project, the RadioMARC Perl module will be released for public use.

MARC Documentation Database

One of the challenges of creating a sustainable interoperability testing environment is to identify potential components of a testbed than can support the automation of activities and procedures. The previous section discussed one aspect for automatic testing software that formulates appropriate test searches, issues those to specific search targets, gathers results, and produces reports. To support those functions we developed a database of MARC documentation that would serve multiple purposes The database stores information about all content designation available in the MARC 21 Format for Bibliographic Data specifications. In addition, the flexible and extensible structure of the underlying relational database allows the storage of information about profile-defined searches necessary to the automatic testing software. Further, we examined how the database could assist in the creation of the radioactive MARC records.

MARCdocs: The MARC 21 Documentation Database, is a pilot effort aimed at structuring the textual documentation from the MARC 21 Format for Bibliographic Data into a relational database. Using a database approach for authoritative MARC documentation provides new opportunities for various applications. This database application uses open source software tools including Linux, MySQL, and PHP. A public version of the application is available at: <http://meta.lis.unt.edu/MARCdocs2/>. A working version of MARCdocs for our current research contains additional project-specific information and is not publicly accessible. Figure 3 is a screen shot showing example MARC field information in the database. Having this documentation, along with other project-specific data included in structured format in the database provides opportunities for automating many aspects of the testbed.

Figure 3. MARCdocs Database Interface Approximately Here

The RadMARC Content

In the discussion above about the RadMARC records, we indicated that the need to know what content designation in a MARC record to populate

with tokens to support the interoperability testing of profile-defined searches. This is tied closely with a specific level of the question space for this interoperability testbed, namely:

- **Metadata record level:** This level focuses on how the information retrieval system indexes fields in the metadata record to provide access points or searchable components of the record. Questions address by this level include:
    - Does the information retrieval system index the appropriate fields in the records for specific access points?
    - Do the system's indexing policies support searches for the searches defined in the Z39.50 profile?

As part of the original Z-Interop testbed, we identified more than 500 MARC fields/subfields in a MARC record that could be indexed to support author, title, and subject searching. The complete list of this content designation is contained in *Indexing Guidelines to Support Z39.50 Profile Searches* (Moen, 2002). In another analysis for that project, we analyzed occurrences of MARC content designation in the Z-Interop test dataset of more than 400,000 MARC records from OCLC's WorldCat database (Moen and Benardino, 2003). We also examined the occurrence of content designation that could be indexed to support author, title, and subject searches, and discovered that 19 of the more than 500 subfields that could be indexed accounted for 80% of all occurrences. Table 3 shows the top 5 of these 19 subfields.

Table 3. Top 5 MARC Indexable Subfields Approximately Here

The data resulting from that analysis were added to the MARCdocs database, which enables us query the database and identify content designation that could be indexed to support the Z39.50 profiles Level 0 and Level 1 searches. We use those frequency counts to select sets of fields/subfields to populate various sets of RadMARC records. The MARCdocs database, then, is used in the creation of the RadMARC records by holding information that designates a specific content designation as a candidate for indexing for particular searches and the frequency count of its occurrence based on the earlier analysis.

RadMARC Record Sets

We have identified several possible sets of RadMARC records to create and have completed the creation of two of these. The first set uses the 19 most commonly occurring indexable fields for author, title, and subject-related data discussed previously. The RadMARC record in Figure 2 shows how these content designation structures have been populated. The second set of RadMARC records uses all author, title, and subject content designation that occurred 1,000 or more times from the earlier analysis. However, we can extend this to include all possible content designation as listed in the Z-Interop *Indexing Guidelines* document.

Two other sources of information are informing the creation of a third set of RadMARC records:

- The Network Development and MARC Standards Office (n.d.) recommendations for national level records
- The Program for Cooperative Cataloging (2003) core record standards.

For example, the recommendations for national level records identify "mandatory" and "mandatory if applicable" content designation. A comparison of those recommended content designation structures with the Z-Interop indexing guidelines indicate 131 fields/subfields that are author, title, and subject related. We can create RadMARC records using these 131 fields/subfields.

We think that the RadMARC approach can be used to develop any set of RadMARC records as well as custom built diagnostic records libraries can use to interrogate their systems' behavior. We are currently in discussions with a number of libraries that want to diagnose the indexing policies actually in effect on their systems to verify vendor configurations. We can create individual RadMARC record that are intended to exercise specific indexing policies.

The Extensibility of the Radioactive Record Approach

To date (Spring 2005), we have proved out the concept and the technologies involved with the RadMARC approach to interoperability testing using Z39.50 clients and servers and online catalog applications. We have identified additional sets of RadMARC records that can be created to analyze more deeply information retrieval system search functionality and indexing policies. The radioactive record approach, however, has the potential to be used to diagnose other system behaviors, for use in other metadata environments, and for use in other protocol environments. We discuss three potential uses below.

In Section 5 we described the question space for the interoperability testbed. The data content level is not being addressed in the current testbed because of

resource constraints. However, it will be possible to create RadMARC records where the tokens include special characters and diacritics, which can then show how a local IR system normalizes or otherwise processes the data that may affect search results.
Another opportunity is in the Open Archives Initiative (OAI) metadata harvesting environment. One of the issues that OAI data providers face is mapping a rich native metadata scheme to simple Dublin Core metadata elements. In the case where library catalogs are being harvested, it would be interesting to see the effects of such local mapping decisions when the source records are specially designed RadMARC records.
Still another opportunity is to explore the creation of RadDC records (radioactive Dublin Core) or radioactive records using other metadata schemes to assist in diagnosing and making visible system behaviors in those application areas.
Finally, new protocols such as Search and Retrieve for the Web (SRW/SRU) could benefit from the current work by providing RadMARC records (or records in other metadata schemes) for information retrieval systems accessible by SRW. For example, a SRW search of a database of bibliographic records (possibly in MARC format) with a request to return results as DC records would allow diagnosis of various system behaviors (search access, search functionality, mapping from native database scheme to DC).

Summary and Conclusion

The current research project is establishing an innovative conceptual and technical foundation for interoperability testing. The scope of this research focuses currently on Z39.50 and online catalogs. The project has provided the opportunity to conduct proof-of-concept for a radioactive record approach for diagnosing interoperability factors in an identified question space. We see this approach as extensible in terms of the current focus (i.e., being able to create different sets of RadMARC records to diagnose general or specific interoperability issues) and to other application environments, metadata schemes, and protocols.

ACKNOWLEDGMENTS

The research for this project has been supported by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services.

REFERENCES

The Bath Group. 2004. The Bath Profile: An International Z39.50 Specification for Library Applications and Resource Discovery, Release 2.0. Available URL: <http://www.collectionscanada.ca/bath/tp-bath2-e.htm>

Hammer S. et al. 2004. YAZ User's Guide and Reference. Available URL:

<http://www.indexdata.dk/yaz/doc/>

Lynch, C. & Garcia-Molina, H. 1995. Interoperability, scaling, and the digital libraries research agenda: A report on the May 18-19, 1995 IITA Digital Libraries Workshop. Available URL: <http://www-diglib.stanford.edu/diglib/pub/reports/iita-dlw/>

Miller, P. 2000. “Interoperability: What it is and why should I want it.” Ariadne, 24. Available URL: <http://www.ariadne.ac.uk/issue24/interoperability/intro.html>

Moen, W.E. and Benardino, P. 2003. “Assessing metadata utilization: An analysis of MARC content designation use.” In 2003 Dublin Core Conference: Supporting Communities of Discourse and Practice – Metadata Research and Application. Seattle, WA, September 28-October 2, 2003. Seattle: Information School of the University of Washington, 2003.

Moen, W.E. 2002. Indexing Guidelines to Support Z39.50 Profile Searches. Texas Center for Digital Knowledge, University of North Texas. Available URL: <http://www.unt.edu/zinterop/Documents/IndexingGuidelines1Feb2002.pdf>

Moen, W.E. 2001a. “Assessing interoperability in the networked environment: Standards, evaluation, and testbeds in the context of Z39.50.” In Evaluating Networked Information Services: Techniques, Policy, and Issues (pp.85-109), edited by Charles R. McClure and John Carlo Bertot, Publisher: ASIS thru Information Today, Inc. 2001.

Moen, W.E. 2001b. "Mapping the Interoperability Landscape for Networked Information Retrieval." In Proceedings of First ACM/IEEE-CS Joint Conference on Digital Libraries, Roanoke, VA, June 24-28, 2001 (pp. 50-52). New York: The Association for Computing Machinery.

National Information Standards Organization. 2003. ANSI/NISO Z39.89 - 2003 The U.S. National Z39.50 Profile for Library Applications. Bethesda, MD: National Information Standards Organization. Available URL: <<http://www.niso.org/standards/resources/Z39_89final.pdf>

Network Development and MARC Standards Office (n.d.). National Level Record---Bibliographic Full Level & Minimal Level. Washington, DC: Library of

Congress. Available URL: <http://www.loc.gov/marc/bibliographic/nlr/>

Program for Cooperative Cataloging. (2003). Chart of PCC BIBCO Core Record Standards. Available URL: <http://www.loc.gov/catdir/pcc/bibco/core2002.html>

Tennant, R. 1998. Interoperability: The Holy Grail. Library Journal. Available URL: <http://www.libraryjournal.com/article/CA156495>

Tables and Figures to Insert in:

An Extensible Approach to Interoperability Testing: The Use of Special Diagnostic Records in the Context of Z39.50 and Online Library Catalogs by William E. Moen, Ph.D., et al.

Table 1. Z-Interop Testbed Search Types

Level 0 Searches	Level 1 Searches
Author Search – Keyword	Author Search – Keyword with Right Truncation
	Author Search – Exact Match
	Author Search – First Words in Field
	Author Search – First Characters in Field
Title Search – Keyword	Title Search – Keyword with Right Truncation
	Title Search – Exact Match
	Title Search – First Words in Field
	Title Search – First Characters in Field
Subject Search – Keyword	Subject Search – Keyword with Right Truncation
	Subject Search – Keyword with Right Truncation
	Subject Search – Exact Match
	Subject Search – First Words in Field
	Subject Search – First Characters in Field
Any Search – Keyword	Any Search – Keyword with Right Truncation

Table 2. Types of Materials Described by MARC Bibliographic Records

Leader/06 Code	Semantics	AACR Categories of Materials
a	Language material	Books, Pamphlets, and Printed Sheets
c	Notated music	Music (Notated and manuscript music)
d	Manuscript notated music	Music (Notated and manuscript music)
e	Cartographic material	Cartographic Materials
f	Manuscript cartographic material	Cartographic Materials
g	Projected medium	Motion pictures and video-recordings (including digital and non-digital)
i	Nonmusical sound recording	Sound Recordings (musical and non-musical)
j	Musical sound recording	Sound Recordings (musical and non-musical)
k	Two-dimensional nonprojectable graphic	
m	Computer file	Electronic Resources
o	Kit	
p	Mixed material	Graphic materials (includes mixed materials, with or without archival control)
r	Three-dimensional artifact or naturally occurring object	Three Dimensional Artifacts and Realia
t	Manuscript language material	Manuscripts (including manuscript collections)
Leader/07 Code	**Semantics**	**AACR Categories of Materials**
s	Serial	Continuing Resources

Table 3. Top 5 MARC Indexable Subfields

# of Occurrences	Marc 21 Field	Subfield	Description	Index
602,362	650	a	Subject added entry Topical Term Subfield a = Topical term or geographic name as entry element	Subject
419,641	245	a	Title Statement Subfield a = Title	Title
329,796	245	c	Title Statement Subfield c = statement of responsibility	Author
326,867	650	x	Subject added entry Topical Term Subfield x = General subdivision	Subject
318,692	100	a	Main entry Personal Name Subfield a = personal name	Author

Figure 1. Interoperability Testing Framework

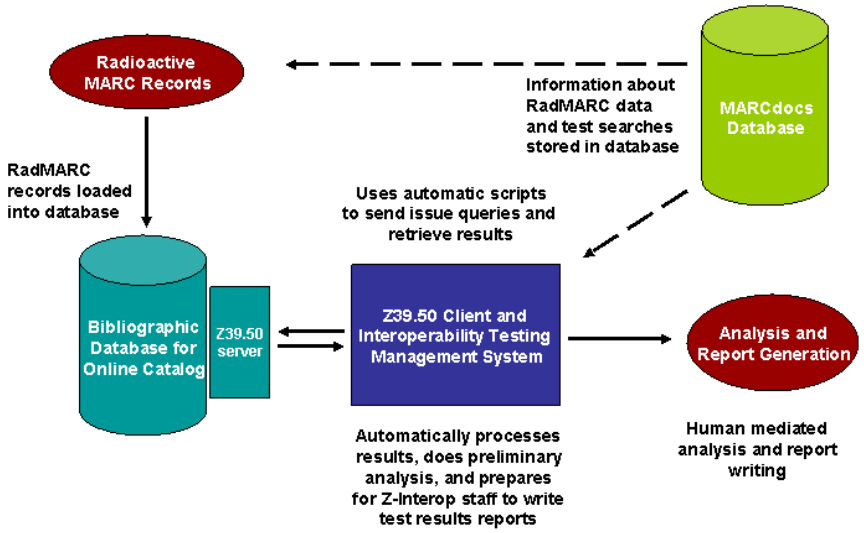


Figure 2. Sample RadMARC Record

001	UNTRadMARC001
040	$a ZinteropUNT
100	$a rm1001a1r, rm1001a2r, $d rm1001d1r.
245	$a rm2451a1r rm2451a2r rm2451a3r : $b rm2451b1r rm2451b2r rm2451b3r / $c rm2451c1r rm2451c2r rm2451c3r.
440	$a rm4401a1r rm4401a2r rm4401a3r
490	$a rm4901a1r rm4901a2r rm4901a3r
583	$a RadMARC $b www.unt.edu/zinterop/001 $d 1 $e ATS $i 1 $k JungWon Yoon $x This is a specially created test record for the Z-Interop2 Project under the direction of Dr. William E. Moen at the Texas Center for Digital Knowledge, University of North Texas. Contact Dr. Moen via email <wemoen@unt.edu> for information about this project. Funding for this project is provided by a National Leadership Grant from the U.S. Federal Institute of Museum and Library Services. This particular record supports testing related to a Books, Pamphlets, and Printed Sheets type of MARC record. The record support test searches for author, title, subject, and any Bath and U.S. National Z39.50 profile-defined searches, Levels 0 and 1, where the threshold of occurrence of the indexable content designation being populated in the record is 19, for the 19 most commonly occurring indexable author, title, and subject fields discovered in a separate

	analysis. This is the first version of this record.
600	$a rm6001a1r rm6001a2r, $d rm6001a1r.
650	$a rm6501a1r rm6501a2r rm6501a3r $x rm6501x1r $v rm6501v1r $z rm6501z1r.
651	$a rm6511a1r rm6511a2r $x rm6511x1r.
653	$a rm6531a1r rm6531a2r rm6531a3r
700	$a rm7001a1r rm7001a2r, $d rm7001d1r.
710	$a rm7101a1r rm7101a2r.

Figure 3. MARCdocs Database Interface