This project is made possible through the generous support of
the **National Science Foundation** and the **Deutsche Forschungsgemeinschaft**

A DISTRIBUTED DIGITAL LIBRARY OF MATHEMATICAL MONOGRAPHS

***A collaborative project of the University of Michigan Library,
Cornell University Library, and the State and University Library Göttingen***

Comments and Questions

A DISTRIBUTED DIGITAL LIBRARY OF MATHEMATICAL MONOGRAPHS

A collaborative project of the University of Michigan Library, Cornell University Library, and the State and University Library Göttingen

PROJECT SUMMARY

This project will create a genuinely distributed repository of significant historical monographs in mathematics. The participation of these three institutions—The State and University Library Göttingen, Cornell University Library, and the University of Michigan Library—is significant not only because of their pioneering work in building digital libraries and conversion techniques, but because of their extraordinary collections in this area. Göttingen is recognized as a wellspring of mathematical thought, and its collections are unparalleled in this area. Michigan and Cornell have built two of the strongest mathematical collections in the United States, a strength long recognized by the mathematics community and reflected in the Research Libraries Group collections "Conspectus." Cornell has digitized 576 volumes of mathematical monographs, and will generate OCR to enhance access to their materials. Michigan will fund the bulk of the conversion of an additional 1,000 monographic volumes focusing on non-Euclidean geometry from its collection. The State and University Library Göttingen will contribute digitized monographs, dissertations and multivolume works of the electronic Mathematical Archive and the database "Jahrbuch über die Fortschritte der Mathematik," funded by the Deutsche Forschungsgemeinschaft.

Funding is sought primarily to develop an interoperability layer with the three strong digital library systems at these institutions. Each institution has robust online access mechanisms. The State and University Library Göttingen relies on the Agora system, combining rich metadata mechanisms and strong architectural components. The University of Michigan Library has developed full-text access mechanisms that it now distributes through its Digital Library eXtension Service, and which current supports the Michigan Making of America system, soon to hold approximately 3 million pages. Cornell University Library has delivered its text based collections through various systems, including those built on top of Dienst and those using the University of Michigan middleware, and within the frame of the funding will work to extend the Dienst-based Euclid system. We will work cooperatively to modify and extend these three architectures as we apply the current XML-aware version of the Dienst system to provide a high level of interoperability. The depth of the proposed collection and the experience of the participants will allow the project to focus on many of the domains within the scope of the call for proposals, including distributed repositories, advanced access and retrieval, high levels of interoperability, and encouragement of the free flow of information. Project funding will result in three important outcomes: the improvement of three key digital library efforts through improved sharing mechanisms, a large and important collection of historical mathematical materials, and a valuable case study in integrating distributed resources.

A DISTRIBUTED DIGITAL LIBRARY OF MATHEMATICAL MONOGRAPHS

A collaborative project of the University of Michigan Library, Cornell University Library, and the State and University Library Göttingen

PROJECT DESCRIPTION

Interoperable Access to Multiple Systems across a Broad Discipline

The State and University Library Göttingen, Cornell University Library, and the University of Michigan Library propose the creation of a digital library of historical mathematical literature, enhancing already rich, standards-based digital library systems at each of the institutions with mechanisms for interoperability. Together, the three institutions will digitize or enhance already digitized resources to build a combined collection of nearly 2,000 volumes of mathematical literature from the 19th and early 20th century, as well as of dissertations and related materials. The three institutions commit themselves to ongoing support of the system and collection. Using robust digital library systems from each of the institutions, access will be enhanced by introducing an interoperability layer that capitalizes on recent developments in the Dienst protocol in order to produce a unified “virtual” collection.

Research libraries have been undertaking important digitization projects for several years, and we believe that these collections are underutilized because the information—even within a single discipline—is distributed throughout a number of different repositories, and is thus fragmented. Through our separate efforts, many large, unique, and essential collections are available online to benefit students and researchers. Historical literature is being digitized by many different institutions and is accessed in unique systems at those separate institutions, each offering a different user interface, a different navigational methodology, and various levels of functionality. Only in rare cases are these collections linked. Consequently, users have difficulty both in discovering and using this literature online. They must discover (and maintain a knowledge of) the different repositories, and they must become proficient in using the different systems. Users must also contend with a lack of efficient, distributed printing options, a problem made especially significant with book-length materials, especially when these books are otherwise out-of-print or inaccessible.

Digital library efforts in research libraries have long recognized the need to develop interoperable mechanisms, but have found no viable options for the relatively complex, standards-based material being created. Work in the Digital Library Federation, for example, has focused on interoperability for several years, without identifying satisfactory mechanisms to achieve this end. And while it is certainly unrealistic to assume that different institutions, with different needs, resources, and users, will adopt a common access and delivery system, there is a clear sense of the need for mechanisms that heterogeneous systems can use to communicate with each other. By defining and implementing an effective level of interoperability, we can aid users by creating “meta-repositories” and uniform access mechanisms for focused subject collections such as that proposed here.

Libraries and librarians have a demonstrated a fundamental interest in the development of mechanisms to provide users with unified access to resources; similarly, they have shown a strong and continuing interest in building digital libraries with long-term viability. The goal of this project is to contribute to the fundamental knowledge required to achieve meaningful interoperability. In the course of the project, we will develop a mechanism capable of unifying standards-based digital materials from a single discipline (i.e., mathematics) distributed among different access systems at different research institutions. After a thorough testing and evaluation, the system will be maintained by the three libraries to allow users to access and exploit these collections in a new and more efficient way, and the results of our work (i.e., an interoperability layer for each library’s digital library system) will be shared broadly.

Partnership of Three Institutions

As a focus for our efforts in the area of interoperability, we will concentrate on historical monographs in mathematics, an area for which the three participating institutions have extraordinary collection strengths and digital library activities.

State and University Library Göttingen

Context

Since the founding of the Library in 1734-37, mathematics has been one of the core collections of the Library and has always been supported by extra funds of the State and generous gifts of famous scientists of the University. For example the splendid scientific library of Gauss is now part of the library, and the mathematicians Klein and Hilbert have always requested special funds for the mathematical collections of the central library during their negotiations with the Preussian Ministry. Springer Verlag is one of several publishers that supported the Göttingen Library in previous years, and as a consequence, the Göttingen mathematical collection is now the strongest one in Europe. The central library holds approximately 60,000 mathematical monographs and about 1,100 current mathematical journal titles, as well as older journal volumes, mathematical preprints, and mathematical dissertations. In addition to the large collections in the Central Library, Göttingen holds the famous Mathematical Library of the Mathematical Department, founded as the “Mathematische Lesezimmer” by Felix Klein, and these volumes too can be used for digitization purposes.

Digital Collection

The Göttingen State and University Library began its digitization of mathematical literature in 1997. Project funding from ERAM (Electronic Research Archive for Mathematics), along with support from the DFG, made it possible to convert the journal “Jahrbuch über die Fortschritte der Mathematik” (1868- 1943) into a database. The ERAM project began in 1998 and is funded through 2003. During the first years of the project, we implemented the “Jahrbuch” and have now completed 80% of the conversion. The ultimate aim of the project is to create a digital library for mathematics, storing the most relevant publications from each period electronically. This full-text portion of the archive will be created over the next six years and will contain approximately 1,200,000 pages of mathematical literature. The titles to be digitized will be chosen by mathematicians who are reviewing parts of the “Jahrbuch über die Fortschritte der Mathematik.” Their criteria will result in all types of mathematical literature being digitized, including monographs, journal articles, dissertations and conference proceedings. Several hundred monographs from the ERAM project will be part of the distributed digital library of mathematical monographs.

University of Michigan

Context

The Michigan and Cornell mathematical collections are among the strongest in the United States, a strength long recognized by the mathematics community and reflected in the Research Libraries Group collections “Conspectus.” The mathematics collection at the University of Michigan is one of the earliest founded collections at the university and one of the most comprehensively developed. In the late 19th and early 20th centuries, the library received many gifts of rare and core mathematics books and journals from university faculty who traveled to Europe to collect mathematics literature. The library established many long-standing exchange and gift agreements with mathematics institutions around the world. The university has traditionally provided generous funding for the mathematics collection. The strength of the mathematics collection was a factor in Mathematical Reviews’ decision to move their offices to Ann Arbor in 1964, from Providence, Rhode Island and Brown University. The library currently holds approximately 33,000 bound mathematics serial volumes, and 32,000 mathematics monographs. These numbers do not include the mathematics titles held in the University Library Special Collections Library or in its engineering collection. Many of the key works in the development of non-Euclidean Geometry, as outlined in histories and bibliographies of non-Euclidean geometry, are held by the University of Michigan Library.

Digital Collection

The University of Michigan will fund the digitization of a thousand monographic volumes in mathematics and will contribute this material for the purposes of the proposed project. A tentative list of items has been prepared and is being reviewed by scholars in appropriate fields. The mathematics monographs suggested for digitization share the following characteristics:

- Held by the University of Michigan University Library
- Published between 1803 and 1923
- Printed on brittle paper (many volumes identified as brittle during mass-deacidification of mathematics collection, 1999-2000)
- Not currently available in digital form
- Printed in English, French, German, Dutch, Russian, Spanish, or Italian
- Are in bound book format
- Works of mathematicians who contributed to the development of non-Euclidean geometry (authors are included in the Bibliography of Non-Euclidean Geometry by D.M.Y. Sommerville, 2nd edition, Chelsea Publishing Company)

Although authors in the Michigan list are all in the bibliography of non-Euclidean geometry by D.M.Y. Sommerville, this collection is much broader than that suggests. The collection of books includes the collected works of many of the most influential mathematicians of the 19th century: A. Cayley, P. G. L. Dirichlet, E. Galois, H. Grassmann, C. G. J. Jacobi, J. L. Lagrange, L. Kronecker, B. Riemann, J. J. Sylvester, and K. Weierstrass, among others. There are also books of Euler, Gauss, Hadamard, Hermite, Hilbert, Kelvin, Klein, Legendre, Leibniz, Lie, Plücker, Poincaré, and Stokes. In addition the collection contains a large number of turn of the century European theses and other mathematical publications that had small print runs and that are currently extremely difficult to find. The ready electronic availability of this collection will have a significant impact on the work of both mathematicians and historians of mathematics. Prior to the proposed funding, work will continue to refine the list by soliciting feedback from scholars primarily at Michigan and Cornell, and ultimately the list will be made publicly available for broader review.

Cornell University

Context

The Cornell Mathematics Library dates from the founding of the university and the collection was enhanced with the purchase of the Kelly Collection in 1871 through Ezra Cornell's personal intervention. Through the remainder of the 19th and early 20th century the collection grew by the addition of several other collections, and then was complemented by Mathematics Department and University Library purchases, producing a very comprehensive collection. In 1953 the Math Library became a fully integrated part the University Library. Close personal involvement of the faculty remains very important. The Mathematics Faculty Library Endowment, started in 1990, has by direct gifts and solicitation raised $260,000 to date for the purchase of library materials. The University Library has continued a high level of support for the Mathematics Library with regular increases of its appropriation and by assigning the Class of 1938 Endowment to the Math Library in 1994. The new facility, significant endowment, renowned comprehensive collection, and the high level of use are a source of pride for the University Library. The Math Library currently holds 53,000 volumes that represent the core of the university's outstanding mathematics collection. Titles related to specific application reside with appropriate subject collections elsewhere on campus and some rare materials are in the Kroch Library.

Digital Collection

In 1991, 576 mathematics monographs were digitized as part of a preservation project. The titles were carefully selected by library staff and reviewed by a faculty advisory committee with an eye to their mathematical significance. Since this was a preservation project many very worthwhile candidate titles were not digitized because microfilm or reprint editions secured their preservation status. Even though this collection was constrained in its selection of titles it has proven to be a very popular. From the beginning there has been a steady demand from libraries and individuals for printed copies of these books. With no active marketing effort more than 1,000 volumes have been sold to over 200 customers. For project details and ordering information, see: http://www.math.cornell.edu/%7Elibrary/reformat.html. A free, online book browsing interface to Cornell University Library's digitized Historical Math Book Collection (http://cdl.library.cornell.edu/math.html) has generated a high level of use and has helped win recognition for the content of the collection.

Digital Library Efforts of the Partners

Our three institutions are also internationally recognized for outstanding accomplishments in the field of digital library research and production.

State and University Library Göttingen

Göttingen State and University Library has initiated a wide range of digital library projects. Because it holds many DFG-funded special collections in a variety of disciplines, resource discovery in a global environment is one of the key issues for the daily work in Göttingen. Göttingen has mounted significant efforts in making scholarly information available through Internet-based subject gateways (including, for example, MathGuide, GeoGuide, and Anglo-American Culture and History).

Göttingen has undertaken significant projects in the field of mathematics both on the national level with Math-Bib-Net (Corporate Information Services of Libraries and Mathematical Departments), and on the international level with EULER (EUropean Libraries and Electronic Resources in Mathematical Sciences). These efforts are especially important because Göttingen Library has the primary collecting responsibility for mathematics for Germany. Offering a "one-stop shopping site," EULER will make it possible for users to search for topics in different databases (e.g., bibliographic databases, library online public access catalogues, and indexes of mathematical Internet resources) in a single pass. A Dublin Core based metadata description plays a central role in achieving interoperability in all of these projects.

Göttingen has digitized a number of collections, with significant activities in the fields of historical travel literature and North Americana, as well as in mathematics. The Jahrbuch-project, building up an Electronic Research Archive for Mathematics (ERAM), is a joint effort of Göttingen and the Department on Mathematics at Berlin University (Prof. Wegener). In DIEPER (DIgitised European PERiodicals) project, Göttingen Library is leading a consortium of eight European Libraries, testing decentralized scanning-production and unified access over local repositories. Another goal of DIEPER is the establishment of a European database for digitized documents, which, like the EROMM (European Register of Microform Masters), will serve as a central reference to avoid duplicate digital conversion; the database will be located at Göttingen.

Göttingen Library moved from digital object description to digitizing the documents themselves in 1996, when it became the base for a new funding activity for the Deutsche Forschungsgemeinschaft. Within the program frame of establishing a distributed digital research library in Germany, a new program was initiated to support retrospective digitization of library holdings. The coordination for the initial phase was undertaken in Göttingen, and the results of the task forces on "Digital technique" and "Content selection" were published under the direction of the project officer in Göttingen.

In May 1997 the Göttingen Digitization Center (GDZ) was established. Göttingen is one of two national supply centers for digitization in Germany, with the second center located at the Bavarian State Library in Munich. The focus of the activities at the GDZ has been on the different fields of technology required to build a digital library.

Following the recommendations of the DFG technical task force, the GDZ chose a strategy of collaboration with an industrial software partner to create a Document Management System (DMS) as a key component for the digital library. The main requirements of the task force were to use open, standard formats to ensure a high degree of scalability and interoperability of the prospective digital collections: to this end, the GDZ worked with a database driven system for data import and export, and for handling highly structured documents and metadata. A prototype of Agora, the new DMS, was presented in Göttingen in April 1999 and is now in production at the GDZ. There are currently five Agora installations in Germany and there is an increasing interest outside Germany in the system.

The Agora system is a RDB (Relational database) driven EDMS based on an extensible metadata model. The model can be implemented on different RDB platforms (e.g., Oracle, DB2, and Sybase). An administrative tool (AdminTool), running on Windows NT, controls all functions. In order to allow for maximum interoperability with other metadata sources, the system works with an import/export format that is based on RDF and XML. The Java servlet of Agora acts as interface between the RDB, web server and browser. The communication with the RDB is made through JDBC. HTML templates for the user interface are used to flexibly achieve different views; they are associated with collections through the AdminTool. Based on the structured information in the underlying RDB, elaborate search functionality can be offered to the user. Advanced searches in metadata for different document types (e.g., monographs, multi-volume works, and journals) can be combined with searching in document structures such as chapters, articles, and figures. It is also possible to browse single or multiple collections.

Agora developers recently integrated the Verity Information Server, a powerful full-text search engine, used in a number of significant digital library efforts. The administrator can now offer to users the search capabilities of both the RDB and Verity, making possible not only traditional SQL-queries, but also the range of search functionality that is part of Verity (e.g., fuzzy search and ranking). The inclusion of Verity now makes it possible to offer effective searching, in metadata such as bibliographic fields, titles of chapters, articles, and figures. Later, Göttingen will offer full-text searching as well, a feature that becomes increasingly important as Göttingen moves from digitization of older text material (often in Fraktur type) to 20th century works. Verity is able to search a wide range of document formats from MSWord over PDF to XML files.

Agora's flexible export functions contribute significantly to interoperability. From its inception, Agora was able to export all data in the RDF/XML format. A recent addition, especially promising for users, was the ability to export PDF files with integrated Bookmarks, created automatically from the structural metadata in the database. Because of the modularity of Agora, the GDZ has been able to add external features to Agora as demonstrated by the successful integration of the on-the-fly conversion of images with the Tif2Gif program, developed at the University of Michigan [TIF2GIF, 1997]. In a related (import) effort, the GDZ has recently developed a tool to convert bibliographic data (Pica/GBV) into Agora's RDF/XML format, and the tool is freely available as Java-Applet from the GDZ's web site.

University of Michigan

The University of Michigan Library has been host to or a major participant in a number of significant digital library efforts, including JSTOR [Guthrie, 1997], PEAK [Bonn1, 1999], and a broad campus-wide digital library initiative begun in 1993 [Lougee, 1998]. In 1996, the Digital Library Production Service was formed within the University Library to provide a permanent support framework for the University's production-level digital library services and collections. Through its Digital Library eXtension Service (DLXS), the University of Michigan's Digital Library Production Service offers a suite of resources designed to aid educational and non-profit institutions in mounting a broad variety of digital library collections [Price-Wilkin, 1999]. The UM DLXS includes a powerful search engine, middleware, and a set of tools for mounting many types of digital library resources. The search engine, XPAT, is specially designed to handle large and highly structured documents found in digital library efforts. The tools that DLXS makes available are designed to tap the power of the XPAT engine for broad "classes" of resources found in most digital libraries. The University of Michigan invests significant production-level financial and staff resources in the ongoing development, maintenance, and support activities directly associated with the DLXS. There are currently nearly two dozen DLXS institutions, primarily in North America but also in Europe and Africa. Most are actively involved in the creation of digital library collections with significant historical value.

A notable feature of DLXS is its support for rich document formats favored by libraries and archives involved in digital library development. This focus on "durable document" formats includes mechanisms to support materials complying with a number of national and international standards, including the ITU TIFF G4 format for bitonal page image, and XML and SGML encoding (e.g., both TEI and EAD). In addition to this support of standards-compliant formats, the DLXS systems support powerful mechanisms such as wavelet compression, enabling support for higher resolutions of continuous tone images. The XPAT engine is an SGML/XML-aware search engine that the University of Michigan has deployed with an extremely diverse set of digital library resources. XPAT provides excellent support for word and phrase searching, indexing of encoded text (i.e., SGML and XML) elements and attributes, fast retrieval, and open systems integration. As part of the DLXS, the University of Michigan Digital Library Production Service has launched a continuous development process in which we hope to add a number of features to XPAT, including better support for XML and support for Unicode. These approaches, focusing on standards and rich document formats, have allowed DLXS institutions to capitalize on the growing array of services developing around the creation of these formats while ensuring a better longterm investment in conversion and creation of digital documents. [Bonn2, 1999; Price-Wilkin, 1997]

In addition to DLXS, the University of Michigan's Digital Library Production Service is responsible for a number of other production-oriented digital library efforts. It includes approximately twenty full-time staff working in areas such as digitization, information retrieval, and architecture. The digitization group within DLPS provides high volume OCR services (approximately 2 million pages per year), text encoding, continuous tone imaging (e.g., approximately 10,000 photographic quality images per year), and bitonal scanning for book and journal collections. The information retrieval group is responsible not only for mounting production systems for the University of Michigan, but also for developing a number of "host" services for non-profits and academic enterprises. In this role, DLPS develops and supports online

subscription resources for such organizations the Association of Asian Studies and the University of Michigan Press. DLPS has been responsible for the development and deployment of significant systems for authentication, usage analysis, and fee-based transactions.

Cornell University

Cornell University Library has been a leader in digital library efforts for over a decade now. It has created and maintained more than a dozen major digital library collections across a wide range of formats and disciplines from the SGML/XML Encoded Archival Description, through the Cornell University Geospatial Information Repository, to the Core Historical Collection of Agriculture. In the process of developing and maintaining these collections and services Cornell University Library has contributed to the creation of best practices in areas such as text and image conversion. Awards received include the Scout Award for the digital math books collection and the USDA Secretary's 1999 Honor Award for the USDA-Cornell Economics and Statistics System. Current major initiatives include the Mellon-funded Project Euclid (ProjectEuclid.org), a scholarly communication initiative to enable independent mathematics and statistics journals to publish their issues effectively and efficiently on the web as part of an aggregation.

Other Cornell units, namely Computer Science's Digital Library Research Group and Cornell Information Technologies have also contributed substantially to the research and practice of digital libraries through such initiatives and systems as Dienst and CUPID. The library has a strong history of partnering with these units to leverage their expertise and to balance the different perspectives that these different units bring to the table.

The Library's latest cooperative effort with Computer Science is Project PRISM, a four-year, $2.2 million, DLI2 funded effort to investigate and develop the policies and mechanisms needed for information integrity in digital libraries. The project will focus on five key areas: preservation, reliability, interoperability, security and metadata. Project PRISM will undertake research on the policies and mechanisms to ensure information integrity in the context of a component-based digital library architecture. Such an architecture allows the seamless federation of distributed content and services facilitating extensibility through the addition of new technologies and services. Supporting integrity in a digital library implemented as a distributed system poses new technical challenges. Known preservation, reliability, and security solutions were not intended for and are not sufficient for handling the novel characteristics of such digital libraries. Project Prism is a collaboration of uniquely skilled librarians, computer scientists, evaluation experts, and international testbed participants.

The Dienst protocol, at the heart of the currently proposed interoperability project, was first developed at the Cornell Digital Library Research Group. Dienst is a system for configuring a set of individual services running on distributed servers to cooperate in providing the services of a digital library. The open architecture of the Dienst system—exposure of the functionality through a defined protocol—makes it possible to combine Dienst services in flexible ways and augment the existing services with other mediator services, which build on the functionality of the existing services. The Dienst system was first implemented in the Computer Science Technical Reports Project, a DARPA-funded collaboration to establish a digital library of computer science technical reports (NCSTRL). Within the Cornell University Library, Dienst has more recently been used as the basis of Project Euclid, a publishing initiative in mathematics and statistics. For this work, Dienst has been modified and extended, and some of this development work will inform the current project.

CUPID, another building block of the distributed library of mathematical monographs that we want to build, was developed at Cornell Information Technologies. It is an architecture and protocol for high-end, distributed network printing. An implementation of the system built at Cornell allows users to stipulate custom finishing operations such as double-sided printing, stapling, covers or binding. The architecture has the capacity to facilitate billing, assess a usage or copyright fee, and access online documents from web sites, ftp servers and document archive systems (e.g. Dienst) without requiring that users download the document to their workstation. Globally distributed printing is being commercially developed by Netpaper.com. Cornell University Library is presently in negotiations for the design of a digital library printing capacity to be implemented world wide.

A History of Collaboration

The partners have a long history of national and international collaboration. The University Michigan Library and Cornell University Library cooperated on the creation of the Making of America, a multi-year project that has produced a substantial online collection of nineteenth century material. The two institutions are currently working with the Library of Congress to provide access to the Making of America available through LC's

American Memory project. Both institutions are active members of the Digital Library Federation, a body whose steering committee was chaired last year by Cornell's University Librarian, Sarah E. Thomas. The Michigan PI, John Price-Wilkin serves on the advisory group of Cornell University Library's IMLS-funded preservation project, directed by Anne Kenney. Cornell University Library has had two major recent international projects. The first one is with the National and University Library of Iceland. This project, called SagaNet, was funded by the Andrew W. Mellon Foundation and the Icelandic Government and Research Council. The other collaboration was the creation and maintenance at Cornell University Library of a mirror site of the major European mathematical indexing and reviewing database, Zentralblatt. The three institutions also have a great deal in common, including their rich collections in mathematics, their strong digital library efforts, their interest in the problems of access (especially with regard to multi-lingual collections of a single discipline) and their involvement in national and international collaboration (including with each other).

The establishment of the Digitization Center (GDZ) at Göttingen State and University Library is closely connected to the DLPS at Michigan and to Cornell University Library. During the establishment of the Center in May 1997, Norbert Lossau, head of the GDZ, together with a Frank Klaproth, visited a number of libraries in the U.S. with a focus on digital library activities. Cornell and Michigan were of special interest for Göttingen because of their extensive experience in fundamental research in the field of digitization techniques and their success in organizing the production of the digital conversion process. Subsequently, in order to share U.S. experiences with a larger audience of German librarians, Anne R. Kenney from Cornell was invited to the first national workshop of the two German Digitization Centers (Göttingen, January 1998). Later, for the third German workshop (Göttingen, October 1999), John Price- Wilkin from Michigan and Sandra Payette from the Cornell Department of Computer Sciences's Digital Library Research Group discussed their successful efforts in Michigan and Cornell. As part of the continuing exchange with their US colleagues, Norbert Lossau again visited Michigan and Cornell in November 1999. During that visit and subsequently, the operations at Göttingen and Michigan have continued to exchange information and experience regarding procedures for sustaining a high quality of metadata capture and with regard to techniques to make digital documents available via the WWW. At Cornell, Norbert Lossau gave a lecture about "Document Management for digitized Books: RDF/XML as solution for mirroring complex metadata- and document structures," and participated in intensive discussions about features of the Göttingen Agora and the various Cornell delivery systems. Beyond these personal contacts, the GDZ remains in close communication with both Cornell and Michigan with regard to various topics of digital library research and production. In January 1998, in collaboration with Anne R. Kenney, Lossau published an article about the GDZ in RLG DigiNews [Lossau, 1998]. Lossau and Klaproth also contributed a sidebar ("TIFF Header: A Reference Stamp for Image Files" to Anne R. Kenney's and Oya Rieger's new publication, Moving Theory into Practice: Digital Imaging for Libraries and Archives. From Göttingen's perspective, the proposed formal project-cooperation with Michigan and Cornell is the result of significant ongoing dialogue and should prove of great value for digital library efforts in Germany and the U.S.

Objectives and Significance

This project will be a significant step toward a unified view of the growing number of digital collections hosted by research libraries. Within the Digital Library Federation, attention has been paid to the problem of developing an architecture capable of supporting distributed collections among the member institutions. By focusing our efforts on this large and important body of historical mathematical materials in our three institutions, and by continuing to insist on a high level of functionality and support for standards, we hope to demonstrate a viable path forward for our peer institutions.

Objectives for the grant period

The major objectives proposed by the partners are:

1. We will integrate the basic Dienst protocol in each of the systems maintained at the three institutions, while ensuring that we continue to maintain each of those systems as highly functional and heterogeneous. In doing this, we will create a mechanism whereby each of those unique and highly functional systems can retrieve information from the others. As a consequence, users will be able to access all three of the collections simultaneously from any of the systems mounted at the three institutions, thus simplifying both discovery and use.
2. By using the Dienst protocol with the three separate systems, we will implement a distributed repository system for richly encoded, standards-based historical literature. This will allow us to grow and maintain the large body of materials through loose coordination.
3. The project participants will integrate OCR in the document repositories, and will use this as a basis for evaluating the value and problems of full-text searching across multi-lingual text

collections at each of the three institutions.
4. Based on the results of the evaluation of full-text searching across multi-lingual text collections at the three institutions, we will develop, implement, and evaluate mechanisms for cross-collection searching in the Dienst protocol.
5. Should cross-collection full-text searching prove impractical, we will explore the costs and value of adding searchable table of contents information to the collections.
6. In addition to providing local printing through the availability of PDF, the project partners wil attempt to integrate support for distributed printing (perhaps through CUPID) in each of the three systems.
7. The project partners will work to ensure that the digitized monographs are linked to available online reviews in Mathematical Reviews, Zentralblatt and the Jahrbuch
8. The project will create a coordination point for historical mathematical information.

Significance of the Developments

Accomplishing the above objectives promises significant benefits both for research and practice. It will add to our knowledge and understanding of the internationally important areas of research into of interoperable access and delivery systems. Lessons learned and solutions can be generalized to other disciplines and materials. Free access to this information will allow a diverse group of users to consult a large and historically significant collection of materials in mathematics. The resulting system will have major benefits to the mathematics community, as well as to the general public at large. By providing the collection without access restrictions, the effort will benefit not only large European and US research libraries, but also persons at institutions without these historically rich collections. Notably, small colleges such as many of the historically black institutions in the U.S., and other institutions world-wide will all benefit from access to this sizable and thematically focused collections. We also hope that ready access to a large, unified body of digitized monographs will help eliminate duplication in future digitization efforts at other libraries; that is, the raised profile of items in this combined collection should help make institutions aware that the items have already been converted.

The work on interoperability is a key element of the proposed activity. Digital library efforts at institutions such as Michigan, Cornell, and Göttingen have not adopted protocols such as Dienst, in the past, because of the inability of those protocols to support rich document encoding such as XML or standards-based storage and delivery formats. We have also insisted on high levels of functionality such as full-text searching across repositories, and these limitations in the protocol have been cited as well. At our institutions, parallel work has progressed in which highly functional systems have been developed around requirements to support complex, standards-based objects and high degrees of functionality (e.g., full-text searching). The Dienst protocol has matured significantly and the current version offers support for many of the requirements of the institutions hosting these rich collections [Dienst, 1999]. The incorporation of the Dienst protocol in the systems at Michigan and Göttingen, and the further development of the Euclid system at Cornell to support greater functionality will be extremely influential in communities such as the Digital Library Federation. Moreover, the ability of the systems at these institutions to be interoperable with initiatives such as the Open Archives Initiative will bridge an important gap in current and historical literature.

Plan of Work

***Phase 1*:** Staff at Michigan and Göttingen will perform a basic mapping of Dienst protocol functions to DLXS TextClass functions and Agora system functions. Differences between those in the two systems and the functions in the protocol will be analyzed, and functions not found in the two systems will be identified. Basic functions found in the systems at Michigan and Göttingen but not found in Dienst will be identified and discussed for possible expansion of the Dienst protocol. Simultaneously, Cornell will begin generating OCR for its math collection, and Michigan will begin converting materials in its collection.

***Phase 2*:** Michigan and Göttingen will incorporate the basic functions of the Dienst protocol in the DLXS and Agora systems, and Cornell will implement the current Math collection using a separate instance of the Euclid system. Dienst protocol mechanisms that support searching across bibliographic data, browsing metadata, and page/document retrieval will be incorporated in the DLXS and Agora systems. While this will be accomplished by developing a Dienst layer for each of the two systems, we expect that both DLXS's TextClass system and Agora will need to be modified to take the protocol layer into account.

***Phase 3*:** Project participants will begin testing basic interoperability between the three systems, using those titles that have been converted at Michigan, as well as the full collection at Cornell and any available titles in Göttingen. Basic deficiencies in cross-collection access will be identified. It is expected that, at the outset,

each of the three sites will use mirroring as needed to address problems of network latency.

***Phase 4*:** The work of providing specifications for full-text access will begin in parallel to work that seeks to address deficiencies identified in Phase 3. Specifications for full-text access should address issues of query formulation (e.g., should we use a subset of Z39.50 to encode query operators and search terms?) and the form in which results are returned. The problem of returning meaningful results, especially in collections of book-length materials, is especially important to address.

***Phase 5*:** The systems at the three institutions will be released to the public for basic browsing and bibliographic searching. Implementation of full-text functionality in the developing protocol will begin. The Michigan and Göttingen systems will work to incorporate support for CUPID in order to provide distributed printing.

Phase 6: Full-text searching across the three repositories will be released. Evaluation and documentation of the distributed system will begin, and results will be shared with the appropriate communities, though especially with the Digital Library Federation in the U.S. and the VDF (Verteilte Digitale Forschungsbibliothek) libraries in Germany

State and University Library Göttingen

The Agora system of the GDZ has its strengths in a rich metadata mechanism, supported by a robust system architecture based on the relational database and a powerful full-text search engine (Verity Information Server). Specialized search capabilities are offered for a variety of types of metadata, and both object and metadata are managed and accessed in a distributed environment, connected via http protocol. To date, resources on the Agora document server are available both directly via the Web-Sites of the GDZ and via the PICA/GBV Online Union Library Catalog. Integrated access is provided at the bibliographic level for digital documents through traditional library catalog, a key issue for German DFGfunded digitization projects. Because it relies on RDF/XML as both an import and export format, the Agora system is designed for data interoperability.

Project work at Göttingen will focus on developing a software module to make the current version of the Dienst protocol available to Agora resources. The result of the development will be a gateway providing all Dienst protocol functions using the strength of RDF/XML as metadata format and TEI/XML as encoding for full text. One focus of the work at Göttingen will be the representation and mapping of different character sets found in a multilingual distributed repository. Serving requests from the distributed Dienst protocol will be an important addition to the sophisticated retrieval functionality of the highly structured digital objects supported by the Agora system. Enabling different ways of access to a single digital object repository, as well as a unified means of access to different decentralized document repositories, will be a very important accomplishment for internationally distributed libraries.

University of Michigan

Although the University of Michigan DLXS middleware offers significant functionality in repository retrieval and navigation, even in a distributed environment, it is not interoperable with other non-DLXS digital library systems. To date, DLXS components have been deployed in multi-machine environments within a single institution, with indexes distributed among several different functionally specialized servers [Weise, 2000]. DLXS resources have also been deployed in a multi-institutional environment, for example with cross-machine full-text searching of EAD-encoded finding aids demonstrated at five institutions, including Oxford University [DFAS, 1999]. Although the DLXS methods for retrieval are highly generalized (e.g., allowing many external projects to by-pass the interface and point to individual pages, sections, or works within collections), and although development has been heavily influenced by object-oriented design, the browse and search mechanisms are not interoperable with any existing higher level protocol (e.g., Z39.50).

Project work at Michigan will focus on developing a software module to make the current version of the Dienst protocol available to DLXS resources. For example, DLXS middleware will continue to use local database applications to manage resolution of object identifiers to their location(s), but will accept and interpret requests for those same objects through the Dienst protocol. Similarly, although digital objects (and parts of digital objects) will be delivered via the protocol, they will continue to capitalize on the digital object management strategies (e.g., the strong reliance on standards-based formats) underlying DLXS systems. The addition of this layer will ensure that a baseline of interoperability will be in place for large digital library projects without compromising the high level of functionality provided by the DLXS.

Cornell University

Cornell University Library has significant history and experience in implementing Dienst, an interoperable digital library architecture and protocol developed by the Cornell Digital Library Research Group in Cornell's Computer Science Department. In 1996, the library developed and implemented a document browse system based on Dienst 3.5. This layered system architecture included a user interface in Perl communicating with low-level repository information via the Dienst protocol. In 1998, library personnel experimented with Dienst 4.0, providing feedback to Dienst developers. Later, the library implemented a revised Dienst-based user interface, called Hunter. Built on Dienst 5.1 and developed by Cornell Information Technologies, Hunter offered increased user functionality by mediating access to a collection of Dienst services, such as repository information now expressed in XML, an indexing and searching service, and hand-off to full-text printing services. More recently, Project Euclid has modified and extended Dienst so that it supports an even more complete range of digital library services and functions.

The library is thus committed to the development of an open digital library architecture, with system functionality exposed through a defined protocol such as Dienst. Project work at Cornell will concentrate on building even greater functionality into the system developed for Project Euclid. A redesigned interface service is anticipated, extending functionality to allow for such services as true cross-collection searching of both bibliographic metadata and full-text OCR output, improved search results delivery and navigation, and enhanced navigation of internal document structure.

Plans for Documentation and Sharing of Content

The interoperability modules developed for each of the separate systems will be documented as part of their maintenance, support, and distribution. At the University of Michigan and at Göttingen, the modules will be supplied with formally supported digital library systems (i.e., DLXS and Agora), and thus will require formal documentation for customers. We will work with customers to determine the adequacy of the documentation, though both the module and the documentation will be made freely available.

The successful digital library of mathematical monographs will be maintained as a production system after the funded project phase. Each participating library considers its collection of digitized math books to be part of its core mission and consequently each institution is committed to maintaining its collection through its library budget. The proposed activity at each institution will be supported by permanently funded, production-level digital library operations, thus enabling us to allocate resources to this end. The three partnering institutions will continue to communicate with each other with the goal of coordinating further efforts.

All digitized material will also be linked from or to different databases, according to their publication year:

> *Up to 1943:* Jahrbuch über die Fortschritte der Mathematik will include links to all online project items
> *1933-present:* Zentralblatt für Mathematik; reviews will be linked from project resources, and we will request that editors add links to project resources in the database
> *1943-present:* Mathematical reviews; reviews will be linked from project resources, and we will request that editors add links to project resources in the database.

This will ensure another important avenue for “discovery” for mathematicians. In addition, at Göttingen State and University Library, materials will be available through two European internet-based projects, EULER and REYNARDUS and by means of MATHNET.

Evaluation, Dissemination and Maintenance of the Results

The work of the project will be evaluated to determine the extent to which the digital library systems at the respective institutions can interoperate using the Dienst protocol, and the extent to which that protocol can accommodate the high degree of functionality in those existing systems. Criteria include successful searching of bibliographic data (including appropriate mechanisms to balance precision and recall), reliable browsing of bibliographic data (including multi-lingual sorting), and successful browsing of books containing page images. This must be accomplished through each interface at the three participating institutions, operating with the three separate systems. A further criterion, especially with regard to extending the capabilities of the current Dienst protocol, includes successful full-text searching across the three repositories, especially with regard to providing users with information on relevance and location of results within monographic volumes (i.e., to aid in user navigation). Support for both local and distributing printing

will also be evaluated.

The partners will share the experience and lessons learned from the project with the profession through papers and conference presentations, and especially with Digital Library Federation partners (e.g., in the Architecture Committee). Further, access to the content of the collection will remain available free of charge to the international public. The modules written to accomplish interoperability will be demonstrated to participants of the Open Archives Initiative and will be made available to any requesting institution; further, we anticipate that eventually the entire Euclid system, developed at Cornell, will be made freely available. Each of the institutions will begin work to incorporate the interoperability layer in other collections.

Management Plan

Management of this project has the added challenge of coordinating the activities of developers at three different international institutions. Communication must be encouraged with both the informal use of email, the use of regularly scheduled telephone conferences, and occasional site visits by all of the major participants. A development infrastructure must be set up to allow easily sharing code, data and documentation. Project Management software tools for timeline management and documentation must be agreed upon and used at all three institutions. There must be a coordinated review of milestones for each individual institution at least once a month.

This project will be split into three overlapping phases:

1. Research & Design: Evaluation of the functionality of the Dienst protocol, assessment of the features of Michigan's DLXS middleware, Cornell's Euclid system, and Göttingen's
Agora middleware, identification & design of the needed enhancements to the Dienst protocol, design of the addition of the Dienst protocol layer each middleware.

Milestones

Development Infrastructure in place: 1 month from inception of project
Evaluation of each institution's requirements for Dienst protocol: 2 months from inception of project
Partitioning of development responsibilities: 2 months from inception of project
Initial High Level Design complete: 3 months from inception of project
Design of Full-Text Searching complete: 12 months from inception of project

2. Development: Software development at three institutions including specified milestones for interim testing of interoperability.

Milestones

Detailed Development Timeline: 4 months from inception of project
Preliminary interoperability testing: 9 months from inception of project
Initial development complete: 12 months from inception of project
Additional functionality (Full-Text Searching complete: 18 months from inception of project

3. Integration Release: Release of fully interoperable system involving three institutions, testing, necessary enhancements, overall system evaluation, and final documentation.

Milestones

Initial public release from three institutions: 13 months from inception of project
Preliminary Evaluation of System: 18 months from inception of project
Public release of Full-Text Searching): 19 months from inception of project
Final Evaluation & Documentation of System: 24 months from inception of project

The Research, Design, and Development work will be partitioned between the three institutions to take advantage of the existing middleware development at each of the digital libraries. Michigan, Cornell and Göttingen will focus on integrating the Dienst software layer to each of their middleware implementations. Because the Dienst protocol is as a project of the CDLRG (Cornell Digital Library Research Group) and not a part of Cornell University Library, we will work to ensure that the addition of functionality to the Dienst software is communicated to CDLRG, and their feedback taken into account. Finally, the benefits of bringing

the development team from all three institutions together at one location are invaluable, and it is anticipated that the full development team will meet once at each institution to coincide with each of the three project phases.

Site visit one (three months from inception of project): project manager and project programmer from Göttingen and the University of Michigan will travel to Ithaca, NY.
Site visit two (twelve months from inception of project): project manager and project programmer from Cornell University and the University of Michigan will travel to Göttingen.
Site visit three (eighteen months from inception of project): project manager and project programmer from Cornell University and the Göttingen will travel to Ann Arbor, MI.

REFERENCES CITED

[Bonn1, 1999] Bonn, Maria; Wendy P. Lougee, Jeffrey K. Mackie Mason and Juan F. Riveros "A Report on the Peak Experiment: Context and Design." D-Lib Magazine, June 1999, online at http://www.dlib.org/dlib/june99/06bonn.html

[Bonn2, 1999] Bonn, Maria. "University of Michigan Polices and Practice for the Long Term Retention of Locally Produced Digital Projects and Materials: A Report Prepared for the Joint RLG/TASK Force on Digital Preservation" online at http://www.umdl.umich.edu/um-rlg.html

[DFAS, 1999] "Supporting Access to Diverse and Distributed Finding Aids: A Final Report to the Digital Library Federation on the Distributed Finding Aid Server Project," available online at http://www.umdl.umich.edu/dfas/dfas-final.html

[Dienst, 1999] "Dienst, Overview and Introduction," available online at http://www.cs.cornell.edu/cdlrg/dienst/DienstOverview.htm

[Lossau, 1998] Lossau, Norbert; Klaproth, Frank "Digitization Efforts at the Center for Retrospective Digitization, Göttingen University Library." RLG DigiNews, 3:1 (1998), 7-10.

[Lougee, 1998] Lougee, Wendy P. "The University of Michigan Digital Library Program: A Retrospective on Collaboration within the Academy." Library Hi-Tech, 16:1 (1998), 51-59.

[Price-Wilkin, 1997] Price-Wilkin, John. "Just-in-time Conversion, Just-in-case Collections: Effectively leveraging rich document formats for the WWW." D-Lib Magazine, May 1997, online at http://www.dlib.org/dlib/may97/michigan/05pricewilkin.html

[Price-Wilkin, 1999] Price-Wilkin, John. "Moving the Digital Library from 'Project' to 'Production.'" DLW99, Tsukuba, Japan. March 1999. [TIF2GIF, 1997] TIF2GIF web site at http://kalex.engin.umich.edu/tif2gif/

[Weise, 2000] Weise, John, Alan Pagliere, and Matthew Stoeffler. "Integrating Heterogeneous Databases in a Distributed Environment: The 'Pictures of Record' System." Proposed for publication in mid-2000.

[Guthrie, 1997] Guthrie, Kevin, and Wendy Lougee. "The JSTOR Solution: Accessing and Preserving the Past." Library Journal 122:2 (1997), 42-44.

A DISTRIBUTED DIGITAL LIBRARY OF MATHEMATICAL MONOGRAPHS

A collaborative project of the University of Michigan Library, Cornell University Library, and the State and University Library Göttingen

PROJECT TEAM

John Price Wilkin
Head, Digital Library Projection Services, University of Michigan Library
Principal Investigator and Project Coordinator

H. Thomas Hickerson
Associate University Librarian for Digital Library and Information Technologies, and Special Collections, Cornell University Library
Principal Investigator

The University of Michigan Library

John Price Wilkin
Head, Digital Library Projection Services
Michigan Project Manager

Jose Blanco
Digital Library Projection Services
Programmer/Analysis

Cornell University Library

David Ruddy
Digital Library and Information Technologies
Cornell Project Manager

David Fielding
Digital Library and Information Technologies
Programmer/Analyst

Göttingen State and University Library

Markus Enders
Göttingen Digitization Center (GDZ)
Göttingen Project Manager

A DISTRIBUTED DIGITAL LIBRARY OF MATHEMATICAL MONOGRAPHS

A collaborative project of the University of Michigan Library, Cornell University Library, and the State and University Library Göttingen

PROJECT SITES

Multi-Collection Access Points

from Cornell

from Michigan

Local Collections

University of Michigan Historical Math Collection

Cornell Historical Math Monographs Collection

SUB Göttingen Mathematica Collection Browse Search (all GDZ collections)

A DISTRIBUTED DIGITAL LIBRARY OF MATHEMATICAL MONOGRAPHS

A collaborative project of the University of Michigan Library, Cornell University Library, and the State and University Library Göttingen

WORKING DOCUMENTS

Current CGM protocol

Implementation profile

History documents:

Protocol-First Draft (Summer 2001)

Cornell's Draft SearchBoolean verb (April 2002)

Search verb notes from May 23-24, 2002 meeting

Search verb, ver. 1.0 alpha (Nov. 2002)

Search verb, ver. 1.0 beta (Dec. 2002)

CGM Verbs (2003-03)

Proposed Search verb ver. 2.0, overview (March 2003)

A DISTRIBUTED DIGITAL LIBRARY OF MATHEMATICAL MONOGRAPHS

A collaborative project of the University of Michigan Library, Cornell University Library, and the State and University Library Göttingen

RELATED RESOURCES

Digital Library Architectures, Protocols

Open Archives Initiative

Dienst [archive site]

Scholnet: Digital Library Testbed (Home)

Scholnet: Digital Library Testbed (Protocol, PDF)

ZING -- Z39.50 International: Next Generation

Common Query Language

XML Query

Cross Repository Full-Text Systems

DLXS

CGM Protocol

Describe Verb

Verb name: DescribeVerb [this still needs work yet]
Verb version: 1.0 alpha
Modification Date: 2003-01-01
Required arguments: value
Optional arguments: none
MIMETYPE of response: text/xml

Verb Description

Returns a structured response that contains a list, where each element of the list provides information on a version of the specified verb that is supported by this service.

Arguments

- **`value`** :: [required] name of the verb to be described.

Response

The following information may be provided at the verb or version level.

- **`description`** :: , description of the verb or a specific version
- **`note`** :: , information pertaining to the verb or a specific version

Each element of the list contains the following information:

- **`version number`** :: of the verb.
- **`arguments`** :: , a list of the names of the fixed and keyword arguments, if any, accepted by the verb in that version.
- **`example`** :: template of request to this repository, with fixed argument indicated in brackets
- **`returns`** :: , optional, contains information about response format.

Note that a service may implement more than one version of a verb.

Error Codes

-

Examples

Request

```
protocol=CGM&verb=DescribeVerb&ver=1.0&value=Formats
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
  <DescribeVerb ver="1.0">
    <verb name="Formats">
      <description>Returns a structured response indicating
        the formats of disseminations available for this document
      </description>
      <versions>
        <version id="1.0">
              <example>http://www.umdl.umich.edu/cgi/b/broker/broker?
                                protocol=CGM&ver=2.0&

verb=Formats&identifier=identifiervalue></example>
          <arguments>
            <required>
              <arg name="identifier" />
            </required>
            <optional>
              <arg name="version" />
              <arg name="view" />
            </optional>
          </arguments>
        </version>
      </versions>
    </Verb>
  </Describe-Verb>
```

Verb History

Modification of Dienst verb Describe-Verb, version 2.0.

Display a Document in Local Viewer

Verb name: Display
Verb version: 1.0
Modification Date: 2003-06-18
Required arguments: identifier
Optional arguments: divID
MIMETYPE of response: will depend on request and viewer mechanism

Verb Description

Request to view a document in a repository's local viewer.

Arguments

- **`identifier`** :: [required] the identifier for the document requested.
- **`divID`** :: one or more identifiers, separated by "|", indicating document subdivisions with query matches. No assumptions are made about what a repository will do with this information. Bad `divID` values are ignored.

Response

The response to a Display verb request is an HTTP redirect, if the request is successful. A repository can display a document however it chooses. If the request is not successful, an error response is returned.

Error Codes

- **`idDoesNotExist`** :: the value of the `identifier` argument is unknown or illegal.
- **`cannotDisplay`** :: the repository is unable to display the document.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Display&ver=1.0&identifier=cul.math/00640001

http://some.cgm.server/script?protocol=CGM
        &verb=Display&ver=1.0&identifier=cul.math/98765432
        &divID=cul.math/98765432/00000024|cul.math/98765432/00000128|
        cul.math/98765432/00000156|cul.math/98765432/00000237
```

Verb History

New to CGM.

Disseminate Content

Verb name: Disseminate
Verb version: 1.0
Modification Date: 2003-06-30
Required arguments: identifier, format-type
Optional arguments: version, div
MIMETYPE of response: Dependent on dissemination requested.

Verb Description

Request a dissemination of a digital object. The characteristics of the dissemination that can be requested (the arguments to the `Disseminate` verb), are determined by the responses to the `Structure`, `Formats`, `ListViews`, and `ListVersions` verbs for the specific document. The response is a MIME-typed byte stream.

Arguments

- **`identifier`** :: [required] the document identifier
- **`version`** :: specifies the document version for which the format information is requested. If omitted, it provides information about the latest document version available (see ListVersions).
- **`div`** :: specifies the id value of a div or formatDiv component provided in the Formats response. Use to request dissemination of a specific component of the specified document. If no `div` argument is submitted, the highest level `div` of the document is disseminated, if possible.
- **`format-type`** :: [required] specifies the type of the content in the dissemination. The list of available formats for a division is indicated by the response to the `Formats` request. Note that the value supplied for the argument is not a mime-type, but the format "type" indicated in the response to the `Formats` request.

Response

The response to Disseminate is the requested content, directly.

Error Codes

- **`idDoesNotExist`** :: the value of the `identifier` argument is unknown or illegal.
- **`badArgument`** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.
- **`cannotDisseminate`** :: no disseminatable format for the requested division.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Disseminate&ver=1.0&identifier=cul.math/00640001
        &div=chunk2&format-type=PDF.600
```

Response

The response returns a PDF file of a portion of the document identified as "chunk2".

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Disseminate&ver=1.0&identifier=cul.math/00640001
        &div=a123-2&format-type=GIF
```

Response

Returns a GIF image of a division of the document identified as "a123-2".

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Disseminate&ver=1.0&identifier=cul.math/00640001
        &format-type=PDF.600
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request protocol="CGM" verb="Disseminate" ver="1.0"
           identifier="cul.math/00640001" format-type="PDF.600">
           http://some.cgm.server/script</request>
  <error code="cannotDisseminate">No format available for dissemination
         of requested division</error>
</CGM>
```

The request included no `div` argument, and so format information on the highest level structural division of the document was assumed. But this division cannot be disseminated (has no disseminatable format), and thus the error. In the response to a Structure verb request for this document, this division would have included a `diss` attribute value of "0".

Verb History

Modification of Dienst Disseminate verb, version 1.0. Our position on the current CGM verb is that the record-oriented components (metadata) are to be replaced by OAI verb `GetRecord`, and that Disseminate needs to continue for handling the actual content.

Removed from Dienst verb description: "The MIME type of the byte stream is either the MIME type of the specified `content-type` or, if a `binder` is specified, the MIME type of that `binder`."

Removed from above, this was a definition of argument div (from jpw?): " narrows the dissemination request to a specific structural component (e.g., a ***chapter***) of the specified view of the document instance. The supplied argument must be one the available `divs` of the view (indicated by the output of the `Structure` request) paired with a value that is an identifier of one of those `divs` (again indicated by the output of the `Structure` request). For example, if a `Structure` request indicates that there is a `div` of type `chapter` with identifier `1`, then this argument could by `chapter=1`. Note that the `div` argument can not be used in combination with the `pageimage` argument."

Also removed from argument section: "<viewID>,where `<viewID>` is one of the available views for the document instance. The resulting dissemination, if there is no `pageimage` argument, is then the specified view of the document instance."

Get Formats

Verb name: Formats
Verb version: 1.0
Modification Date: 2003-06-18
Required arguments: identifier
Optional arguments: version, div
MIMETYPE of response: text/xml

Verb Description

Returns a structured response indicating the formats available for one or more structural components of a document.

Arguments

- **`identifier`** :: [required] the document identifier
- **`version`** :: specifies the document version for which the format information is requested. If omitted, it provides information about the latest document version available (see ListVersions).
- **`div`** :: specifies one or more ID values corresponding to one or more div components provided in the Structure response. Use to request format information for a specific structual component. If no `div` argument is included, the highest level div for this doc is assumed. If multiple values are included, separate with "|".

Response

Each response includes one or more `divReq` elements.

- **`<divReq>`** :: Any `div` for which format information has been requested becomes a `divReq` (requested div) element in the response. A `divReq` element has the same attributes as a Structure response `div` (except that the `order` attribute is not used):
  - `id` :: [required] an ID value for this division, usable as is in Disseminate and other requests. This is not defined as an XML ID.
  - `type` :: type of division.

    - `label` :: a displayable label for this division. These are applied by the local repository and should make sense in a navigational context.

  The `divReq` element must contain at least one `format` or `formatSeq` element. It may contain more of either.
- **`<format>`** :: an empty element with attributes describing a single format available for its parent element, which may be `divReq`, `div`, or `formatDiv`. The `format` element has four attributes:
    - `type` :: [required] from a controlled vocabulary of format types.
    - `mime` :: [required] the mime-type of the format.
    - `size` :: in bytes.
    - `label` :: [required] a displayable label identifying this format for use in a user interface.
- **`<formatSeq>`** :: an element that contains a sequence of `formatDiv` elements. The `formatSeq` element has two attributes:
    - `type` :: [required] from a controlled vocabulary of format types.
    - `label` :: [required] a displayable label identifying this group of formats, for use in a user interface.
- **`<formatDiv>`** :: a division within `formatSeq` that contains one or more `format` elements. `formatDiv` is used for divisions that are not structural--not represented in Structure verb responses--but exist because of arbitrary segmenting of documents for delivery purposes. This element has three attibutes:
    - `id` :: [required] an ID value for this division, usable as is in Disseminate requests. This is not defined as an XML ID.
    - `order` :: [required] the order of this `formatDiv` among its siblings. Values are positive integers, beginning with 1.
    - `label` :: a displayable label for this division. These are applied by the local repository and should make sense in a navigational context.

Error Codes

- **`idDoesNotExist`** :: the value of the `identifier` argument is unknown or illegal.
- **`badArgument`** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.
- **`noFormatAvailable`** :: no disseminatable formats are available for the requested document division.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Formats&ver=1.0&identifier=cul.math/00640001
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request protocol="CGM" verb="Formats" ver="1.0"
           identifier="cul.math/00640001">
           http://some.cgm.server/script</request>
  <Formats ver="1.0">
    <identifier value="cul.math/00640001"/>
    <divReq id="a123" type="maindocument" label="Entire Monograph">
      <format type="PDF.600" mime="application/pdf" size="5500000"
              label="Entire Document in PDF"/>
      <formatSeq label="PDF Files" type="pdf-chunks" >
        <formatDiv id="chunk1" order="1" label="Pages 1-30">
          <format type="PDF.600" mime="application/pdf" size="1620000"
                  label="Multi-page PDF"/>
        </formatDiv>
        <formatDiv id="chunk2" order="2" label="Pages 31-60">
          <format type="PDF.600" mime="application/pdf" size="1650000"
                  label="Multi-page PDF"/>
```

```
        </formatDiv>
        <formatDiv id="chunk3" order="3" label="Pages 61-90">
          <format type="PDF.600" mime="application/pdf" size="1650000"
                  label="Multi-page PDF"/>
        </formatDiv>
        <formatDiv id="chunk4" order="4" label="Pages 91-103">
          <format type="PDF.600" mime="application/pdf" size="550000"
                  label="Multi-page PDF"/>
        </formatDiv>
      </formatSeq>
      <formatSeq label="ASCII Text Files" type="ascii-chunks">
        <formatDiv id="chunk1" order="1" label="Pages 1-50">
          <format type="OCR" mime="text/plain" size="220000" label="Un-proofed
                  OCR text"/>
        </formatDiv>
        <formatDiv id="chunk2" order="2" label="Pages 51-103">
          <format type="OCR" mime="text/plain" size="240000" label="Un-proofed
                  OCR text"/>
        </formatDiv>
      </formatSeq>
    </divReq>
  </Formats>
</CGM>
```

No `div` argument was used, so information about the the highest level structural `div` for this document is returned in the response. Three formats are available for this structural division: a single PDF of the entire document; a sequence of PDF files that include arbitrary amounts of the document; a sequence of ASCII text files that include arbitrary amounts of the document.

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Formats&ver=1.0&identifier=cul.math/00520098
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request protocol="CGM" verb="Formats" ver="1.0"
           identifier="cul.math/00520098">
           http://some.cgm.server/script</request>
  <error code="noFormatAvailable">No format available for
         requested division</error>
</CGM>
```

No `div` argument was used, so the system looked for information about the the highest level structural `div` for this document. But there is no format for delivering the entire document, and thus the error message. In a Structure verb response, this `div` would have a `diss` attribute value of "0", indicating that there is no format for disseminating the division as a whole. It is possible that subdivisions of this `div` have disseminatable divisions (`diss="1"`).

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Formats&ver=1.0&identifier=cul.math/00640001
        &div=a123-1|a123-2|a123-3
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request protocol="CGM" verb="Formats" ver="1.0"
           identifier="cul.math/00640001">
           http://some.cgm.server/script</request>
  <Formats ver="1.0">
```

```
    <identifier value="cul.math/00640001"/>
    <divReq id="a123-1" type="page" label="Page i">
      <format type="GIF" mime="application/gif" size="63488"
              label="Page Image"/>
      <format type="PDF.600" mime="application/pdf" size="54272"
              label="Page in PDF"/>
      <format type="ascii" mime="text/plain" size="4096"
              label="Un-proofed OCR text"/>
    </divReq>
    <divReq id="a123-2" type="page" label="Page 1">
      <format type="GIF" mime="application/gif" size="64292"
              label="Page Image"/>
      <format type="PDF.600" mime="application/pdf" size="55674"
              label="Page in PDF"/>
      <format type="ascii" mime="text/plain" size="3995"
              label="Un-proofed OCR text"/>
    </divReq>
    <divReq id="a123-3" type="page" label="Page 2">
      <format type="GIF" mime="application/gif" size="65123"
              label="Page Image"/>
      <format type="PDF.600" mime="application/pdf" size="60234"
              label="Page in PDF"/>
      <format type="ascii" mime="text/plain" size="4123"
              label="Un-proofed OCR text"/>
    </divReq>
  </Formats>
</CGM>
```

Format information is requested for three structural divisions. Each division is available in three formats: GIF image, PDF file, and ASCII text file.

Verb History

Modification of Dienst verb Formats, version 4.0.

List Verbs

Verb name: ListVerbs
Verb version: 1.0
Modification Date: 2003-06-18
Required arguments: none
Optional arguments: none
MIMETYPE of response: text/xml

Verb Description

Returns a structured response containing the name and version number of each CGM verb supported by this repository. Different versions of the same verb should be included, within separate `verb` elements.

Arguments

Response

The List Verbs response contains a single, repeatable element.

- **`<verb>`** :: an EMPTY element with two attributes:
  - `name` :: the verb name.
  - `ver` :: the version of the verb.

Error Codes

- **badArgument** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=ListVerbs&ver=1.0
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="ListVerbs" ver="1.0">
           http://some.cgm.server/script</request>
  <ListVerbs ver="1.0">
    <verb name="Disseminate" ver="1.0" />
    <verb name="Display" ver="1.0" />
    <verb name="Formats" ver="1.0" />
    <verb name="Search" ver="1.0" />
    <verb name="Search" ver="2.0" />
    <verb name="Structure" ver="1.0" />
    <verb name="ListViews" ver="1.0" />
    <verb name="ListVerbs" ver="1.0" />
  </ListVerbs>
</CGM>
```

Verb History

Dienst verb, version 2.0

List Document Versions Available

Verb name: ListVersions
Verb version: 1.0
Modification Date: 2003-06-18
Required arguments: identifier
Optional arguments: none
MIMETYPE of response: text/xml

Verb Description

Returns a structured response describing the current versions available for the requested document.

Arguments

- **identifier** :: [required] the document identifier for which version information is requested.

Response

- **<version>** :: its attribute `value` is the version number, a positive integer, used when requesting a specific version in Structure, Formats, and Disseminate verbs. The element contains two subelements:
  - **<date>** :: the date (or date-time) the version was last modified, expressed in using ISO 8601.

<comment> :: a comment or description.

Error Codes

- **idDoesNotExist** :: the value of the `identifier` argument is unknown or illegal.
- **badArgument** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=ListVersions&ver=1.0&identifier=cul.math/00640001
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="ListVersions" ver="1.0" identifier="cul.math/00640001">
           http://some.cgm.server/script</request>
  <ListVersions ver="1.0">
    <identifier value="cul.math/00640001">
    <version value="2">
      <date>2002-03-18</date>
      <comment>Author revised version.</comment>
    </version>
    <version value="1">
      <date>1999-10-01</date>
      <comment>Original published version.</comment>
    </version>
  </ListVersions>
</CGM>
```

Verb History

Modification of Dienst verb List-Versions, version 1.0.

List Views Available

Verb name: ListViews
Verb version: 1.0
Modification Date: 2003-06-18
Required arguments: identifier
Optional arguments: none
MIMETYPE of response: text/xml

Verb Description

Returns a structured list of the possible views available for a single document.

Arguments

- **identifier** :: [required] an identifier of the document for which a list of views is requested.

Response

A default view is mandatory. One view, but only one, must be identified as the default view.

- **`<view>`** :: an empty element, one per view, with three attributes:
  - `id` :: [required] an id of the view (XML ID).
  - `label` :: [required] a displayable, short label describing this view. These are applied by the local repository and should help end users choose an appropriate document view.
  - `default` :: [required for default view] possible values are `1` or `0`. If not included, `default="0"` is assumed.

Error Codes

- **`idDoesNotExist`** :: the value of the `identifier` argument is unknown or illegal.
- **`badArgument`** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=ListViews&ver=1.0&identifier=cul.math/00640001
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="ListViews" ver="1.0" identifier="cul.math/00640001">
           http://some.cgm.server/script</request>
  <ListViews ver="1.0">
    <identifier value="cul.math/00640001">
    <view id="v123-a" label="Page listing" default="1" />
    <view id="v123-b" label="Chapter and Sections" />
  </ListViews>
</CGM>
```

Verb History

New to CGM. No corresponding Dienst verb.

Submit a Query

Verb name: Search
Verb version: 1.0
Modification Date: 2003-06-18
Required arguments: field, value, op
Optional arguments: set, sort, startResult, resultSize
MIMETYPE of response: text/xml

Verb Description

Specifies a search request to an index server. Arguments indicate search criteria and the order and amount of search results to be returned. The response is a structured list of documents that match the

search criteria, and includes a results summary. It is possible to request the results summary only.

Arguments

The arguments `field[n]` and `value[n]` enumerate, beginning with 1. For `op[n]`, see below.

- **`set`** :: an optional grouping construct, identical in definition to OAI's use. With CGM Search, however, multiple sets can be indicated in a single query by using a delimiter (|); for example: `set=set1|set2|set3`. When searching across multiple sets, the complete query is executed within each set (effectively OR'ing the sets). When no set is indicated, the default is either 1) all sets, when a repository supports sets, or 2) all repository content, when the repository does not support sets.
- **`sort`** :: a requested sort order for result sets. Default is none. Possible values are:
  - `none`
  - `rank`
  - `title`
  - `author`
  - `pubdate`
- **`startResult`** :: the numerical order of the first result to be returned in the response. Value is an integer, 0 or greater; default is 1. If the value is set to 0, no results will be returned, regardless of the value of `resultSize`.
- **`resultSize`** :: the number of result records requested. Value is an integer, 0 or greater; default is all results. If the value is set to 0, no results will be returned, regardless of the value of `startResult`.
- **`field[n]`** :: [required] the field, or index, within which to search. Supported values for bibliographic field arguments may vary among index servers. The `DescribeVerb` verb returns the set of field values supported by a particular server.
- **`value[n]`** :: [required] the query string
- **`op[n]`** :: [required, if multiple field/value pairs submitted] op[n] will normally operate on `n` and `n-1` field/value arguments. See the separate document on RPN Notation. Following is a list of permitted argument values.
  - `and`
  - `or`
  - `not`
  - `within`
  - `including`

Rules for field matching: the value of the `value` argument is treated as a single string (word or phrase) and matched exactly as submitted. Truncation is possible on single word queries. The wild card symbol `*` should be used to indicate word truncation.

Operator Precedence: queries are assumed to be submitted using Reverse Polish Notation (RPN). See the separate document on RPN Notation

Response

Unless there is an error, all responses include a single `resultsSummary` and zero to many `record` elements.

A well-formed query that executes properly and yet has no matches is not considered an error. In this case, a `resultsSummary` would be returned, but no `record` elements.

- **`<resultsSummary>`** :: [required] an empty element with data about the results returned. All of the following attributes must be included, using defaults if not specified in request. These values describe the results actually returned; they may differ from values in the request.
  - `repositoryIdentifier` :: the repository identifier.
  - `set` :: if no set was indicated in the response, then return either 1) a list of all sets searched, if sets are supported (`set="set1|set2|set3"`); or 2) an empty value, if sets are not supported

(`set="0"`).
    - `sort` :: use default when not in request. Note that the requested sort order can be ignored by the indexer if the requested sort value is not supported by the indexer, or if the indexer's `sortThreshold` has been exceeded. An indexer's supported sort orders and `sortThreshold` value is publicized in the Describe Verb response.
    - `totalResults` :: the total number of matches on a query, regardless of how many or which results are returned.
    - `startResult` :: the order of the first result returned in this response. Value is 0 when no results are returned.
    - `resultSize` :: the number of results returned in this response. This may be less than the `resultSize` requested, if there are fewer results than requested. Note that the requested result size can be ignored and reset to any value by the indexer if the indexer's `resultsThreshold` has been exceeded. An indexer's `resultsThreshold` value is publicized in the Describe Verb response.
- **<record>** :: zero to many `record` elements allowed. Currently allowed `record` elements are the following:
    - **<identifier>** :: [required] the unique document identifier. Its format is determined by the repository.
    - **<title>** ::
    - **<author>** ::
    - **<pubdate>** :: The date of publication of the document.
    - **<rank>** :: Relevance data.
    - **<resultDivs>** :: in the response to a full-text search, this element may contain ID values of document subdivisions containing matches. Each ID value is contained within a separate <divID> element.

Error Codes

- **badArgument** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.
- **noSetHierarchy** :: the repository does not support sets

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Search&ver=1.0&set=math&resultSize=100
        &field1=author&value1=todhunter&field2=title
        &value2=trigonometry&op2=and
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-01T19:20:30Z</responseDate>
  <request protocol="CGM" verb="Search" ver="1.0" set="math" resultSize="100"
           field1="author" value1="todhunter" field2="title"
           value2="trigonometry" op2="and">
           http://some.cgm.server/script</request>
  <Search ver="1.0">
    <resultsSummary repositoryIdentifier="CULmath" set="math" sort="title"
                    totalResults="2" startResult="1" resultSize="2" />
    <record>
      <identifier>cul.math/98765432</identifier>
      <title>Fun with Trigonometry</title>
      <author>Todhunter, I.</author>
      <author>Disney, W.</author>
      <pubdate>1888</date>
      <rank>2859</rank>
      <resultDivs>
```

```
        <divID>cul.math/98765432/00000024</divID>
        <divID>cul.math/98765432/00000128</divID>
        <divID>cul.math/98765432/00000156</divID>
        <divID>cul.math/98765432/00000237</divID>
      </resultDivs>
    </record>
    <record>
      <identifier>cul.math/00640001</identifier>
      <title>Spherical Trigonometry</title>
      <author>Todhunter, I.</author>
      <pubdate>1886</date>
      <rank>2857</rank>
      <resultDivs>
        <divID>cul.math/00640001/00000345</divID>
        <divID>cul.math/00640001/00000346</divID>
      </resultDivs>
    </record>
  </Search>
</CGM>
```

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Search&ver=1.0&set=math&startResult=0
        &field1=fulltext&value1=geometry&field2=fulltext
        &value2=trigonometry&op2=and

http://some.cgm.server/script?protocol=CGM
        &verb=Search&ver=1.0&set=math&resultSize=0
        &field1=fulltext&value1=geometry&field2=fulltext
        &value2=trigonometry&op2=and
```

These two requests are equivalent. The response below is to the first one.

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request protocol="CGM" verb="Search" ver="1.0" set="math" startResult="0"
           field1="fulltext" value1="geometry" field2="fulltext"
           value2="trigonometry" op2="and">
           http://some.cgm.server/script</request>
  <Search ver="1.0">
    <resultsSummary repositoryIdentifier="CULmath" set="math" sort="title"
                    totalResults="1024" startResult="0" resultSize="0" />
  </Search>
</CGM>
```

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Search&ver=1.0&set=math
        &field1=author&value1=bush&field2=fullbib
        &value2=basic%20counting&op2=and
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request protocol="CGM" verb="Search" ver="1.0" set="math"
           field1="author" value1="bush"field2="fullbib"
           value2="basic counting" op2="and">http://some.cgm.server/script
           </request>
  <Search ver="1.0">
    <resultsSummary repositoryIdentifier="CULmath" set="math" sort="title"
                    totalResults="0" startResult="0" resultSize="0" />
  </Search>
</CGM>
```

Verb History

Substantial reworking and extension of Dienst verb SearchBoolean, version 5.0.

Get Document Structure

Verb name: Structure
Verb version: 1.0
Modification Date: 2003-06-18
Required arguments: identifier
Optional arguments: version, view
MIMETYPE of response: text/xml

Verb Description

This verb returns a structured response describing a structural view available for a document. A client may use this structural information as the basis for document requests using the `Disseminate` verb.

Arguments

- **`identifier`** :: [required] the document identifier
- **`version`** :: specifies the document version for which the structure information is requested. If omitted, the response provides information about the latest document version.
- **`view`** :: specifies the the view of the document for which structure information is requested. If omitted, the response provides information about the default view. Specific views are requested using the view id value, obtained using the `ListViews` verb.

Response

Each response will include one `view` element. There will always be one and only one default view of a document.

- **`<view>`** :: contains one high-level, or root, `div` element, within which nested `div` hierarchies are allowed. `view` element attributes are:
  - `id` :: [required] an id value for this view. This id is defined as an XML ID.
  - `label` :: [required] a displayable label identifying this view. These are applied by the local repository and should help end users choose an appropriate document view.
  - `default` :: [required for default view] equals `1` for the sole default view; equals `0` for all other views. If absent, `default="0"` is assumed.
- **`<div>`** :: each `view` element contains one immediate child `div` element. This single `div` may itself contain one or more `div` elements. `div` element attributes are:
  - `id` :: [required] an ID value for this division, usable as is in Formats and other requests. This is not defined as an XML ID.
  - `type` :: type of division.
  - `order` :: [required] the order of this `div` among its siblings. Values are positive integers, beginning with 1.
  - `label` :: a displayable label for this division. These are applied by the local repository and should make sense in a navigational context.
  - `diss` :: indicates whether this division can be requested successfully using the Disseminate verb. Values are "0" (cannot be disseminated), or "1" (can be requested with Disseminate).

Error Codes

- **`idDoesNotExist`** :: the value of the `identifier` argument is unknown or illegal.
- **`badArgument`** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Structure&ver=1.0&identifier=cul.math/00640001
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request protocol="CGM" verb="Structure" ver="1.0"
           identifier="cul.math/00640001">
           http://some.cgm.server/script</request>
  <Structure ver="1.0">
    <identifier value="cul.math/00640001"/>
    <view id="v123-a" label="Page Listing" default="1">
      <div id="a123" type="maindocument" order="1" label="Entire Monograph"
           diss="0">
        <div id="a123-1" type="page" order="1" label="Page NA" diss="1">
        <div id="a123-2" type="page" order="2" label="Page i" diss="1">
        <div id="a123-3" type="page" order="3" label="Page ii diss="1"">
        <div id="a123-4" type="page" order="4" label="Page 1" diss="1">
        <div id="a123-5" type="page" order="5" label="Page 2" diss="1">
        <div id="a123-6" type="page" order="6" label="Page 3" diss="1">
        <div id="a123-7" type="page" order="7" label="Page 4" diss="1">
        <div id="a123-8" type="page" order="8" label="Page 5" diss="1">
        <div id="a123-9" type="page" order="9" label="Page 6" diss="1">
      </div>
    </view>
  </Structure>
</CGM>
```

The response above describes the physical structure of a document. Because no view was specified in the request, the default view was returned. This `diss` attributes indicate that only the individual pages can be requested with the Disseminate verb.

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Structure&ver=1.0&identifier=cul.math/00640001&view=v123-b
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request protocol="CGM" verb="Structure" ver="1.0"
           identifier="cul.math/00640001" view="v123-b">
           http://some.cgm.server/script</request>
  <Structure ver="1.0">
    <identifier value="cul.math/00640001"/>
    <view id="v123-b" label="Chapter Listing" default="0">
      <div id="a123" type="maindocument" order="1" label="Entire Monograph"
           diss="0">
        <div id="a123-1" type="front" order="1" label="Front Matter" diss="1">
          <div id="a123-1-1" type="page" order="1" label="Page i" diss="1">
          <div id="a123-1-2" type="page" order="2" label="Page ii" diss="1">
          <div id="a123-1-3" type="page" order="3" label="Page iii" diss="1">
        </div>
        <div id="a123-2" type="chapter" order="2" label="Chapter I" diss="1">
          <div id="a123-2-1" type="page" order="1" label="Page 1" diss="1">
```

```
          <div id="a123-2-2" type="page" order="2" label="Page 2" diss="1">
          <div id="a123-2-3" type="page" order="3" label="Page 3" diss="1">
          <div id="a123-2-4" type="page" order="4" label="Page 4" diss="1">
          ....
          <div id="a123-2-36" type="page" order="36" label="Page 36" diss="1">
        </div>
        <div id="a123-3" type="chapter" order="3" label="Chapter II" diss="1">
          <div id="a123-3-37" type="page" order="1" label="Page 37" diss="1">
          <div id="a123-3-38" type="page" order="2" label="Page 38" diss="1">
          <div id="a123-3-39" type="page" order="3" label="Page 39" diss="1">
          <div id="a123-3-40" type="page" order="4" label="Page 40" diss="1">
          ....
          <div id="a123-3-77" type="page" order="41" label="Page 75" diss="1">
        </div>
        ...
      </div>
    </view>
  </Structure>
</CGM>
```

This response describes a non-default view of the identified document. The document view is structured into a front section and following chapters, with individual page images available within these structures. ("..." indicates where additional pages and chapters have been omitted in the example for brevity). The `diss` attributes indicate that the pages and also entire chapters can be requested with the Disseminate verb. A Formats request would be needed to determine what formats were available for dissemination.

Verb History

Modification of Dienst verb Structure, version 2.0. CGM has removed all format related functions from the Dienst Structure.

Earlier NSF-DFG note: For `Structure`, we discussed replacing the unnumbered DIVs, which can nest, with numbered DIVs (e.g., DIV1, DIV2, etc.). Further, `Structure` should use real ID values (i.e., only NAME characters, beginning with an alpha character), and need to have a sense of relationship between layers (e.g., what shows first, GIF or PDF?) and a repository's desire to suppress or hold back a layer (e.g., don't show the OCR without this caveat, or don't show it until you've shown other layers, and then only with this caveat).

Terms of Use

Verb name: Terms
Verb version: 1.0
Modification Date: 2003-06-18
Required arguments: identifier
Optional arguments: version
MIMETYPE of response: text/xml

Verb Description

Returns terms of use and rights information for a document.

Arguments

- **`identifier`** :: [required] the document identifier for which terms of use information is requested.
- **`version`** :: specifies the document version for which the terms of use information is requested. If omitted, it provides information about the latest document version available (see ListVersions).

Response

- **<statement>** :: a single `statement` element is returned, the most current statement available for the document requested. A `statement` element can contain one or more `p` elements. `p` elements in turn contain text, a link to another document, or a combination of text and links.
- **<p>** :: a paragraph element, for encoding a text statement.
- **<link>** :: for creating a link. `link` can contain data or be empty. The behavior of the link is controlled with three attributes:
  - `actuate` :: defines whether a link occurs automatically or must be activated by the user.
    - `onload` :: the link is activated, and the resource is displayed, automatically.
    - `onrequest` :: the resource is displayed if the user activates the link.
  - `show` :: determines how a remote resource appears when the link is activated.
    - `embed` :: the target resource displays at the point of the link.
    - `new` :: the target resource appears in a new window.
    - `replace` :: the target resource displays in the same window as the resource that contained the link.
  - `href` :: the locator for a remote resource, expressed as a URL.

Error Codes

- **idDoesNotExist** :: the value of the `identifier` argument is unknown or illegal.
- **badArgument** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.
- **noTermsAvailable** :: there is no terms-of-use statement available for this document.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Terms&ver=1.0&identifier=cul.math/00640001
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="Terms" ver="1.0" identifier="cul.math/00640001">
           http://some.cgm.server/script</request>
  <Terms ver="1.0">
    <identifier value="cul.math/00640001">
    <statement>
      <p>This digital material is made available by Cornell
         University Library, and is to be used for personal or research use
         only. Any other use, including but not limited to commercial or
         scholarly reproductions, redistribution, publication, or transmission,
         whether by electronic means or otherwise, without prior written
         permission of the Library is prohibited.</p>
    </statement>
  </Terms>
</CGM>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="Terms" ver="1.0" identifier="cul.math/00640001">
           http://some.cgm.server/script</request>
  <Terms ver="1.0">
    <identifier value="cul.math/00640001">
    <statement>
      <p>For rights information, see the <link actuate="onrequest"
         show="new" href="http://cdl.library.cornell.edu/guidelines.html">
         Cornell University Library Rights Statement</link>.</p>
    </statement>
  </Terms>
</CGM>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="Terms" ver="1.0" identifier="cul.math/00640001">
           http://some.cgm.server/script</request>
  <Terms ver="1.0">
    <identifier value="cul.math/00640001">
    <statement>
      <p>
         <link actuate="onload" show="replace"
         href="http://cdl.library.cornell.edu/guidelines.html" />
      </p>
    </statement>
  </Terms>
</CGM>
```

Verb History

Search Verb, ver. 1.0

Search Request

`sort` value `rank` is based on whatever relevance data is provided by the repository.

`field` values that must be supported (indexed and searchable):

- `title`
- `author`
- `pubtype` :: indicates whether `monograph` or `serial`
- `language` :: use ISO639-1 or ISO639-2
- `fullbib`
- `pubdate`
- `publisher`

`field` values that may be supported:

- `identifier`
- `subject`
- `abstract`
- `fulltext`

`op` values that must be supported:

- `and`
- `or`

Search Response

`<record>` subelement requirements (for the indexer to return):

- `identifier` :: Required.
- `title` :: Required if available.
- `author` :: Required if available. Repeatable. Each author is in `last, first middle` format.
- `pubdate` :: Required if available. The date of publication of the document is in W3C-DTF format (YYYY-MM-DD, YYYY-MM, YYYY).
- `rank` :: Optional. Relevance data in any form. Should be made as meaningful as possible.
- `resultDivs` :: Optional.

Structure Verb, ver. 1.0

Structure Response

`type` attribute on the element. If any of the following division types are present, they must be identified as such using these values:

- `maindocument`
- `front`
- `body`
- `back`
- `chapter`
- `section`
- `page`

Formats Verb, ver. 1.0

Formats Response

`type` attribute on the <divReq> element. If any of the following division types are present, they must be identified as such using these values:

- `maindocument`
- `front`
- `body`
- `back`
- `chapter`
- `section`
- `page`

NSF-DFG Dienst Implementation

Introduction

Work in this project to accomplish interoperability between three large Math-oriented production digital library systems is funded by the National Science Foundation (NSF) and the Deutsche Forschungsgemeinshaft (DFG). This is *not* the primary site for work on this project; Cornell University Library will put in place a formal site for the project later this year.

This document is a draft intended to capture the conclusions of the first meeting of representatives from Cornell University Library, State and University Library G&#65533;ttingen, and the University of Michigan Library, in their review of the Dienst protocol verbs relative to the proposed work, etc. Each Verb is structured in the following way:

1. `Verb function`: a descriptive label for the verb in question.
2. `Verb name`:
3. `Verb version`:
4. `NSF-DFG-Math usage`: The following values are provided:

label	value
`req`	Required, used *largely* as specified in Dienst protocol (except that name=value pairs after the ? will be used by project participants)
`rec`	Recommended, used as specified in Dienst protocol
`opt`	Optional, used as specified in the Dienst protocol
`dep`	Deprecated, used as specified in the Dienst protocol
`mod`	Required, with modifications to be specified by this project
`rep`	Not used; replaced by a corresponding OAI verb; requires a reference to the appropriate OAI verb

5. `Fixed arguments`:
6. `Keyword arguments`:
7. `MIMETYPE response`:
8. `Status Codes responses`:
9. `Example Request`:
10. `Example Response`:

Note, too that in general project team members will probably be using ISO8859-1 instead of UTF-8 in their character set declarations.

Dienst Service Components

- Repository Service
- Index Service
- Query Mediator Service
- Collection Service
- Info Service

Repository

A repository is administratively divided into a set of partitions. Each partition stores documents, as defined by the Dienst document model, and provides messages to deposit documents, discover their structure, and obtain disseminations of documents. A number of dissemination variables effect the nature of a particular dissemination request. For more on the Repository Service, please see: http://www.cs.cornell.edu/cdlrg/dienst/protocols/DienstProtocol.htm#4.1%20Repository%20Service

Verb function: Describe Verb
Verb name: Describe-Verb
Verb version: 2.0
NSF-DFG-Math usage: req
Fixed arguments: value
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response that contains a list, where each element of the list provides information on a version of the specified verb that is supported by this service. The following information may be provided at the verb or version level.

- **description**, description of the verb or a specific version
- **note**, information pertaining to the verb or a specific version

Each element of the list contains the following information:

- **version number** of the verb.
- **arguments**, a list of the names of the fixed and keyword arguments, if any, accepted by the verb in that version.
- **example** template of request to this repository, with fixed argument indicated in brackets
- **returns**, optional, contains information about response format.

Note that a service may implement more than one version of a verb.

Example Request:

```
Protocol=dienst;Service=Repository;ver=2.0;verb=Describe-Verb;value=Formats
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Describe-Verb version="2.0">
    <Verb name="Formats">
      <description>Returns a structured response indicating
        the formats of disseminations available for this document
      </description>
      <versions>
        <version id="4.0">

              <example>http://www.umdl.umich.edu/cgi/b/broker/broker?
Protocol=dienst;Service=Repository;ver=2.0;verb=Formats;handle=handlevalue></example>
          <arguments>
            <fixed>
              <arg name="handle" />
            </fixed>
            <keyword>
              <arg name="version" />
              <arg name="view" />
            </keyword>
          </arguments>
```

```
        </version>
        <version id="2.0">
          <note>Depricated</note>
          <example>http://cs-
tr.cs.cornell.edu:80/Dienst/Repository/2.0/Formats/<handle></example>
          <arguments>
            <fixed>
              <arg name="handle" />
            </fixed>
          </arguments>
        </version>
      </versions>
    </Verb>
  </Describe-Verb>
```

Verb function: Disseminate Content
Verb name: Disseminate
Verb version: 1.0
NSF-DFG-Math usage: mod
NSF-DFG-Math Note(s): Our tentative position on this is that the record-oriented components are to be replaced by OAI verb `GetRecord`, and that this Dienst verb needs to continue for handling the actual content.
Fixed arguments: handle, view, content-type
Keyword arguments: version, , pageimage, binder, encoding
MIMETYPE response: dependent on dissemination requested.
Status Codes responses: 200, 400, 404, 415, 501, 503

Verb Description:

Request a dissemination of a digital object. The characteristics of the dissemination that can be requested (the arguments to the `Disseminate` verb), are determined by the responses to the `List-Versions`, `Formats`, and `Structure` verbs for the specific digital object. The response is a MIME-typed byte stream. The MIME type of the byte stream is either the MIME type of the specified `content-type` or, if a `binder` is specified, the MIME type of that `binder`. Refer to the examples below for more information on the stream that is returned.

In addition to the `handle`, the required fixed arguments are:

- `view` specifies the view of the document instance from which a dissemination is requested. This argument takes one of three forms:
  - `#<meta-format>` ,where `<meta-format>` is one of the metadata formats (as returned from the `ListMetadataFormats` request). The resulting dissemination is then the top-level (document instance) metadata records in the requested format. For example, `#rfc1807` requests the rfc1807 metadata record for the document instance.
  - `<viewID>#<meta-format>` ,where `<viewID>` is one of the metadata formats (as returned from the `ListMetadataFormats` request). The resulting dissemination is then the view specific metadata records in the requested format. For example, `view=body#rfc1807` requests the rfc1807 metadata record for the `body` view of the document instance.
  - `<viewID>`,where `<viewID>` is one of the available views for the document instance. The resulting dissemination, if there is no `pageimage` argument, is then the specified view of the document instance.
- `content-type` is a *required* argument that specifies the MIME-type of the content in the dissemination. The list of available content-types for a document instance is indicated by the response to the `Formats` request. Note that the value supplied for the argument is not the actual MIME-type, but the tag for that MIME type as indicated in the response to the `Formats` request (because of the complications with the "/" character in HTTP requests). For metadata requests, the only `content-type` that is supported is `xml`. If there is no `binder` argument, then this is the MIME-type of the stream disseminated to the client.

The Keyword Arguments are:

- `version` specifies the document instance from which a dissemination is requested. If omitted, the latest document instance is used by default.
- `<div>` narrows the dissemination request to a specific structural component (e.g., a *chapter*) of the specified view of the document instance. The supplied argument must be one the available `divs` of the view (indicated by the output of the `Structure` request) paired with a value that is an identifier of one of those `divs` (again indicated by the output of the `Structure` request). For example, if a `Structure` request indicates that there is a `div` of type `chapter` with identifier `1`, then this argument could by `chapter=1`. Note that the `div` argument can not be used in combination with the `pageimage` argument.
- `pageimage` narrows the dissemination request to a specific page of the specified view of the document instance. The value must be an identifier of one the available pages of the view (indicated by the output of the `Structure` request).
- `binder` specifies the encapsulating binder for the dissemination. The MIME type of the stream disseminated to the requesting client is then the output of the binder application. A `binder` argument is *required* if the dissemination contains multiple logical objects (e.g., multiple page images in `image/tiff`). The list of Binders supported by a repository is available through the List-Binders protocol request.
- `encoding` specifies the compression encoding scheme that should be applied to the dissemination. The `encoding` applied is reflected in the HTTP `Content-encoding` header returned from the `Disseminate` request. The list of Encodings supported by a repository is available through the List-Encodings protocol request.

Example Request:

```
Protocol=Dienst;Service=Repository;verb=Disseminate;ver=1.0;handle=handlecorp/970101;view=body;
type=postscript
```

Example Response:

Note: The PostScript rendition (MIME-type `postscript`) of "body" view of the document.

Example Request:

```
Protocol=Dienst;Service=Repository;verb=Disseminate;ver=1.0;handle=handlecorp/970101;content-
type=oams/xml
```

Example Response:

Note: The Open Archives metadata record for the document instance (in the only available content-type for metadata which is `xml`).

```
<?xml version="1.0"
encoding="UTF-8"?>
  <Disseminate version="1.0">
    <oams:oams xmlns:oams="http://www.openarchives.org/sfc/sfc_oams.htm">
      <oams:title>A protocol for Interoperable Archives</oams:title>
      <oams:accession date="1994-06-24 />
      <oams:fullId>ncstrl.cornell/TR94-1418</oams:fullId>
      <oams:author>
        <oams:name>James R. Davis</oams:name>
        <oams:organization>Xerox</oams:organization>
      </oams:author>
      <oams:author>
        <oams:name>Carl Lagoze</oams:name>
        <oams:organization>Cornell</oams:organization>
      </oams:author>
    </oams:oams>
  </Disseminate>
```

Example Request:

```
Protocol=Dienst;Service=Repository;verb=Disseminate;ver=1.0;handle=handlecorp/970101;view=book;
```

```
type=gif;pageimage=3
```

Example Response:

Note: Page 3 of the book view of the document instance as a GIF file (MIME type `image/gif`).

Example Request:

```
Protocol=Dienst;Service=Repository;verb=Disseminate;ver=1.0;handle=handlecorp/970101;view=book;
type=gif;chapter=3&binder=tar&encoding=gzip
```

Example Response:

Note: Chapter of the book view of the document instance as a series of `GIF` files, bound together using `tar`, and then compressed using `gzip`. The MIME type of the disseminated byte stream is than `application/tar`.

Verb function: List Formats
Verb name: Formats
Verb version: 4.0
NSF-DFG-Math usage: dep
NSF-DFG-Math Note(s): Subsumed in modification of the verb `Structure`. We also agreed that Formats was inadequate and needed to be tied more closely to Structure. See also the discussion regarding `Structure`.
Fixed arguments: handle
Keyword arguments: version, view
MIMETYPE response: text/xml
Status Codes responses: 200, 400, 404, 501

Verb Description:

Returns a structured response indicating the formats of disseminations available for this document instance.

The available arguments are:

- `version` specifies the document instance for which the format information is requested. If omitted, it provides information about the latest document instance.
- `view` specifies the view of the document instance for which format information is requested. If omitted, it provides information about the entire document instance.

The response includes the following information:

- a list of `content-types` with the following information for each content type:
  - `tag`, which is the string that can be used in a Disseminate request.
  - `MIME-type`, which is the corresponding MIME type of this tag.
  - `size`, an estimate of the number of bytes of the dissemination, expressed in that content-type. This is returned only if the size can be determined.
  - `URL`, an alternative URL (if available) that can be used to download the document instance in this content-type. This is intended for sites wish to offer a "backdoor" to the Dienst protocol for retrieving the respective representation of the document. Note that this alternative URL will not allow the functionality of the Dienst Disseminate request, such as version selection, part selection, and the like

Example Request:

```
Protocol=Dienst;Service=Repository;verb=Formats;ver=4.0;handle=handlecorp/970101?
version=1&part=body
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Formats version="4.0">
    <formats>
      <postscript name="application/postscript"
        size="25555"
        URL="http://foo.com/bar.ps" />
      <gif name="image/gif" />
    </formats>
  </Formats>
```

Verb function: Get Records
Verb name: GetRecord
Verb version: 1.0
NSF-DFG-Math usage: req
NSF-DFG-Math Note(s): This OAI verb is introduced into the NSF/DFG project to bring our version of the protocol into compliance with OAI. It replaces the record-oriented parts of the verb `Disseminate`.
Fixed arguments: none
Keyword arguments: identifier, metadataPrefix
MIMETYPE response: text/xml
Status Codes responses:

Verb Description:

This verb is used to retrieve an individual record (metadata) from an item in a repository. Required arguments specify the identifier, or key, of the requested record and the format of the metadata that should be included in the record.

The arguments are used as follows:

- **identifier**: a REQUIRED argument that specifies the unique identifier that should be used as a key to extract the requested record from an item in the repository.
- **metadataPrefix**: a REQUIRED argument that specifies the metadata prefix of the format that should be included in the metadata part of the returned record. The value of this argument must be a non-space embedded string consisting of alphanumeric and underscore [_] characters; [A-Za-z0-9_]. The metadata formats supported by a repository and for a particular record can be retrieved using the ListMetadataFormats request.

Exception conditions include:

- `Identifier does not exist` - The response will not contain a record container.
- `Metadata format can not be disseminated for the identifier` - The record in the response contains a header but no metadata container.

For more information on this verb, please see:
http://www.openarchives.org/OAI/openarchivesprotocol.htm#GetRecord.

Example Request:

Note: Request a record in the Dublin Core metadata format [URL shown without encoding for better readibility].

```
http://an.oa.org/OAI-script?verb=GetRecord&identifier=oai:arXiv:quant-
ph/9901001&metadataPrefix=oai_dc
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
 <GetRecord     xmlns="http://www.openarchives.org/OAI/1.0/OAI_GetRecord"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
    xsi:schemaLocation="http://www.openarchives.org/OAI/1.0/OAI_GetRecord
                  http://www.openarchives.org/OAI/1.0/OAI_GetRecord.xsd">
  <responseDate>2000-10-10T14:09:57-07:00</responseDate>
  <requestURL>http://an.oa.org/OAI-script?verb=GetRecord
&identifier=oai%3AarXiv%3Aquant-ph%2F9901001
&metadataPrefix=oai_dc</requestURL>
  <record>
   <header>
    <identifier>oai:arXiv:quant-ph/9901001</identifier>
    <datestamp>1999-01-01</datestamp>
   </header>
   <metadata>
   <dc xmlns="http://purl.org/dc/elements/1.1/"
       xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
       xsi:schemaLocation="http://purl.org/dc/elements/1.1/
                           http://www.openarchives.org/OAI/dc.xsd">
     <title>Quantum slow motion</title>
     <creator>Hug, M.</creator>
     <creator>Milburn, G. J.</creator>
     <description>We simulate the center of mass motion of cold atoms in a standing,
amplitude modulated, laser field as an example of a system that has a classical
      mixed phase-space.</description>
     <date>1999-01-01</date>
     <type>e-print</type>
     <identifier>http://arXiv.org/abs/quant-ph/9901001</identifier>
    </dc>
   </metadata>
  </record>
 </GetRecord>
```

Example Request:

Note: Request a record in the Dublin Core metadata format. The requested record, however, can not be returned because the identifier does not exist. Therefore, the response does not contain a record container [URL shown without encoding for better readibility].

```
http://an.oa.org/OAI-script?verb=GetRecord&identifier=oai:arXiv:quant-
ph/0201001&metadataPrefix=oai_dc
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
 <GetRecord     xmlns="http://www.openarchives.org/OAI/1.0/OAI_GetRecord"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
    xsi:schemaLocation="http://www.openarchives.org/OAI/1.0/OAI_GetRecord
                  http://www.openarchives.org/OAI/1.0/OAI_GetRecord.xsd">
  <responseDate>2000-10-10T14:09:57-07:00</responseDate>
  <requestURL>http://an.oa.org/OAI-script?verb=GetRecord
&identifier=oai%3AarXiv%3Aquant-ph%2F0201001
&metadataPrefix=oai_dc</requestURL>
 </GetRecord>
```

Example Request:

Note: Request a record in the oai_marc metadata format. However, the requested metadata format can not be disseminated for this identifier. Therefore, the response record contains a header but no metadata container [URL shown without encoding for better readibility].

```
http://an.oa.org/OAI-script?verb=GetRecord&identifier=oai:arXiv:quant-
ph/9901001&metadataPrefix=oai_marc
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
 <GetRecord     xmlns="http://www.openarchives.org/OAI/1.0/OAI_GetRecord"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
    xsi:schemaLocation="http://www.openarchives.org/OAI/1.0/OAI_GetRecord
                  http://www.openarchives.org/OAI/1.0/OAI_GetRecord.xsd">
  <responseDate>2000-10-10T14:09:57-07:00</responseDate>
  <requestURL>http://an.oa.org/OAI-script?verb=GetRecord
&identifier=oai%3AarXiv%3Aquant-ph%2F9901001
```

```
&metadataPrefix=oai_marc</requestURL>
  <record>
   <header>
    <identifier>oai:arXiv:quant-ph/9901001</identifier>
    <datestamp>1999-01-01</datestamp>
   </header>
  </record>
 </GetRecord>
```

Verb function: List Authorities
Verb name: List-Authorities
Verb version: 1.0
NSF-DFG-Math usage: req
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Return a structured list of the authorities stored in this repository.

Example Request:

```
Protocol=Dienst;Service=Repository;verb=List-Authorities;ver=1.0
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <List-Authorities version="1.0">
    <authority>
      <name>ncstrl.cornell</name>
      <display>Cornell University Computer Science Department</display>
    </authority>
  </List-Authorities>
```

Verb function: List Binders
Verb name: List-Binders
Verb version: 1.0
NSF-DFG-Math usage: req
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Return a structured list of the types of binders available in this repository.

Example Request:

```
Protocol=Dienst;Service=Repository;verb=List-Binders;ver=1.0
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <List-Binders version="1.0">
    <binder>tar</binder>
  </List-Binders>
```

Verb function: List Contents
Verb name: List-Contents
Verb version: 4.0

NSF-DFG-Math usage: rep
NSF-DFG-Math Note(s): This is replaced by the OAI verbs `ListIdentifiers` and `ListRecords`.
Fixed arguments: none
Keyword arguments: partitionspec, file-after, file-before, meta-format
MIMETYPE response: text/xml
Status Codes responses: 200, 400, 501

Verb Description:

Return a structured list of the handles for documents stored in this repository service. Without any arguments the list includes all stored documents.

The meaning of the Keyword Arguments is as follows:

- `file-after` limits the list to those handles for documents that were added or modified since *time*, a universal time expressed in ISO 8601 format. If the server is not able to determine date of modification to the resolution of a day, or if the server is not able to selectively extract records on a time scale of a day, the server may return additional records, e.g. all those modified during the week, month, or even century containing the date.
- `file-before` limits the list to those handles for documents that were added or modified prior to *time*, a universal time expressed in ISO 8601 format. If the server is not able to determine date of modification to the resolution of a day, or if the server is not able to selectively extract records on a time scale of a day, the server may return additional records, e.g. all those modified during the week, month, or even century containing the date.
- `partitionpsec` limits the returned handles to those in the specified partition specification. The partitions available for a repository are returned from the `List-Partitions` request..
- `meta-format` returns, in addition to the handle, metadata for each document in the specified format. The metadata is empty for any document that does not have metadata in that format. Legal values for this argument is one of the metadata formats (as returned from the `ListMetadataFormats` request).

Example Request:

Note: List the handles in the high energy (hep) partition within the physics partition.

```
Protocol=Dienst;Service=Repository;verb=List-Contents;ver=4.0;partitionspec=physics;hep
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <List-Contents version="4.0">
    <record>
      handlecorp/970101
    </record>
    <record>
      handlecorp/970102
    </record>
  </List-Contents>
```

Example Request:

Note: List the rfc1807 metadata along with the handles

```
Protocol=Dienst;Service=Repository;verb=List-
Contents;ver=4.0;partitionspec=physics;hep&meta-format=rfc1807
```

Example Response:

Note: Note the empty rfc1807 record for handlecorp/970102 indicating that there is no rfc1807 metadata for this document.

```
<?xml version="1.0" encoding="UTF-8"?>
  <List-Contents version="4.0">
    <record>
      handlecorp/970101
      <rfc1807:rfc1807 xmlns:rfc1807="ftp://nic.merit.edu/document/rfc/rfc1807.txt">
        <rfc1807:author>William Shakespeare</rfc1807:author>
        <rfc1807:title>Romeo and Juliet</rfc1807:title>
        <rfc1807:id>handlecorp//970101</rfc1807:id>
        <rfc1807:entry>June 24 1506</rfc1807:entry>
      </rfc1807:rfc1807>
    </record>
    <record>
      handlecorp/970102
      <rfc1807:rfc1807 xmlns:rfc1807="ftp://nic.merit.edu/document/rfc/rfc1807.txt"> >
      </rfc1807:rfc1807>
    </record>
  </List-Contents>
```

Verb function: List Identifiers
Verb name: ListIdentifiers
Verb version: 1.0
NSF-DFG-Math usage: req
NSF-DFG-Math Note(s): This OAI verb is introduced into the NSF/DFG project to bring our version of the protocol into compliance with OAI. Along with `ListRecords`, it replaces the Dienst verb `List-Contents`.
Please note: the `resumptionToken` is not implemented at Michigan.
Fixed arguments: none
Keyword arguments: until, from, set, resumptionToken
MIMETYPE response: text/xml
Status Codes responses:

Verb Description:

This verb is used to retrieve the identifiers of records that can be harvested from a repository. Optional arguments permit selectivity of the identifiers - based on their membership in a specific Set in the repository or based on their modification, creation, or deletion within a specific date range.

The arguments are used as follows:

- **until** an OPTIONAL argument with a date value, which specifies that only the unique identifiers of records with a datestamp older than or equal to the specified date should be returned.
- **from** an OPTIONAL argument with a date value, which specifies that only the unique identifiers of records with a datestamp that is more recent than or equal to the specified date should be returned.
- **set** an OPTIONAL argument with a setSpec value, which specifies that only the unique identifiers of records from the specified Set should be returned.
- **resumptionToken** an EXCLUSIVE argument with a value that is the flow control token returned by a previous ListIdentifiers request that issued a partial response.

Exception Conditions include: No records match the request - The response will not contain any identifier elements.

For more information on this verb, please see:
http://www.openarchives.org/OAI/openarchivesprotocol.htm#ListIdentifers.

Example Request:

Note: List the unique identifiers of records added or modified since January 15, 1998 in the "hep" Set of the "physics" Set [URL shown without encoding for better readibility].

```
http://an.oa.org/OAI-script?verb=ListIdentifiers&from=1998-01-15&set=physics:hep
```

Example Response:

Note: A list of four identifiers is returned, one of which has a "deleted" status. In addition, a resumption token has been returned, indicating the list of identifiers is incomplete and one or more subsequent requests will need to be issued to complete the list.

```
<?xml version="1.0" encoding="UTF-8"?>
  <ListIdentifiers       xmlns="http://www.openarchives.org/OAI/1.0/OAI_ListIdentifiers"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
      xsi:schemaLocation="http://www.openarchives.org/OAI/1.0/OAI_ListIdentifiers
                  http://www.openarchives.org/OAI/1.0/OAI_ListIdentifiers.xsd">
    <responseDate>2000-10-01T19:20:30-04:00</responseDate>
    <requestURL>http://an.oa.org/OAI-script?verb=ListIdentifiers
&from=1998-01-15&setSpec=physics%3Ahep</requestURL>
     <identifier>oai:arXiv:hep-th/9801001</identifier>
     <identifier>oai:arXiv:hep-th/9801002</identifier>
     <identifier>oai:arXiv:hep-th/9801005</identifier>
     <identifier status="deleted">oai:arXiv:hep-th/9801010</identifier>
    <resumptionToken>xxx45abttyz</resumptionToken>
  </ListIdentifiers>
```

Example Request:

Note: Issue a subsequent request to the one issued above, with a single resumptionToken argument whose value is that returned in the previous response.

```
http://an.oa.org/OAI-script?verb=ListIdentifiers&resumptionToken=xxx45abttyz
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <ListIdentifiers       xmlns="http://www.openarchives.org/OAI/1.0/OAI_ListIdentifiers"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
      xsi:schemaLocation="http://www.openarchives.org/OAI/1.0/OAI_ListIdentifiers
                  http://www.openarchives.org/OAI/1.0/OAI_ListIdentifiers.xsd">
    <responseDate>2000-10-01T19:20:30-04:00</responseDate>
    <requestURL>http://an.oa.org/OAI- script?verb=ListIdentifiers
              &resumptionToken=xxx45abttyz</requestURL>
     <identifier>oai:arXiv:hep-th/9801020</identifier>
     <identifier>oai:arXiv:hep-th/9801060</identifier>
  </ListIdentifiers>
```

Example Request:

Note: List the unique identifiers of records added or modified on January 1, 2001 in the "hep" Set of the "physics" Set. There are no matches for this request, hence, the response does not contain any identifier tags [URL shown without encoding for better readibility].

```
http://an.oa.org/OAI-script?verb=ListIdentifiers&from=2001-01-01&until=2001-01-
01&set=physics:hep
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <ListIdentifiers       xmlns="http://www.openarchives.org/OAI/1.0/OAI_ListIdentifiers"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
      xsi:schemaLocation="http://www.openarchives.org/OAI/1.0/OAI_ListIdentifiers
                  http://www.openarchives.org/OAI/1.0/OAI_ListIdentifiers.xsd">
    <responseDate>2000-10-01T19:20:30-04:00</responseDate>
    <requestURL>http://an.oa.org/OAI-script?verb=ListIdentifiers
&from=2001-01-01&until=2001-01-01                &setSpec=physics%3Ahep</requestURL>
  </ListIdentifiers>
```

Verb function: List Encodings
Verb name: List-Encodings
Verb version: 1.0

NSF-DFG-Math usage: req
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Return a structured list of the types of encodings available in this repository.

Example Request:

```
Protocol=Dienst;Service=Repository;verb=List-Encodings;ver=1.0
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <List-Encodings version="1.0">
    <Encoding>gzip</Encoding>
  </List-Encodings>
```

Verb function: Get Metadata Formats
Verb name: List-Meta-Formats
Verb version: 1.0
NSF-DFG-Math usage: rep
NSF-DFG-Math Note(s): Replaced by OAI verb `ListMetadataFormats`.
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400, 501

Verb Description:

Returns a structured response indicated the metadata formats that are supported by this repository service. Note that the fact that a metadata format is supported does *not* mean that it is available for all digital objects in that repository. For each metadata format, the following information is returned:

- The `name`, which is the identifier for the metadata format that can be used in other Dienst protocol requests.
- The `namespace ID`, which is a URL that refers to a document describing the metadata format.

Example Request:

```
/Dienst/Repository/1.0/List-Meta-Formats
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <List-Meta-Formats version="1.0">
    <meta-format name="rfc1807"
       namespace="ftp://nic.merit.edu/document/rfc/rfc1807.txt" />
    <meta-format name="dc"
       namespace="http://purl.org/dc" />
  </List-Meta-Formats>
```

Verb function: List Metadata Formats
Verb name: ListMetadataFormats
Verb version: 1.0
NSF-DFG-Math usage: req

NSF-DFG-Math Note(s): This OAI verb is introduced into the NSF/DFG project to bring our version of the protocol into compliance with OAI. It replaces the Dienst verb `List-Meta-Formats`. In my notes from the meeting, I wrote "need agreement on name values" (jpw).
Fixed arguments: none
Keyword arguments: identifier
MIMETYPE response: text/xml
Status Codes responses:

Verb Description:

This verb is used to retrieve the metadata formats available from a repository. An optional argument restricts the request to the formats available for a specific record.

The arguments are used as follows:

- **identifier** an OPTIONAL argument that specifies the unique identifier for which available metadata formats is being requested. If this argument is omitted, then the response includes all metadata formats supported by this repository. Note that the fact that a metadata format is supported by a repository does not mean that it can be disseminated by all items in that repository.

Exception Conditions include: No records match the request - The response will not contain a metadataFormat container.

For more information on this verb, please see:
http://www.openarchives.org/OAI/openarchivesprotocol.htm#ListMetadataFormats.

Example Request:

Note: List the metadata formats that can be disseminated from the record with unique identifier oai:arXiv:hep-th/9901001 [URL shown without encoding for better readibility].

```
http://an.oa.org/OAI-script?verb=ListMetadataFormats&identifier=oai:arXiv:hep-th/9901001
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <ListMetadataFormats
xmlns="http://www.openarchives.org/OAI/1.0/OAI_ListMetadataFormats"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
      xsi:schemaLocation="http://www.openarchives.org/OAI/1.0/OAI_ListMetadataFormats
                  http://www.openarchives.org/OAI/1.0/OAI_ListMetadataFormats.xsd">
    <responseDate>2000-10-01T19:20:30-04:00</responseDate>
    <requestURL>http://an.oa.org/OAI-script?verb=ListMetadataFormats
               &identifier=oai%3AarXiv%3Ahep-th%2F9901001</requestURL>
    <metadataFormat>
      <metadataPrefix>oai_rfc1807</metadataPrefix>
      <schema>http://www.openarchives.org/OAI/rfc1807.xsd</schema>
    </metadataFormat>
    <metadataFormat>
      <metadataPrefix>oai_dc</metadataPrefix>
      <schema>http://www.openarchives.org/OAI/dc.xsd</schema>
      <metadataNamespace>http://purl.org/dc/elements/1.1/</metadataNamespace>
    </metadataFormat>
</ListMetadataFormats>
```

Example Request:

Note: List the metadata formats that can be disseminated from the repository http://an.oa.org/OAI-script.

```
http://an.oa.org/OAI-script?verb=ListMetadataFormats
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <ListMetadataFormats
xmlns="http://www.openarchives.org/OAI/1.0/OAI_ListMetadataFormats"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
      xsi:schemaLocation="http://www.openarchives.org/OAI/1.0/OAI_ListMetadataFormats
                    http://www.openarchives.org/OAI/1.0/OAI_ListMetadataFormats.xsd">
    <responseDate>2000-10-01T19:20:30-04:00</responseDate>
    <requestURL>http://an.oa.org/OAI-script?verb=ListMetadataFormats</requestURL>
    <metadataFormat>
      <metadataPrefix>oai_rfc1807</metadataPrefix>
      <schema>http://www.openarchives.org/OAI/rfc1807.xsd</schema>
    </metadataFormat>
    <metadataFormat>
      <metadataPrefix>oai_dc</metadataPrefix>
      <schema>http://www.openarchives.org/OAI/dc.xsd</schema>
      <metadataNamespace>http://purl.org/dc/elements/1.1/</metadataNamespace>
    </metadataFormat>
    <metadataFormat>
      <metadataPrefix>oai_marc</metadataPrefix>
      <schema>http://www.openarchives.org/OAI/oai_marc.xsd</schema>
    </metadataFormat>
</ListMetadataFormats>
```

Example Request:

Note: List the metadata formats that can be disseminated from the record with unique identifier oai:arXiv:hep-th/0101001. The identifier, however, does not exist and therefore, the response does not contain a metadataFormat container. [URL shown without encoding for better readibility].

```
http://an.oa.org/OAI-script?verb=ListMetadataFormats&identifier=oai:arXiv:hep-th/0101001
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <ListMetadataFormats
xmlns="http://www.openarchives.org/OAI/1.0/OAI_ListMetadataFormats"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
      xsi:schemaLocation="http://www.openarchives.org/OAI/1.0/OAI_ListMetadataFormats
                    http://www.openarchives.org/OAI/1.0/OAI_ListMetadataFormats.xsd">
    <responseDate>2000-10-01T19:20:30-04:00</responseDate>
    <requestURL>http://an.oa.org/OAI-script?verb=ListMetadataFormats
                &identifier=oai%3AarXiv%3Ahep-th%2F0101001</requestURL>
</ListMetadataFormats>
```

Verb function: List Partitions
Verb name: List-Partitions
Verb version: 2.0
NSF-DFG-Math usage: rep
NSF-DFG-Math Note(s): Replaced by OAI verb `ListSets`
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400, 501

Verb Description:

Return a structured list of the administrator-determined partitions for documents stored in this repository service. The list contains the hierarchy of partitions and sub-partitions. For each partition, both the short name and long description is returned.

Example Request:

```
/Dienst/Repository/2.0/List-Partitions
```

Example Response:

Note: The following response indicates a partition hierarchy with two top level

partitions - `Oceanside` and `ValleyView` - each with partitions hierarchies within them.

```
<?xml version="1.0" encoding="UTF-8"?>
  <List-Partitions version="2.0">
    <partition name="Oceanside">
      <display>Oceanside University of Nebraska</display>
      <partition name="CompEnt">
        <display>Department of Computational Entomology</display>
      </partition>
      <partition name="MetPhen">
        <display>Department of Metaphysical Phenomenology</display>
      </partition>
    </partition>
    <partition name="ValleyView">
      <display>Valley View University of Florida</display>
      <partition name="Fren">
        <display>Department of Frenetics</display>
      </partition>
      <partition name="Hist">
        <display>Department of Histrionics</display>
      </partition>
    </partition>
  </List-Partitions>
```

Note: This example shows the nested partition lay out described earlier.

Verb function: List Records
Verb name: ListRecords
Verb version: 1.0
NSF-DFG-Math usage: req
NSF-DFG-Math Note(s): This OAI verb is introduced into the NSF/DFG project to bring our version of the protocol into compliance with OAI. Along with `ListIdentifiers`, replaces the Dienst verb `List-Contents`.
Fixed arguments: none
Keyword arguments: until, from, set, resumptionToken, metadataPrefix
MIMETYPE response: text/xml
Status Codes responses:

Verb Description:

This verb is used to harvest records from a repository. Optional arguments permit selectivity of the harvesting - based on the membership of records in a specific Set in the repository or based on their modification, creation, or deletion within a specific date range.

The arguments are used as follows:

- **until** an OPTIONAL argument with a date value, which specifies that only records with a datestamp older than or equal to the specified date should be returned.
- **from** an OPTIONAL argument with a date value, which specifies that only records with a datestamp that is more recent than or equal to the specified date should be returned.
- **set** an OPTIONAL argument with a setSpec value, which specifies that only records from the specified Set should be returned.
- **resumptionToken** an EXCLUSIVE argument with a value that is the flow control token returned by a previous ListRecords request that issued a partial response.
- **metadataPrefix** a REQUIRED argument that specifies the metadata prefix of the format that should be included in the metadata part of the returned records. The value of this argument must be a non-space embedded string consisting of alphanumeric and underscore [_] characters; [A-Za-z0-9_]. The metadata formats supported by a repository and for a particular record can be retrieved using the ListMetadataFormats request.

Exception conditions include:

- *No records match the request* - The response will not contain any record containers.
- *Requested metadata format can not be disseminated for a matching record* - The response record contains a header but no metadata container.

For more information on this verb, please see:
http://www.openarchives.org/OAI/openarchivesprotocol.htm#ListRecords.

Example Request:

Note: List the records in rfc1807 format added or modified since January 15, 1998 in the "hep" set of the "physics" set [URL shown without encoding for better readibility].

```
http://an.oa.org/OAI-script?verb=ListRecords&from=1998-01-
15&set=physics:hep&metadataPrefix=oai_rfc1807
```

Example Response:

Note:

Three records are returned:

- The first includes the rfc1807 metadata. This record also has an "about" part.
- The second only has a header without the rfc1807 metadata container because the metadata for that record can not be expressed in rfc1807 format.
- The third has a status="deleted" attribute (and therefore no metadata part).

```
<?xml version="1.0" encoding="UTF-8"?>
  <ListRecords       xmlns="http://www.openarchives.org/OAI/1.0/OAI_ListRecords"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
      xsi:schemaLocation="http://www.openarchives.org/OAI/1.0/OAI_ListRecords
                          http://www.openarchives.org/OAI/1.0/OAI_ListRecords.xsd">
    <responseDate>2000-10-01T19:20:30-04:00</responseDate>
    <requestURL>http://an.oa.org/OAI-script?verb=ListRecords                &from=1998-
01-15&setSpec=physics%3Ahep                 &metadataPrefix=oai_rfc1807</requestURL>
    <record>
      <header>
        <identifier>oai:arXiv:hep-th/9901001</identifier>
        <datestamp>1999-12-25</datestamp>
      </header>
      <metadata>
       <rfc1807          xmlns="http://info.internet.isi.edu:80/in-
notes/rfc/files/rfc1807.txt"
         xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
         xsi:schemaLocation=             "http://info.internet.isi.edu:80/in-
notes/rfc/files/rfc1807.txt
              http://www.openarchives.org/OAI/rfc1807.xsd">          <bib-
version>v2</bib-version>          <id>hep-th/9901001</id>
          <entry>January 1, 1999</entry>          <title>Investigations of
Radioactivity</title>
          <author>Ernest Rutherford</author>
          <date>March 30, 1999</date>
        </rfc1807>
      </metadata>
      <about>
       <dc xmlns="http://purl.org/dc/elements/1.1/"
         xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
         xsi:schemaLocation="http://purl.org/dc/elements/1.1/
                             http://www.openarchives.org/OAI/dc.xsd">
        <publisher>Los Alamos arXiv</publisher>
        <rights>Metadata may be used without restrictions as long as the oai
identifier remains attached to it.</rights>
```

```
        </dc>
      </about>
    </record>
    <record>
      <header>
        <identifier>oai:arXiv:hep-th/9901004</identifier>
        <datestamp>1999-12-26</datestamp>
      </header>
    </record>
    <record status="deleted">
      <header>
        <identifier>oai:arXiv:hep-th/9901007</identifier>
        <datestamp>1999-12-21</datestamp>
      </header>
    </record>
  </ListRecords>
```

Verb function: List Sets
Verb name: ListSets
Verb version: 1.0
NSF-DFG-Math usage: req
NSF-DFG-Math Note(s): This OAI verb is introduced into the NSF/DFG project to bring our version of the protocol into compliance with OAI. It replaces the Dienst verb `List-Partitions`.
Fixed arguments: none
Keyword arguments: resumptionToken
MIMETYPE response: text/xml
Status Codes responses:

Verb Description:

This verb is used to retrieve the set structure in a repository.

The arguments are used as follows:

- **resumptionToken** an EXCLUSIVE argument with a value that is the flow control token returned by a previous ListSets request that issued a partial response.

Exception conditions include: Repository contains no set hierarchy - The response will not contain any set containers.

For more information on this verb, please see:
http://www.openarchives.org/OAI/openarchivesprotocol.htm#ListSets.

Example Request:

```
http://an.oa.org/OAI-script?verb=ListSets
```

Example Response:

Note: The following response indicates a set hierarchy with two top level sets - Oceanside and ValleyView - each with two sets within them.

```
<?xml version="1.0" encoding="UTF-8"?>
  <ListSets       xmlns="http://www.openarchives.org/OAI/1.0/OAI_ListSets"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
      xsi:schemaLocation="http://www.openarchives.org/OAI/1.0/OAI_ListSets
                  http://www.openarchives.org/OAI/1.0/OAI_ListSets.xsd">
    <responseDate>2000-10-01T19:20:30-04:00</responseDate>
    <requestURL>http://an.oa.org/OAI-script?verb=ListSets</requestURL>
    <set>
      <setSpec>Oceanside</setSpec>
      <setName>Oceanside University of Nebraska</setName>
    </set>
    <set>
```

```
      <setSpec>Oceanside:CompEnt</setSpec>
      <setName>Department of Computational Entomology</setName>
    </set>
    <set>
      <setSpec>Oceanside:MetPhen</setSpec>
      <setName>Department of Metaphysical Phenomenology</setName>
    </set>
    <set>
      <setSpec>ValleyView</setSpec>
      <setName>Valley View University of Florida</setName>
    </set>
    <set>
      <setSpec>ValleyView:Fren</setSpec>
      <setName>Department of Frenetics</setName>
    </set>
    <set>
      <setSpec>ValleyView:Hist</setSpec>
      <setName>Department of Histrionics</setName>
    </set>
</ListSets>
```

Example Request:

```
http://another.oa.org/OAI-script?verb=ListSets
```

Example Response:

Note: The response shows that the repository does not have a set hierarchy.

```
<?xml version="1.0" encoding="UTF-8"?>
  <ListSets       xmlns="http://www.openarchives.org/OAI/1.0/OAI_ListSets"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
      xsi:schemaLocation="http://www.openarchives.org/OAI/1.0/OAI_ListSets
                  http://www.openarchives.org/OAI/1.0/OAI_ListSets.xsd">
    <responseDate>2000-10-01T19:20:30-04:00</responseDate>
    <requestURL>http://another.oa.org/OAI-script?verb=ListSets</requestURL>
  </ListSets>
```

Verb function: List Verbs
Verb name: List-Verbs
Verb version: 2.0
NSF-DFG-Math usage: req
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response containing the name of the verbs defined by this service.

Example Request:

```
Protocol=Dienst;Service=Repository;verb=List-Verbs;ver=2.0
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <List-Verbs version="2.0">
    <verb>Disseminate</verb>
    <verb>Formats</verb>
    <verb>List-Contents</verb>
    <verb>List-Verbs</verb>
  </List-Verbs>
```

Verb function: List Document Versions
Verb name: List-Versions

Verb version: 1.0
NSF-DFG-Math usage: req
Fixed arguments: handle
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400, 501

Verb Description:

Returns a structured response describing the current versions (*document instances*) available for the requested handle. The information returned consists of:

`id`
: is the version number, a positive integer

`date`
: is the date the version was last modified, expressed as in ISO 8601, or * if this can't be determined

`comment`
: is a comment or description. It may not contain a newline.

Example Request:

```
Protocol=Dienst;Service=Repository;verb=List-Versions;ver=1.0;handle=handlecorp%2f970101
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <List-Versions version="1.0">
    <version id="2">
      <date>1999-02-01</date>
      <comment>this version</comment>
    </version>
    <version id="1">
      <date>1998-02-01</date>
      <comment>that version</comment>
    </version>
  </List-Versions>
```

Verb function: Submit a New Version of a Document
Verb name: New-Version
Verb version: 1.0
NSF-DFG-Math usage: opt
Fixed arguments: handle
Keyword arguments: comment
MIMETYPE response: text/xml
Status Codes responses: 200, 400, 401, 404, 501

Verb Description:

Submit a new version of a document with identifier `handle` that is currently stored in the repository. The version number of the new version is then one greater than the previous most current version (and it becomes the *default* version for `Disseminate` and `Structure` requests are made without `version` arguments).

Note on Permissions: It is the responsibility of the repository service implementation to provide some level of access control over the versioning process. One common mechanism is for a repository service implementation to only accept `New-Version` requests from clients with specific DNS client identifiers. In this case the client is left with the responsibility that the party requesting a new version is authorized to do so.

This is transmitted as a HTTP `POST` request where the input stream is a MIME type multipart/mixed. This input stream has two parts ordered as follows:

- an RFC1807 metadata file with MIME type text/x-rfc1807.
- the *content* in one of the formats that this repository is capable of accepting as an input format. Refer to the `Submit-Formats` verb for the method of retrieving the input formats acceptable to a repository.

The meaning of the arguments is as follows:

- `comment` - a free text description of the new version.

If the new version request is successful the structured return contains:

- `version #` of the latest version.
- `date` the version was submitted
- `comment` for this version.

Example Request:

Note: In addition an input stream should be supplied with the POST that includes this request.

```
Protocol=Dienst;Service=Repository;verb=New-
Version;ver=1.0;description=fixed+wrong+algorithm
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <New-Version version="1.0">
    <version id="2">
      <date>1999-02-01></date>
      <comment>fixed wrong algorithm</comment>
    </version>
  </New-Version>
```

Verb function: Get Document Structure
Verb name: Structure
Verb version: 2.0
NSF-DFG-Math usage: mod
NSF-DFG-Math Note(s): For `Structure`, we discussed replacing the unnumbered DIVs, which can nest, with numbered DIVs (e.g., DIV1, DIV2, etc.). Further, `Structure` should use real ID values (i.e., only NAME characters, beginning with an alpha character), and need to have a sense of relationship between layers (e.g., what shows first, GIF or PDF?) and a repository's desire to suppress or hold back a layer (e.g., don't show the OCR without this caveat, or don't show it until you've shown other layers, and then only with this caveat).
Fixed arguments: handle
Keyword arguments: version, view, content-type
MIMETYPE response: text/xml
Status Codes responses: 200, 400, 404, 415, 501, 503

Verb Description:

This verb returns a structured response that describes the decompositions available for a document instance. A client may use the decomposition information as the basis for document requests using the `Disseminate` verb.

The meaning of the Keyword Arguments is as follows:

- `version` specifies the document instance for which the structure information is requested. If omitted, it provides information about the latest document instance.
- `view` specifies the view of the document instance for which structure information is requested. If omitted, it provides information about the entire document instance. The

distinguished view `#` specifies that only the types of metadata available for the document instance should be returned.
- `content-type` specifies a particular content type for which structure information is requested. If omitted, it provides information about all decompositions for all content-types.

The response lists the `meta-formats`, `views`, `divs`, and `pageimages` as described above.

Example Request:

```
Protocol=Dienst;Service=Repository;verb=Structure;ver=2.0;handle=handlecorp/970101;version=1
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Structure version="2.0">
    <meta-formats>
      <rfc1807 />
      <dc />
    </meta-formats>
    <view id="book">
      <div id="section 1">
        <div id="chapter 1">
          <pageimage id="1">
          <pageimage id="2">
          <pageimage id="3">
        </div>
      </div>
    </view>
  </Structure>
```

Note: This response says that the document instance can disseminate two metadata formats at the top level, `rfc1807` and `dc` (Dublin Core). The document instance also has one disseminable view called `body`. This view is structured into a section and chapter hierarchy, with individually disseminable pages available at the chapter. (Note that additional sections and chapters have been omitted in the example for brevity).

Example Request:

```
Protocol=Dienst;Service=Repository;verb=Structure;ver=2.0;handle=handlecorp/970101
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Structure version="2.0">
    <meta-format>
      <rfc1807 />
    </meta-format>
    <view id="body" divs="pageimage" min="1" max="150" />
  </Structure>
```

Note: This is the "shorthand" response for documents with simple structure. The response indicates one metadata type for the document instance, `rfc1807`, and a single view, `body`, with a set of sequential pages numbered from 1 to 150 inclusive.

Example Request:

```
Protocol=Dienst;Service=Repository;verb=Structure;ver=2.0;handle=handlecorp/970101;view=body
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Structure version="2.0">
    <view id="body" divs="pageimage" min="1" max="150" />
```

```
  </Structure>
```

Note: This request, and the associated response, demonstrates the use of the view argument in the request, which specifies that structure information only for the specified view should be returned.

Example Request:

```
/Dienst/Repository/2.0/Structure/handlecorp/970101?view=%23
```

Note: This request specifies that only the metadata formats for the document should be returned.

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Structure version="2.0">
    <meta-format>
      <rfc1807 />
      <dc />
    </meta-format>
  </Structure>
```

Note: This response says that the document can disseminate two metadata formats `rfc1807` and `dc` (Dublin Core).

Verb function: Submit a Document
Verb name: Submit
Verb version: 1.0
NSF-DFG-Math usage: opt
Fixed arguments:
Keyword arguments: id, partitionspec, authority
MIMETYPE response: text/xml
Status Codes responses: 200, 400, 401, 501

Verb Description:

Submit a new document to be stored in the repository. This is transmitted as a HTTP `POST` request where the input stream is a MIME type multipart/mixed. This input stream has two parts ordered as follows:

- an RFC1807 metadata file with MIME type text/x-rfc1807.
- the *content* in one of the formats that this repository is capable of accepting as an input format. Refer to the `Submit-Formats` verb for the method of retrieving the input formats acceptable to a repository.

The meaning of the arguments is as follows:

- `id` is a *suggested* handle that should be given to this document. The repository may reject the submission because the handle is not unique, doesn't conform to the naming conventions for the repository, or if the repository service does not allow client-specified names. If the id argument is omitted then the repository service will assign its own name.
- `partitionpec` is a partition specification identifying the partition in the repository into which the document should be deposited. The `List-Partitions` verb provides information on the available partitions in the repository. This argument is *required* if the respective repository

has multiple partitions.
- `authority` is required and indicates the naming authority to associate with this document. The repository must be configured to support the authority specified for this document..

If the submit request is successful the structured return contains:

- `handle` of the submitted document.
- `partitionspec` indicates the partition specification for the partition which the document was submitted.

Note on Permissions: It is the responsibility of the repository service implementation to provide some level of access control over the submission process. One common mechanism is for a repository service implementation to only accept `Submit` requests from clients with specific DNS client identifiers.

Example Request:

```
Protocol=Dienst;Service=Repository;verb=Submit;ver=2.0;id=handlecorp/970101;partitionspec=comps
```

Note: In addition an input stream should be supplied with the POST that includes this request.

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Submit version="1.0">
    <handle>handlecorp/970101</handle>
    <partition name="compsci">
      <display>
        Computer Science
      </display>
      <partition name="ai">
        <display>
          Artificial Intelligence
        </display>
      </partition>
    <partition>
  </Submit>
```

Verb function: Get Submit Formats
Verb name: Submit-Formats
Verb version: 1.0
NSF-DFG-Math usage: opt
Fixed arguments:
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400, 501

Verb Description:

Return a structured list of the MIME types of content streams that are accepted by this server in the `Submit` verb.

Example Request:

```
Protocol=Dienst;Service=Repository;verb=Submit-Formats;ver=1.0
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Submit-Formats version="1.0">
    <format>application/postscript</format>
    <format>application/pdf</format>
```

```
  </Submit-Formats>
```

Verb function: Get Terms and Conditions
Verb name: Terms
Verb version: 4.0
NSF-DFG-Math usage: req
Fixed arguments: handle
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400, 404, 501

Verb Description:

Returns a natural language statement of the terms and conditions for use of the object. The intention is not to create a contract, but only to advise a human of the reasonable expectations of the server or author.

Example Request:

```
Protocol=Dienst;Service=Repository;verb=Terms;ver=4.0;handle=handlecorp/970101
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Terms version="4.0">
    <text>Copyright 1999 by the Regents of Oceanview University, Kansas
    </text>
  </Terms>
```

Verb function: Withdraw a Document
Verb name: Withdraw
Verb version: 1.0
NSF-DFG-Math usage: opt
Fixed arguments: handle
Keyword arguments: reason, delete, nosave
MIMETYPE response: text/xml
Status Codes responses: 200, 400, 401, 404, 501

Verb Description:

Removes a document from a repository. The following arguments are available:

- `reason` is a text string specifying the reason for withdrawal. A common use of this string is to state a new access point for the document.
- `delete` specifies that the information in the bibliographic record (metadata) for the document should be deleted. If not specified, then the metadata will remain, but the content will be deleted.
- `nosave` is a suggestion to the repository that the deleted content should not be archived (e.g., maintained by the repository but not accessible through protocol.

If the withdraw request is successful the structured return contains:

- `handle` of the withdrawn document.

Note on Permissions: It is the responsibility of the repository service implementation to provide some level of access control over the withdraw process. One common mechanism is for a repository service implementation to only accept `Withdraw` requests from clients with specific DNS client identifiers. In this case the client is left with the responsibility that the party requesting a withdrawal is authorized to do so.

Note on Policy: The purpose of the `Withdraw` protocol request is not to dictate policy but provide a mechanism for the implementation of a range of policies. It is expected that each repository will have individual policies for document withdrawal. For example, some repositories may not allow withdrawal at all (will always respond with a `401` error). In other cases, a repository may allow withdrawal of content but insist on maintaining access to the bibliographic record.

Example Request:

```
Protocol=Dienst;Service=Repository;verb=Withdraw;ver=1.0;handle=handlecorp/970101;reason=I+was+
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Withdraw version="1.0">
    <handle>handlecorp/970101</handle>
  </Withdraw>
```

Index Service

The index service searches a set of descriptions of documents and return handles for those that match. Typically an index service operates by harvesting metadata from one or more Repository services and storing that metadata in an easily searched index. For more information, please see: http://www.cs.cornell.edu/cdlrg/dienst/protocols/DienstProtocol.htm#4.2%20Index%20Service.

Verb function: Describe Verb
Verb name: Describe-Verb
Verb version: 2.0
NSF-DFG-Math usage: req
Fixed arguments: value
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response that contains a list, where each element of the list provides information on a version of the specified verb that is supported by this service. The following information may be provided at the verb or version level.

- `description`, description of the verb or a specific version
- `note`, information pertaining to the verb or a specific version

Each element of the list contains the following information:

- `version number` of the verb.
- `arguments`, a list of the names of the `fixed` and `keyword` arguments, if any, accepted by the verb *in that version*.
- `example` template of request to this repository, with fixed argument indicated in brackets
- `returns` optional, contains information about response format.

Note that a service may implement more than one version of a verb.

Example Request:

```
Protocol=Dienst;Service=Index;ver=2.0;verb=Describe-Verb;value=SearchBoolean
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Describe-Verb version="2.0">
    <Verb name="SearchBoolean">
      <description>Handles search requests for this index server.
      </description>
      <version id="6.0">
        <example>http://broker.umdl.umich.edu/cgi/b/broker/broker?
protocol=Dienst;Service=Index;ver=6.0;verb=SearchBoolean</example>

        <arguments>
          <keyword>
            <arg name="title" />
            <arg name="itemno" />
            <arg name="author" />
            <arg name="pubtype" />
            <arg name="subject" />
            <arg name="language" />
            <arg name="fullbib" />
            <arg name="abstract" />
            <arg name="pubdate" />
            <arg name="publisher" />
            <arg name="authority" />
```

```
            <arg name="added-after" />
          </keyword>
        </arguments>
        <arguments>
          <keyword>
            <arg name="fieldN" />
            <arg name="valueN" />
            <arg name="opN" />
          </keyword>
        </arguments>
        <arguments>
          <keyword>
            <arg name="and" />
            <arg name="or" />
            <arg name="not" />
            <arg name="within" />
            <arg name="including" />
          </keyword>
        </arguments>
      </version>
    </Verb>
  </Describe-Verb>
```

Verb function: Get Tags Returned from a Query
Verb name: Header-Tags
Verb version: 1.0
NSF-DFG-Math usage: mod
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Return a structured response that is a list of the information elements that are returned from a SearchBoolean request.

Example Request:

```
Protocol=Dienst;Service=Index;ver=1.0;verb=Header-Tags
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Header-Tags version="1.0">
    <tag>handle</tag>
    <tag>rank</tag>
    <tag>author</tag>
    <tag>title</tag>
    <tag>date</tag>
  </Header-Tags>
```

Verb function: Submit a Query
Verb name: SearchBoolean
Verb version: 6.0
NSF-DFG-Math usage: mod
Fixed arguments: none
Keyword arguments: see below
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Specifies a search request to the index server. Keyword Arguments are a set of bibliographic fields and values specifying the search criteria. Returns a structured response that is a list of each document that matches the search criteria. Each element of the list has the following

contents:

- The `handle` of the document.
- The `relevance information` of the document for the specific query, if available.
- The `free-text title` of the document, if available
- The `author(s)` of the document, if available. Each author is in `last, first middle` format.
- The `date of publication` of the document in ISO 8601 format. The exact meaning of the term "date of publication" is determined by the administrator of the index service. It is included in the record that is returned so that user interface services may use the date for ordering of a result set.

Note that clients should not assume any logical ordering of the records returned.

Bibliographic Field Arguments:

Valid bibliographic field arguments can vary among index servers. The `Describe-Verb` verb returns the set of field arguments for a particular server. The minimal and standard set of bibliographic field arguments is as follows:

- **Parameters** (available in enumerated form, as in field1, field2, etc.)
  - field
  - value
  - operator
- **Fields searched**
  - title (req)
  - itemno (opt)
  - author (req)
  - pubtype (req, indicates whether monograph or serial))
  - subject (opt)
  - language (req)
  - fullbib (req)
  - abstract (opt)
  - pubdate (req)
  - publisher (req)
- **Operators**
  - and (req)
  - or (req)
  - not (opt)
  - within (opt)
  - including (opt)

Scope Argument:

Two additional fields are available:

`authority`
: the name of the authority in the index server to which the search is to be limited. The default is `all` authorities. The `authority` argument may be repeated, in which case the the search is carried out in each authority (effectively or'ing the authority arguments).

`added-after`
: limits the list to those handles for documents that were added or modified since *time*, a universal time expressed in ISO 8601 format. If the server is not able to determine date of modification to the resolution of a day, or if the server is not able to selectively extract records on a time scale of a day, the server may return additional records, e.g. all those modified during the week, month, or even century containing the date.

Rules for bibliographic field matching: A token in a field is either an unquoted word or a quoted string. Tokens are matched to bibliographic entries according to the following rules:

- Words or phrases are matched exactly as submitted; the wild card symbol * should be used to indicate truncation.
-

Example Request:

```
Protocol=Dienst;Service=Index;ver=6.0;verb=SearchBoolean;field1=author;value1=davis;op1=or;fiel
```

Example Request:

```
Protocol=Dienst;Service=Index;ver=6.0;verb=SearchBoolean;field1=author;value1=donald;op1=and;fi
```

Example Request:

```
Protocol=Dienst;Service=Index;ver=6.0;verb=SearchBoolean;field1=author;value1=donald;op1=or;fie
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <SearchBoolean version="6.0">
    <record>
       <handle>handlecorp/1112</handle>
       <rank>2857</rank>
       <author>Doe, J.</author>
       <author>Public, J. Q.</author>
       <title>Answering Queries</title>
       <date>19941106</date>
    </record>
    <record>
       <handle>handlecorp/1110</handle>
       <rank>2859</rank>
       <author>Mouse, M.</author>
       <author>Disney, W.</author>
       <title>Success with Cartoons</title>
       <date>19941106</date>
    </record>
  </SearchBoolean>
```

Verb function: List Verbs
Verb name: List-Verbs
Verb version: 2.0
NSF-DFG-Math usage: yes
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response containing the name of the verbs defined by this service.

Example Request:

```
Protocol=Dienst;Service=Index;ver=2.0;verb=List-Verbs
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <List-Verbs version="2.0">
    <verb>SearchBoolean</verb>
    <verb>List-Verbs</verb>
  </List-Verbs>
```

Query Mediator Service

The query mediator service performs searches over the entire collection. For more information, please consult: http://www.cs.cornell.edu/cdlrg/dienst/protocols/DienstProtocol.htm#4.3%20Query%20Mediator%20Service .

Verb function: Describe Verb
Verb name: Describe-Verb
Verb version: 2.0
NSF-DFG-Math usage: req
Fixed arguments: verb
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response that contains a list, where each element of the list provides information on a version of the specified verb that is supported by this service. The following information may be provided at the verb or version level.

- `description`, description of the verb or a specific version
- `note`, information pertaining to the verb or a specific version

Each element of the list contains the following information:

- `version number` of the verb.
- `arguments`, a list of the names of the `fixed` and `keyword` arguments, if any, accepted by the verb *in that version*.
- `example` template of request to this repository, with fixed argument indicated in brackets
- `returns`**,** optional, contains information about response format.

Note that a service may implement more than one version of a verb.

Example Request:

```
Protocol=Dienst;Service=QM;ver=2.0;verb=Describe-Verb;args=SearchBoolean
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Describe-Verb version="2.0">
    <Verb name="SearchBoolean">
      <description>Handles search requests for collection.
      </description>
      <version id="2.0">
        <example>http://cs-tr.cs.cornell.edu:80/Dienst/QM/2.0/SearchBoolean</example>
        <arguments>
          <keyword>
            <arg name="title" />
            <arg name="author" />
            <arg name="abstract" />
            <arg name="keywords" />
            <arg name="boolean" />
            <arg name="file-after" />
          </keyword>
        </arguments>
      </version>
    </Verb>
  </Describe-Verb>
```

Verb function: List Verbs
Verb name: List-Verbs
Verb version: 2.0
NSF-DFG-Math usage: req
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response containing the name of the verbs defined by this service.

Example Request:

```
Protocol=Dienst;Service=QM;ver=2.0;verb=List-Verbs
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <List-Verbs version="2.0">
    <verb>SearchBoolean</verb>
    <verb>List-Verbs</verb>
  </List-Verbs>
```

Verb function: Submit a Query
Verb name: SearchBoolean
Verb version: 3.0
NSF-DFG-Math usage: mod
Fixed arguments: none
Keyword arguments:

- **Bibliographic Field Arguments**
  - Valid bibliographic field arguments can vary among index servers. The `Describe-Verb` verb returns the set of field arguments for a particular server. The minimal and standard set of bibliographic field arguments is as follows:
    - **Parameters** (available in enumerated form, as in field1, field2, etc.)
      - field
      - value
      - operator
    - **Fields searched**
      - title (req)
      - itemno (opt)
      - author (req)
      - pubtype (req, indicates whether monograph or serial))
      - subject (opt)
      - language (req)
      - fullbib (req)
      - abstract (opt)
      - pubdate (req)
      - publisher (req)
    - **Operators**
      - and (req)
      - or (req)
      - not (opt)
      - within (opt)
      - including (opt)
- **Scope Argument**: Two additional fields are available:

`authority`
the name of the authority in the index server to which the search is to be limited. The default is `all` authorities. The `authority` argument may be repeated, in which case the the search is carried out in each authority (effectively or'ing the authority arguments).
`added-after` limits the list to those handles for documents that were added or modified since *time*, a universal time expressed in ISO 8601 format. If the server is not able to determine date of modification to the resolution of a day, or if the server is not able to selectively extract records on a time scale of a day, the server may return additional records, e.g. all those modified during the week, month, or even century containing the date.

- **Rules for bibliographic field matching**: Words or phrases are matched exactly as submitted; the wild card symbol `*` should be used to indicate truncation.

MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Specifies a search request over the entire collection. Keyword Arguments are a set of bibliographic fields and values specifying the search criteria. Returns a structured response that contains optional search statistics and a list of documents that matches the search criteria.

Each element of the document list has the following contents:

- The `handle` of the document.
- The `ranking score` of the document for the specific query, if available.
- The `free-text title` of the document, if available
- The `author(s)` of the document, if available. Multiple authors are delimited by a colon (:) character. Each author is in `last, first middle` format.
- The `date of publication` of the document in ISO 8601 format. The exact meaning of the term "date of publication" is determined by the administrator of the index service. It is included in the record that is returned so that user interface services may use the date for ordering of a result set.

Note that clients should not assume any logical ordering of the records returned.

The statistics section contains information about which partitions of the collection returned hits along with information about errors that occurred while processing the search request. Collections are currently segmented by institutional publishing authorities. Authorities returning hits are grouped into buckets according to the number of hits returned.

The statistics error section is organized according to the error received from the remote server since most servers generate similar error messages. Authorites with the same error are included under the same error element.

Example Request:

```
Protocol=Dienst;Service=QM;ver=2.0;verb=SearchBoolean;field1=author;value1=davis;op1=or;field2=
```

Example Request:

```
Protocol=Dienst;Service=QM;ver=2.0;verb=SearchBoolean;field1=author;value1=donald;op1=and;field
```

Example Request:

```
Protocol=Dienst;Service=QM;ver=2.0;verb=SearchBoolean;field1=author;value1=donald;op1=or;field2
```

Note: Note the use of the `special character` "+" to indicate spaces between word tokens in the

arguments.

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <SearchBoolean version="2.0">
    <statistics grouping="hits" segmentation="authority" count="2">
      <hits count="2" authorities="1">
        <authority name="handlecorp">
      </hits>
      <errors count="1">
        <error text="Can't connect to errorcorp:9999" authorities="1">
          <authority name="errorcorp" />
        </error>
      </errors>
    </statistics>
    <records>
      <record>
         <handle>handlecorp/1112</handle>
         <rank>2857</rank>
         <author>Doe, J.</author>
         <author>Public, J. Q.</author>
         <title>Answering Queries</title>
         <date>19941106</date>
      </record>
      <record>
         <handle>handlecorp/1110</handle>
         <rank>2859</rank>
         <author>Mouse, M.</author>
         <author>Disney, W.</author>
         <title>Success with Cartoons</title>
         <date>19941106</date>
      </record>
    </records>
  </SearchBoolean>
```

Collection Service

The collection service provides information about the individual components that make up a collection. For more information, please consult http://www.cs.cornell.edu/cdlrg/dienst/protocols/DienstProtocol.htm#4.4%20Collection%20Service.

Verb function: Describe Verb
Verb name: Describe-Verb
Verb version: 2.0
NSF-DFG-Math usage: req
Fixed arguments: value
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response that contains a list, where each element of the list provides information on a version of the specified verb that is supported by this service. The following information may be provided at the verb or version level.

- `description`, description of the verb or a specific version
- `note`, information pertaining to the verb or a specific version

Each element of the list contains the following information:

- `version number` of the verb.
- `arguments`, a list of the names of the `fixed` and `keyword`arguments, if any, accepted by the verb *in that version*.
- `example` template of request to this repository, with fixed argument indicated in brackets
- `returns`**,** optional, contains information about response format.

Note that a service may implement more than one version of a verb.

Example Request:

```
Protocol=Dienst;Service=Collection;ver=2.0;verb=Describe-Verb;value=Regions
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Describe-Verb version="2.0">
    <Verb name="Regions">
      <description>List the regions for this collection.
      </description>
      <version id="1.0">
        <example>http://cs-tr.cs.cornell.edu:80/Dienst/Collection/1.0/Regions</example>
      </version>
    </Verb>
  </Describe-Verb>
```

Verb function: List Verbs
Verb name: List-Verbs
Verb version: 2.0
NSF-DFG-Math usage: req
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response containing the name of the verbs defined by this service.

Example Request:

```
Protocol=Dienst;Service=Collection;ver=2.0;verb=List-Verbs
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <List-Verbs version="2.0">
    <verb>Regions</verb>
    <verb>Publishers</verb>
    <verb>QueryMediators</verb>
    <verb>Indices</verb>
    <verb>Repositories</verb>
  </List-Verbs>
```

Verb function: Regions
Verb name: Regions
Verb version: 1.0
NSF-DFG-Math usage: opt
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response containing a list of regions.

Example Request:

```
Protocol=Dienst;Service=Collection;ver=1.0;verb=Regions
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Regions version="1.0">
    <Region host="region1host.cs.yourinst.edu" port="9090"
      symbol="NA-EAST" name="North America - Eastern Region" />
    <Region host="region2host.cs.yourinst.edu" port="9898"
      symbol="NA-WEST" name="North America - Western Region" />
  </Regions>
```

Verb function: Collection
Verb name: Collection
Verb version: 3.0
NSF-DFG-Math usage: opt
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response containing information about the collection server. This request provides information about the verbs supported by the collection service.

Example Request:

```
Protocol=Dienst;Service=Collection;ver=1.0;verb=Collection
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Collection version="1.0">
    <CollectionServer host="chost.cs.yourinst.edu"
      port="9891" priority="1">
      <Verbs>
        <Publishers>
          <version>1.0</version>
          <version>2.0</version>
          <version>3.0</version>
        </Publishers>
        <Repositories>
          <version>2.0</version>
          <version>3.0</version>
          <version>3.0</version>
        </Repositories>
      </Verbs>
    </CollectionServer>
  </Collection>
```

Verb function: Publishers
Verb name: Publishers
Verb version: 3.0
NSF-DFG-Math usage: req
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response containing the list of publishers that are represented in the collection. All documents in the collection are associated with a publisher.

Example Request:

```
Protocol=Dienst;Service=Collection;ver=3.0;verb=Publishers
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Publishers version="3.0">
    <publisher pretty="Cornell University Computer Science
      Department" authority="ncstrl.cornell" publisher="CORNELLCS">
    <publisher pretty="CNR - Instituto di Cibernetica (Napoli)"
      authority="ercim.cnr.ic" publisher="ercim.cnr.ic">
  </Publshers>
```

Verb function: Query Mediators
Verb name: QueryMediators
Verb version: 2.0
NSF-DFG-Math usage: req
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response containing the name of the verbs defined by this service.

Example Request:

```
Protocol=Dienst;Service=Collection;ver=2.0;verb=QueryMediators
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <QueryMediators version="2.0">
    <QueryMediator host="qmhost.cs.yourinst.edu" port="80"
      priority="1">
      <Verbs>
        <Describe-Verb><version>2.0</version></Describe-Verb>
        <List-Verbs><version>2.0</version><List-Verbs>
        <SearchBoolean>
          <version>2.0</version><version>3.0</version>
        <SearchBoolean>
      </Verbs>
    </QueryMediator>
  </QueryMediators>
```

Verb function: Indexers
Verb name: Indices
Verb version: 4.0
NSF-DFG-Math usage: req
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response containing the list of indexers that provide search services on a portion of the collection.

Example Request:

```
Protocol=Dienst;Service=Collection;ver=4.0;verb=Indices
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Indices version="4.0">
    <Indexer host="ihost.cs.inst.edu" port="80" priority="1">
      <Authorities>
        <authority name="ercim.cnr.ic" />
        <authority name="ncstrl.cornell" />
      </Authorities>
      <Verbs>
        <Describe-Verb><version>1.0</version></Describe-Verb>
        <List-Verbs><version>1.0</version><List-Verbs>
        <SearchBoolean>
          <version>2.0</version><version>3.0</version>
        <SearchBoolean>
      </Verbs>
    </Indexer>
  </Indices>
```

Verb function: Repositories
Verb name: Repositories
Verb version: 4.0
NSF-DFG-Math usage: req
NSF-DFG-Math Note(s): List Cornell, Gottingen, and Michigan here
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response containing the list of repositories that store documents contained in the collection.

Example Request:

```
Protocol=Dienst;Service=Collection;ver=4.0;verb=Repositories
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Repositories version="4.0">
    <Repository host="rhost.cs.inst.edu" port="80" priority="1">
      <Authorities>
        <authority name="ercim.cnr.ic" />
        <authority name="ncstrl.cornell" />
      </Authorities>
      <Verbs>
        <Describe-Verb><version>1.0</version>
          <version>2.0</version></Describe-Verb>
        <List-Verbs><version>1.0</version>
          <version>2.0</version><List-Verbs>
        <Structure><version>2.0</version><version>4.0</version>
        </Structure>
        <Disseminate><version>3.0</version>
        </Disseminate>
        <Formats><version>4.0</version>
        </Formats>
        <List-Contents><version>2.0</version><version>4.0</version>
        </List-Contents>
      </Verbs>
    <Repository>
  </Repositories>
```

Info Service

These messages return general information about the server. For more information, please see http://www.cs.cornell.edu/cdlrg/dienst/protocols/DienstProtocol.htm#4.3%20Info%20Service.

Verb function: Describe Verb
Verb name: Describe-Verb
Verb version: 2.0
NSF-DFG-Math usage: opt
Fixed arguments: value
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response that contains a list, where each element of the list provides information on a version of the specified verb that is supported by this service. The following information may be provided at the verb or version level.

- `description`, description of the verb or a specific version
- `note`, information pertaining to the verb or a specific version

Each element of the list contains the following information:

- `version number` of the verb.
- `arguments`, a list of the names of the `fixed` and `keyword` arguments, if any, accepted by the verb *in that version*.
- `example` template of request to this repository, with fixed argument indicated in brackets
- `returns`, optional, contains information about response format.

Note that a service may implement more than one version of a verb.

Example Request:

```
Protocol=Dienst;Service=Info;ver=2.0;verb=Describe-Verb;value=List-Services
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Describe-Verb version="2.0">
    <Verb name="List-Services">
      <description>List services available at this site.
      </description>
      <version id="1.0">
        <example>http://cs-tr.cs.cornell.edu:80/Dienst/Info/1.0/List-Services</example>
      </version>
    </Verb>
  </Describe-Verb>
```

Verb function: List Verbs
Verb name: List-Verbs
Verb version: 2.0
NSF-DFG-Math usage: opt
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response containing the name of the verbs defined by this service.

Example Request:

```
Protocol=Dienst;Service=Info;ver=2.0;verb=List-Verbs
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <List-Verbs version="2.0">
    <verb>Describe-Verb</verb>
    <verb>List-Services</verb>
    <verb>Identity</verb>
  </List-Verbs>
```

Verb function: Server Identification
Verb name: Identity
Verb version: 1.0
NSF-DFG-Math usage: rep
NSF-DFG-Math Note(s): This Dienst verb is replaced by the OAI verb `Identify`.
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response containing information about the identity of the Dienst server.

Example Request:

```
/Dienst/Info/1.0/Identity
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Identity version="1.0">
    <server>This Server</server>
    <localhost>foo.bar.com</localhost>
    <localport>80</localport>
    <maintainer>jane@bar.com</maintainer>
    <daylight_savings_time_zone>EDT</daylight_savings_time_zone>
    <standard_time_zone>EST</standard_time_zone>
  </Identity>
```

Verb function: Server Identification
Verb name: Identify
Verb version: 1.0
NSF-DFG-Math usage: req
NSF-DFG-Math Note(s): This OAI verb is replaces by the Dienst verb `Identity`.
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses:

Verb Description:

This verb is used to retrieve information about a repository, including administrative, identity, and community-specific information. For more information, please see:

http://www.openarchives.org/OAI/openarchivesprotocol.htm#Identify.

Example Request:

```
http://an.oa.org/OAI-script?verb=Identify
```

Example Response:

Note:

The response to this protocol request includes the following elements that must be provided by every OAI compliant repository:

- **repositoryName**: a human readable name for the repository, in the example "The University of Spa E-print System";
- **baseURL**: the BASE-URL of the repository;
- **protocolVersion**: the version of the OAI protocol supported by the repository;
- **adminEmail**: the e-mail address of the administrator of the repository.

In addition, the response may contain a list of description containers, which provide an extensible mechanism for communities to describe their repositories. The description container, could -- for instance -- be used to include collection-level metadata in the response to the Identify request. Each description container must be accompanied by the URL of an XML schema, which provides the semantics of the descriptive container. The below example of a response to the Identify request contains two description containers:

- The oai-identifier container complies to an XML Schema, which is available at http://www.openarchives.org/OAI/oai-identifier.xsd. This schema, shown in Appendix 2, is used by repositories that choose to comply with a specific format of unique identifiers for records. The format of that identifier is explained by means of comments in the oai-identifier.xsd XML Schema.
- The eprints container complies to an XML Schema, which is available at http://www.openarchives.org/OAI/eprints.xsd. This schema, shown in Appendix 2, has been agreed upon by the OAI e-print community, and contains information specific to repositories in that community.

```
<?xml version="1.0" encoding="UTF-8"?>
  <Identify      xmlns="http://www.openarchives.org/OAI/1.0/OAI_Identify"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
     xsi:schemaLocation="http://www.openarchives.org/OAI/1.0/OAI_Identify
                  http://www.openarchives.org/OAI/1.0/OAI_Identify.xsd">
    <responseDate>2000-10-01T19:20:30-04:00</responseDate>
    <requestURL>http://an.oa.org/OAI-script?verb=Identify</requestURL>
    <repositoryName>The University of Spa E-print System</repositoryName>
    <baseURL>http://an.oa.org/OAI-script</baseURL>
    <protocolVersion>1.0</protocolVersion>
    <adminEmail>mailto:adm@spa.ac.be</adminEmail>
    <description>
```

```
      <oai-identifier xmlns="http://www.openarchives.org/OAI/oai-identifier"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
            xsi:schemaLocation="http://www.openarchives.org/OAI/oai-identifier
                           http://www.openarchives.org/OAI/oai-identifier.xsd">
        <scheme>oai</scheme>
        <repositoryIdentifier>bespa</repositoryIdentifier>
        <delimiter>:</delimiter>
        <sampleIdentifier>oai:bespa:medi99-123</sampleIdentifier>
      </oai-identifier>
    </description>
    <description>
      <eprints xmlns="http://www.openarchives.org/OAI/eprints"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
            xsi:schemaLocation="http://www.openarchives.org/OAI/eprints
                                http://www.openarchives.org/OAI/eprints.xsd">
        <content>
          <URL>http://an.oa.org/info/content.htm</URL>
        </content>
        <metadataPolicy>
          <text>Metadata can be used by commercial and non-commercial service
providers</text>
          <URL>http://an.oa.org/info/metadata_use.htm</URL>
        </metadataPolicy>
        <dataPolicy>
          <text>Full content, i.e. preprints may not be harvested by robots</text>
        </dataPolicy>
        <submissionPolicy>
          <URL>http://an.oa.org/info/submission.htm</URL>
        </submissionPolicy>
      </eprints>
    </description>
  </Identify>
```

Verb function: List Services
Verb name: List-Services
Verb version: 1.0
NSF-DFG-Math usage: opt
Fixed arguments: none
Keyword arguments: none
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Returns a structured response containing the names of the defined Dienst services that are supported by this server.

Example Request:

```
Protocol=Dienst;Service=Info;ver=1.0;verb=List-Services
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <List-Services version="1.0">
    <service>Repository</service>
    <service>Index</service>
  </List-Services>
```

This describes how SearchBoolean is implemented at Cornell as of today (30 April 2002).

Verb function: Submit a Query
Verb name: SearchBoolean
Verb version: 6.0
NSF-DFG-Math usage: mod
Fixed arguments: none
Keyword arguments: see below
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Specifies a search request to the index server. Keyword Arguments are a set of bibliographic fields and values specifying the search criteria. Returns a structured response that is a list of each document that matches the search criteria. Each element of the list has the following contents:

- The `handle` of the document.
- The `relevance information` of the document for the specific query, if available.
- The `free-text title` of the document, if available
- The `author(s)` of the document, if available. Each author is in `last, first middle` format.
- The `date of publication` of the document in ISO 8601 format. The exact meaning of the term "date of publication" is determined by the administrator of the index service. It is included in the record that is returned so that user interface services may use the date for ordering of a result set.

Note that clients should not assume any logical ordering of the records returned.

Bibliographic Field Arguments:

Valid bibliographic field arguments can vary among index servers. The `Describe-Verb` verb returns the set of field arguments for a particular server. The minimal and standard set of bibliographic field arguments is as follows:

- **Parameters** (available in enumerated form, as in field1, field2, etc.)
  - field
  - value
  - boolean [NSF/DFG = operator]
- **Fields searched**
  - title (req)
  - author (req)
  - abstract (opt)
  - subject (opt)
  - keywords
  - anywhere [NSF/DFG = fullbib (req); does not include full-text] Fields not implemented:
  - pubtype (req, indicates whether monograph or serial)
  - language (req)
  - pubdate (req)
  - publisher (req)
  - itemno (opt)
- **Operators**
  - and (req)
  - or (req) Operators not implemented:

- not (opt)
- within (opt)
- including (opt)

Scope Argument:

Two additional fields are available:

`authority`
: the name of the authority in the index server to which the search is to be limited. The default is `all` authorities. The `authority` argument may be repeated, in which case the the search is carried out in each authority (effectively or'ing the authority arguments).

`added-after`
: limits the list to those handles for documents that were added or modified since *time*, a universal time expressed in ISO 8601 format. If the server is not able to determine date of modification to the resolution of a day, or if the server is not able to selectively extract records on a time scale of a day, the server may return additional records, e.g. all those modified during the week, month, or even century containing the date.

Rules for bibliographic field matching: A token in a field is either an unquoted word or a quoted string. Tokens are matched to bibliographic entries according to the following rules:

- Words or phrases are matched exactly as submitted; the wild card symbol `*` should be used to indicate truncation.

Examples

Example Request:

```
Protocol=Dienst&Service=Index&ver=6.0&verb=SearchBoolean&field1=author&value1=davis&op1=or&fiel
```

Example Request:

```
Protocol=Dienst&Service=Index&ver=6.0&verb=SearchBoolean&field1=author&value1=donald&op1=and&fi
```

Example Request:

```
Protocol=Dienst&Service=Index&ver=6.0&verb=SearchBoolean&field1=author&value1=donald&op1=or&fie
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <SearchBoolean version="6.0">
    <record>
       <handle>euclid.mathbk/1112</handle>
       <rank>2857</rank>
       <author>Doe, J.</author>
       <author>Public, J. Q.</author>
       <title>Math Book One</title>
       <date>1888</date>
    </record>
    <record>
       <handle>euclid.mathbk/1110</handle>
```

```
      <rank>2859</rank>
      <author>Mouse, M.</author>
      <author>Disney, W.</author>
      <title>Math Book Two</title>
      <date>1879</date>
    </record>
  </SearchBoolean>
```

HOME

Search verb

Search verb notes from May 23-24 meeting.

Verb function: Submit a Query
Verb name: Search
Verb version: 1.0
NSF-DFG-Math usage: required
Fixed arguments: none
Keyword arguments: see below
MIMETYPE response: text/xml
Status Codes responses: 200, 400

Verb Description:

Specifies a search request to the index server. Keyword Arguments are a set of bibliographic fields and values specifying the search criteria. Returns a structured response that is a list of each document that matches the search criteria.

Requests

Valid bibliographic field arguments can vary among index servers. The `Describe-Verb` verb returns the set of field arguments for a particular server. The minimal and standard set of bibliographic field arguments is as follows:

- `docstruct[n]` (opt -- default=maindocument)
  Must enumerate, beginning with 1. Legal parameter values:
  - maindocument (req)
  - page (req)
  - chapter(opt)
  - paragraph (opt)
  - table of contents (opt)
  - appendix (opt)
  - table (opt)
  - illustration (opt)
  - article (opt)
  - other (opt)
- `field[n]` (req)
  Must enumerate, beginning with 1. Legal parameter values:
  - title (req)
  - itemno (opt)
  - author (req)
  - pubtype (req, indicates whether monograph or serial))
  - subject (opt)
  - language (req)
  - fullbib (req)
  - abstract (opt)
  - pubdate (req)
  - publisher (req)
  - fulltext
- `value[n]` (req)
  Must enumerate, beginning with 1.
- `op[n]` (req if applicable)
  Must enumerate, beginning with 1. Legal parameter values:
  - and (req)

    - or (req)
    - not (opt)
    - within (opt)
    - including (opt)
- `relop[n]` (opt, requires use of op)
  Must enumerate, beginning with 1??? Legal parameter values:
    - same (req)
    - child (req)
    - parent (opt)
- `authority` (optional)
  The name of the authority in the index server to which the search is to be limited. The default is `all` authorities. The `authority` argument may be repeated, in which case the the search is carried out in each authority (effectively or'ing the authority arguments).
- `added-after` (optional)
  Limits the list to those handles for documents that were added or modified since *time*, a universal time expressed in ISO 8601 format. If the server is not able to determine date of modification to the resolution of a day, or if the server is not able to selectively extract records on a time scale of a day, the server may return additional records, e.g. all those modified during the week, month, or even century containing the date.

Operator Precedence: queries are assumed to be submitted using Reverse Polish Notation (RPN)

Rules for bibliographic field matching: A token in a field is either an unquoted word or a quoted string. Tokens are matched to bibliographic entries according to the following rules:

- Words or phrases are matched exactly as submitted; the wild card symbol `*` should be used to indicate truncation.

Responses

Note that clients should not assume any logical ordering of the records returned.

- `handle` (required)
  the doc handle
- `rank` (required)
  Relevance data in any form. Should be made as meaningful as possible.
- `author` (required if available)
  Repeatable. Each author is in `last, first middle` format.
- `title` (required)
- `date` (required)
  The `date of publication` of the document in ISO 8601 format. The exact meaning of the term "date of publication" is determined by the administrator of the index service. It is included in the record that is returned so that user interface services may use the date for ordering of a result set.

Example Request:

```
Protocol=Dienst&Service=Index&ver=6.0&verb=Search&field1=author&value1=davis&op1=or&field2=auth
```

Example Request:

```
Protocol=Dienst&Service=Index&ver=6.0&verb=Search&field1=author&value1=donald&op1=and&field2=ti
```

Example Request:

```
Protocol=Dienst&Service=Index&ver=6.0&verb=Search&field1=author&value1=donald&op1=or&field2=tit
```

Example Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <Search version="1.0">
    <record>
       <handle>handlecorp/1112</handle>
       <rank>2857</rank>
       <author>Doe, J.</author>
       <author>Public, J. Q.</author>
       <title>Answering Queries</title>
       <date>19941106</date>
    </record>
    <record>
       <handle>handlecorp/1110</handle>
       <rank>2859</rank>
       <author>Mouse, M.</author>
       <author>Disney, W.</author>
       <title>Success with Cartoons</title>
       <date>19941106</date>
    </record>
  </SearchBoolean>
```

Submit a Query

Verb name: Search
Verb version: 1.0
Required arguments: set [?], field, value, op
Optional arguments: sort, startResult, endResult, authority [?]
MIMETYPE of response: text/xml

Verb Description

Specifies a search request to an index server. Arguments indicate search criteria and the order and amount of search results to be returned. The response is a structured list of documents that match the search criteria, and includes a results summary. It is possible to request the results summary only.

Arguments

The arguments `field[n]`, `value[n]`, `op[n]`, all enumerate, beginning with 1. [if we set a max value for n, we can write a schema to validate responses.]

Supported values for bibliographic field arguments may vary among index servers. The `DescribeVerb` verb returns the set of field values supported by a particular server. The arguments and values that must be submitted and/or supported are marked "required" in the following:

- **`set`** :: [required??] these would be OAI sets. We have no mechanism to indicate multiple sets per query. See 'authority' below.
- **`sort`** :: a requested sort order for result sets. Default is ??? Possible values are:
  - `title` [required?]
  - `author`
  - `date`
  - `rank` based on whatever relevance data is provided by the repository.
- **`startResult`** :: the numerical order of the first result to be returned in the response. Value is an integer, 0 or greater; default is 1. If the value is 0, no results will be returned, regardless of the value of `endResult`.
- **`endResult`** :: the numerical order of the last result to be returned in the response. Value is an integer, 0 or greater; default is the last result available. If the value is less than `startResult`, the default value will be used [or should this be a badArgument error?].
- **`field[n]`** :: [required] Following is a list of permitted argument values. Those marked "[required]" must be supported (indexed).
  - `title` [required]
  - `itemno` [should this be 'identifier'?]
  - `author` [required]
  - `pubtype` [required] indicates whether monograph or serial
  - `subject`
  - `language` [required] [are we controlling the format of this in some way?]
  - `fullbib` [required]
  - `abstract`
  - `pubdate` [required]
  - `publisher` [required]
  - `fulltext`
- **`value[n]`** :: [required] the query string
- **`op[n]`** :: [required, if applicable] Defines the relationship between `n` and `n + 1` field/value arguments. Following is a list of permitted argument values. Those marked "[required]" must be supported.
  - `and` [required]
  - `or` [required]

    - not
    - within
    - including
- **authority** :: [I don't think we need this anymore, right?] The name of the authority in the index server to which the search is to be limited. The default is all authorities. The `authority` argument may be repeated, in which case the the search is carried out in each authority (effectively or'ing the authority arguments).

Operator Precedence: queries are assumed to be submitted using Reverse Polish Notation (RPN)

Rules for field matching: the value of the `value` argument is treated as a single string (word or phrase) and matched exactly as submitted; the wild card symbol `*` should be used to indicate truncation.

Response

Unless there is an error, all responses include a single `resultsSummary` and zero to many `record` elements.

- **<resultsSummary>** :: [required] an empty element with data about the results returned. All of the following attributes are included, using defaults if not specified in request. The value of `endResult` may be less than that requested (if there were fewer results than requested).
    - repositoryIdentifier
    - set
    - sort
    - totalResults
    - startResult
    - endResult
- **<record>** :: zero to many `record` elements allowed. Each contains the following elements:
    - **<identifier>** :: [required] the unique document identifier [do these need to be OAI identifiers? did we agree to that?]
    - **<title>** :: [required]
    - **<author>** :: [required if available] Repeatable. Each author is in `last, first middle` format.
    - **<pubdate>** :: [required] The date of publication of the document in ISO 8601 format.
    - **<rank>** :: [required] Relevance data in any form. Should be made as meaningful as possible.

Error Codes

- **badArgument** :: the request includes illegal arguments or is missing required arguments.
- **noRecordsMatch** :: no documents match the the submitted search criteria.
- **unsupportedArgumentValue** :: an argument value was submitted that is not supported by this repository. [This is not in OAI. Perhaps badArgument would suffice, but something more specific could be useful. It could also be broken up into even more specific errors, such as sortArgumentValueUnsupported, fieldArgumentValueUnsupported, etc.]
- **noSetHierarchy** :: a repository may not have sets?

Examples

Request

```
http://some.cgm.server/script?protocol=CGM&service=Index
        &verb=Search&ver=1.0&set=math&endResult=100
        &field1=author&value1=todhunter&op1=and
        &field2=title&value2=trigonometry
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
  <CGM service="Index">
    <responseDate>2002-10-01T19:20:30Z</responseDate>
    <request verb="Search" ver="1.0" set="math" endResult="100"
             field1="author" value1="todhunter" op1="and"
             field2="title" value2="trigonometry">
             http://some.cgm.server/script</request>
    <Search ver="1.0">
      <resultsSummary repositoryIdentifier="CULmath" set="math"
                      sort="title" totalResults="2" startResult="1"
                      endResult="2" />
      <record>
        <identifier>CULmath:cul.math/98765432</identifier>
        <title>Fun with Trigonometry</title>
        <author>Todhunter, I.</author>
        <author>Disney, W.</author>
        <pubdate>1888</date>
        <rank>2859</rank>
      </record>
      <record>
        <identifier>CULmath:cul.math/00640001</identifier>
        <title>Spherical Trigonometry</title>
        <author>Todhunter, I.</author>
        <pubdate>1886</date>
        <rank>2857</rank>
      </record>
    </Search>
  </CGM>
```

Request

```
http://some.cgm.server/script?protocol=CGM&service=Index
        &verb=Search&ver=1.0&set=math&startResult=0
        &field1=fulltext&value1=geometry&op1=and
        &field2=fulltext&value2=trigonometry
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
  <CGM service="Index">
    <responseDate>2002-10-02T19:20:30Z</responseDate>
    <request verb="Search" ver="1.0" set="math" startResult="0"
             field1="fulltext" value1="geometry" op1="and"
             field2="fulltext" value2="trigonometry">
             http://some.cgm.server/script</request>
    <Search ver="1.0">
      <resultsSummary repositoryIdentifier="CULmath" set="math"
                      sort="title" totalResults="1024" startResult="0"
                      endResult="0" />
    </Search>
  </CGM>
```

Request

```
http://some.cgm.server/script?protocol=CGM&service=Index
        &verb=Search&ver=1.0&set=CULmathbks
        &field1=author&value1=bush&op1=and
        &field2=fullbib&value2=basic%20counting
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
  <CGM service="Index">
    <responseDate>2002-10-02T19:20:30Z</responseDate>
    <request verb="Search" ver="1.0" set="math" field1="author"
             value1="bush" op1="and" field2="fullbib"
             value2="basic counting">http://some.cgm.server/script
             </request>
    <error code="noRecordsMatch" />
  </CGM>
```

Verb History

Substantial reworking and extension of Dienst verb SearchBoolean, version 5.0.

Submit a Query

Verb name: Search
Verb version: 1.0
Required arguments: field, value, op
Optional arguments: set, sort, startResult, resultSize
MIMETYPE of response: text/xml

Verb Description

Specifies a search request to an index server. Arguments indicate search criteria and the order and amount of search results to be returned. The response is a structured list of documents that match the search criteria, and includes a results summary. It is possible to request the results summary only.

Arguments

The arguments `field[n]` and `value[n]` enumerate, beginning with 1. For `op[n]`, see below. [if we set a max value for n (in the 3-5 range), we can write a schema to validate responses.]

Supported values for bibliographic field arguments may vary among index servers. The `DescribeVerb` verb returns the set of field values supported by a particular server. The arguments and values that must be submitted and/or supported are marked "required" in the following:

- **`set`** :: an optional grouping construct, identical in definition to OAI's use. With CGM Search, however, multiple sets can be indicated in a single query by using a delimiter (|); for example: `set=set1|set2|set3`. When searching across multiple sets, the complete query is executed within each set (effectively OR'ing the sets). When no set is indicated, the default is either 1) all sets, when a repository supports sets, or 2) all repository content, when the repository does not support sets.
- **`sort`** :: a requested sort order for result sets. Default is rank. Possible values are:
  - `rank` [required] based on whatever relevance data is provided by the repository.
  - `title`
  - `author`
  - `date`
- **`startResult`** :: the numerical order of the first result to be returned in the response. Value is an integer, 0 or greater; default is 1. If the value is set to 0, no results will be returned, regardless of the value of `resultSize`.
- **`resultSize`** :: the number of result records requested. Value is an integer, 0 or greater; default is all results. If the value is set to 0, no results will be returned, regardless of the value of `startResult`.
- **`field[n]`** :: [required] Following is a list of permitted argument values. Those marked "[required]" must be supported (indexed).
  - `title` [required]
  - `identifier`
  - `author` [required]
  - `pubtype` [required] indicates whether `monograph` or `serial`
  - `subject`
  - `language` [required] use ISO639-1 or ISO639-2
  - `fullbib` [required]
  - `abstract`
  - `pubdate` [required]
  - `publisher` [required]
  - `fulltext`
- **`value[n]`** :: [required] the query string
- **`op[n]`** :: [required] op[n] will normally operate on `n` and `n-1` field/value arguments. See the separate document on RPN Notation. Following is a list of permitted argument values. Those marked

"[required]" must be supported.
    - `and` [required]
    - `or` [required]
    - `not`
    - `within`
    - `including`

Rules for field matching: the value of the `value` argument is treated as a single string (word or phrase) and matched exactly as submitted; the wild card symbol `*` should be used to indicate truncation.

Operator Precedence: queries are assumed to be submitted using Reverse Polish Notation (RPN). See the separate document on RPN Notation

Response

Unless there is an error, all responses include a single `resultsSummary` and zero to many `record` elements.

A well-formed query that executes properly and yet has no matches is not considered an error. In this case, a `resultsSummary` would be returned, but no `record` elements.

- **`<resultsSummary>`** :: [required] an empty element with data about the results returned. All of the following attributes must be included, using defaults if not specified in request. These values describe the results actually returned; they may differ from values in the request.
    - `set` :: If no set was indicated in the response, then return either 1) a list of all sets searched, if sets are supported (`set="set1|set2|set3"`); or 2) an empty value, if sets are not supported (`set=""`)
    - `sort` :: use default when not in request.
    - `totalResults` :: the total number of matches on a query, regardless of how many or which results are returned.
    - `startResult` :: the order of the first result returned in this response. Value is 0 when no results are returned.
    - `resultSize` :: the number of results returned in this response. This may be less than the `resultSize` requested, if there are fewer results than requested.
- **`<record>`** :: zero to many `record` elements allowed. Each contains the following elements:
    - **`<identifier>`** :: [required] the unique document identifier. Its format is determined by the repository.
    - **`<title>`** :: [required]
    - **`<author>`** :: [required if available] Repeatable. Each author is in `last, first middle` format.
    - **`<pubdate>`** :: [required] The date of publication of the document in ISO 8601 format.
    - **`<rank>`** :: [required] Relevance data in any form. Should be made as meaningful as possible.

Error Codes

- **`badArgument`** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.
- **`noSetHierarchy`** :: the repository does not support sets

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Search&ver=1.0&set=math&endResult=100
        &field1=author&value1=todhunter&field2=title
        &value2=trigonometry&op2=and
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
  <CGM>
    <responseDate>2002-10-01T19:20:30Z</responseDate>
    <request verb="Search" ver="1.0" set="math" resultSize="100"
             field1="author" value1="todhunter" field2="title"
             value2="trigonometry" op2="and">
             http://some.cgm.server/script</request>
    <Search ver="1.0">
      <resultsSummary set="math" sort="title" totalResults="2"
                      startResult="1" resultSize="2" />
      <record>
        <identifier>cul.math/98765432</identifier>
        <title>Fun with Trigonometry</title>
        <author>Todhunter, I.</author>
        <author>Disney, W.</author>
        <pubdate>1888</date>
        <rank>2859</rank>
      </record>
      <record>
        <identifier>cul.math/00640001</identifier>
        <title>Spherical Trigonometry</title>
        <author>Todhunter, I.</author>
        <pubdate>1886</date>
        <rank>2857</rank>
      </record>
    </Search>
  </CGM>
```

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Search&ver=1.0&set=math&startResult=0
        &field1=fulltext&value1=geometry&field2=fulltext
        &value2=trigonometry&op2=and

http://some.cgm.server/script?protocol=CGM
        &verb=Search&ver=1.0&set=math&resultSize=0
        &field1=fulltext&value1=geometry&field2=fulltext
        &value2=trigonometry&op2=and
```

These two requests are equivalent. The response below is to the first one.

Response

```
<?xml version="1.0" encoding="UTF-8"?>
  <CGM>
    <responseDate>2002-10-02T19:20:30Z</responseDate>
    <request verb="Search" ver="1.0" set="math" startResult="0"
             field1="fulltext" value1="geometry" field2="fulltext"
             value2="trigonometry" op2="and">
             http://some.cgm.server/script</request>
    <Search ver="1.0">
      <resultsSummary set="math" sort="title" totalResults="1024"
                      startResult="0" resultSize="0" />
    </Search>
  </CGM>
```

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Search&ver=1.0&set=math
        &field1=author&value1=bush&field2=fullbib
        &value2=basic%20counting&op2=and
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="Search" ver="1.0" set="math" field1="author"
           value1="bush"field2="fullbib" value2="basic counting"
           op2="and">http://some.cgm.server/script
           </request>
  <Search ver="1.0">
    <resultsSummary set="math" sort="title" totalResults="0"
                    startResult="0" resultSize="0" />
  </Search>
</CGM>
```

Verb History

Substantial reworking and extension of Dienst verb SearchBoolean, version 5.0.

Display a Document in Local Viewer

Verb name: Display
Verb version: 1.0
Required arguments: id
Optional arguments: none
MIMETYPE of response: will depend on request and viewer mechanism

Verb Description

Request to view a document in a repository's local viewer.

Arguments

- **`id`** :: the identifier for the document requested.

Response

A repository can display a document however it chooses.

Error Codes

- **`idDoesNotExist`** :: the value of the `identifier` argument is unknown or illegal.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Display&ver=1.0&id=cul.math/00640001
```

Verb History

New to CGM.

CGM Protocol

Document overview

Stage I Critical verbs:

Stage II Verbs--Not critical, but not optional?

Optional verbs:

Stage I Verbs

Critical verbs:

- Search
- Display
- Structure
- Formats
- Disseminate

Submit a Query

Verb name: Search
Verb version: 1.0
Modification Date: 2003-03-24
Required arguments: field, value, op
Optional arguments: set, sort, startResult, resultSize
MIMETYPE of response: text/xml

Verb Description

Specifies a search request to an index server. Arguments indicate search criteria and the order and amount of search results to be returned. The response is a structured list of documents that match the search criteria, and includes a results summary. It is possible to request the results summary only.

Arguments

The arguments `field[n]` and `value[n]` enumerate, beginning with 1. For `op[n]`, see below.

Supported values for bibliographic field arguments may vary among index servers. The `DescribeVerb` verb returns the set of field values supported by a particular server. The arguments and values that must be submitted and/or supported are marked "required" in the following:

- **`set`** :: an optional grouping construct, identical in definition to OAI's use. With CGM Search, however, multiple sets can be indicated in a single query by using a delimiter (|); for example: `set=set1|set2|set3`. When searching across multiple sets, the complete query is executed within each set (effectively OR'ing the sets). When no set is indicated, the default is either 1) all sets, when a repository supports sets, or 2) all repository content, when the repository does not support sets.
- **`sort`** :: a requested sort order for result sets. Default is rank. Possible values are:
  - `rank` [required] based on whatever relevance data is provided by the repository.
  - `title`
  - `author`
  - `pubdate`
- **`startResult`** :: the numerical order of the first result to be returned in the response. Value is an integer, 0 or greater; default is 1. If the value is set to 0, no results will be returned, regardless of the value of `resultSize`.
- **`resultSize`** :: the number of result records requested. Value is an integer, 0 or greater; default is all results. If the value is set to 0, no results will be returned, regardless of the value of `startResult`.
- **`field[n]`** :: [required] Following is a list of permitted argument values. Those marked "[required]" must be supported (indexed).
  - `title` [required]
  - `identifier`
  - `author` [required]
  - `pubtype` [required] indicates whether `monograph` or `serial`
  - `subject`
  - `language` [required] use ISO639-1 or ISO639-2
  - `fullbib` [required]
  - `abstract`
  - `pubdate` [required]
  - `publisher` [required]
  - `fulltext`
- **`value[n]`** :: [required] the query string
- **`op[n]`** :: [required, if multiple field/value pairs submitted] op[n] will normally operate on `n` and `n-1` field/value arguments. See the separate document on RPN Notation. Following is a list of permitted argument values. Those marked "[required]" must be supported.
  - `and` [required]
  - `or` [required]
  - `not`
  - `within`
  - `including`

Rules for field matching: the value of the `value` argument is treated as a single string (word or phrase) and matched exactly as submitted; the wild card symbol `*` should be used to indicate truncation.

Operator Precedence: queries are assumed to be submitted using Reverse Polish Notation (RPN). See the separate document on RPN Notation

Response

Unless there is an error, all responses include a single `resultsSummary` and zero to many `record` elements.

A well-formed query that executes properly and yet has no matches is not considered an error. In this case, a `resultsSummary` would be returned, but no `record` elements.

- **<resultsSummary>** :: [required] an empty element with data about the results returned. All of the following attributes must be included, using defaults if not specified in request. These values describe the results actually returned; they may differ from values in the request.
  - `repositoryIdentifier` :: the repository identifier.
  - `set` :: if no set was indicated in the response, then return either 1) a list of all sets searched, if sets are supported (`set="set1|set2|set3"`); or 2) an empty value, if sets are not supported (`set=""`).
  - `sort` :: use default when not in request.
  - `totalResults` :: the total number of matches on a query, regardless of how many or which results are returned.
  - `startResult` :: the order of the first result returned in this response. Value is 0 when no results are returned.
  - `resultSize` :: the number of results returned in this response. This may be less than the `resultSize` requested, if there are fewer results than requested.
- **<record>** :: zero to many `record` elements allowed. Each contains the following elements:
  - **<identifier>** :: [required] the unique document identifier. Its format is determined by the repository.
  - **<title>** :: [required]
  - **<author>** :: [required if available] Repeatable. Each author is in `last, first middle` format.
  - **<pubdate>** :: [required] The date of publication of the document in ISO 8601 format.
  - **<rank>** :: [required] Relevance data in any form. Should be made as meaningful as possible.

Error Codes

- **badArgument** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.
- **noSetHierarchy** :: the repository does not support sets

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Search&ver=1.0&set=math&resultSize=100
        &field1=author&value1=todhunter&field2=title
        &value2=trigonometry&op2=and
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-01T19:20:30Z</responseDate>
  <request verb="Search" ver="1.0" set="math" resultSize="100"
           field1="author" value1="todhunter" field2="title"
           value2="trigonometry" op2="and">
           http://some.cgm.server/script</request>
  <Search ver="1.0">
    <resultsSummary repositoryIdentifier="CULmath" set="math" sort="title"
                    totalResults="2" startResult="1" resultSize="2" />
    <record>
      <identifier>cul.math/98765432</identifier>
      <title>Fun with Trigonometry</title>
      <author>Todhunter, I.</author>
      <author>Disney, W.</author>
      <pubdate>1888</date>
      <rank>2859</rank>
    </record>
    <record>
      <identifier>cul.math/00640001</identifier>
      <title>Spherical Trigonometry</title>
      <author>Todhunter, I.</author>
      <pubdate>1886</date>
```

```
      <rank>2857</rank>
    </record>
  </Search>
</CGM>
```

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Search&ver=1.0&set=math&startResult=0
        &field1=fulltext&value1=geometry&field2=fulltext
        &value2=trigonometry&op2=and

http://some.cgm.server/script?protocol=CGM
        &verb=Search&ver=1.0&set=math&resultSize=0
        &field1=fulltext&value1=geometry&field2=fulltext
        &value2=trigonometry&op2=and
```

These two requests are equivalent. The response below is to the first one.

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="Search" ver="1.0" set="math" startResult="0"
           field1="fulltext" value1="geometry" field2="fulltext"
           value2="trigonometry" op2="and">
           http://some.cgm.server/script</request>
  <Search ver="1.0">
    <resultsSummary repositoryIdentifier="CULmath" set="math" sort="title"
                    totalResults="1024" startResult="0" resultSize="0" />
  </Search>
</CGM>
```

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Search&ver=1.0&set=math
        &field1=author&value1=bush&field2=fullbib
        &value2=basic%20counting&op2=and
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="Search" ver="1.0" set="math" field1="author"
           value1="bush"field2="fullbib" value2="basic counting"
           op2="and">http://some.cgm.server/script
           </request>
  <Search ver="1.0">
    <resultsSummary repositoryIdentifier="CULmath" set="math" sort="title"
                    totalResults="0" startResult="0" resultSize="0" />
  </Search>
</CGM>
```

Verb History

Substantial reworking and extension of Dienst verb SearchBoolean, version 5.0.

Display a Document in Local Viewer

Verb name: Display

Verb version: 1.0
Modification Date: 2003-01-06
Required arguments: identifier
Optional arguments: none
MIMETYPE of response: will depend on request and viewer mechanism

Verb Description

Request to view a document in a repository's local viewer.

Arguments

- **`identifier`** :: the identifier for the document requested.

Response

A repository can display a document however it chooses.

Error Codes

- **`idDoesNotExist`** :: the value of the `identifier` argument is unknown or illegal.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Display&ver=1.0&identifier=cul.math/00640001
```

Verb History

New to CGM.

Get Document Structure

Verb name: Structure
Verb version: 1.0 beta
Modification Date: 2003-03-03
Required arguments: identifier
Optional arguments: version, view
MIMETYPE of response: text/xml

Verb Description

This verb returns a structured response describing a structural view available for a document. A client may use this structural information as the basis for document requests using the `Disseminate` verb.

Arguments

- **`identifier`** :: [required] the document identifier
- **`version`** :: specifies the document version for which the structure information is requested. If

omitted, the response provides information about the latest document version.
- **`view`** :: specifies the the view of the document for which structure information is requested. If omitted, the response provides information about the default view. Specific views are requested using the view id value, obtained using the `ListViews` verb.

Response

Each response will include one `view` element. There will always be one and only one default view of a document.

- **`<view>`** :: contains one high-level, or root, `div` element, within which nested `div` hierarchies are allowed. `view` element attributes are:
  - `id` :: [required] an id value for this view. This id is defined as an XML ID.
  - `label` :: [required] a displayable label identifying this view. These are applied by the local repository and should help end users choose an appropriate document view.
  - `default` :: [required for default view] equals `1` for the sole default view; equals `0` for all other views. If absent, `default="0"` is assumed.
- **`<div>`** :: may contain one or more `div` elements. `div` element attributes are:
  - `id` :: [required] an ID value for this division, usable as is in Disseminate and other requests. This is not defined as an XML ID.
  - `type` :: if any of the following `div` types are present, they must be identified as such with the type attribute [open issue--coordinating this with possible docstruct values used in enhance Search and ListDocstruct]:
    - `maindocument`
    - `front`
    - `body`
    - `back`
    - `chapter`
    - `section`
    - `page`
  - `order` :: [required] the order of this `div` among its siblings. Values are positive integers, beginning with 1.
  - `label` :: a displayable label for this division. These are applied by the local repository and should make sense in a navigational context.

Error Codes

- **`idDoesNotExist`** :: the value of the `identifier` argument is unknown or illegal.
- **`badArgument`** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Structure&ver=1.0&identifier=cul.math/00640001
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="Structure" ver="1.0" identifier="cul.math/00640001">
           http://some.cgm.server/script</request>
  <Structure ver="1.0">
    <identifier value="cul.math/00640001"/>
```

```
    <view id="v123-a" label="Page Listing" default="1">
      <div id="a123-1" type="maindocument" order="1" label="Entire Monograph">
        <div id="a123-1" type="page" order="1" label="Page NA">
        <div id="a123-2" type="page" order="2" label="Page i">
        <div id="a123-3" type="page" order="3" label="Page ii">
        <div id="a123-4" type="page" order="4" label="Page 1">
        <div id="a123-5" type="page" order="5" label="Page 2">
        <div id="a123-6" type="page" order="6" label="Page 3">
        <div id="a123-7" type="page" order="7" label="Page 4">
        <div id="a123-8" type="page" order="8" label="Page 5">
        <div id="a123-9" type="page" order="9" label="Page 6">
      </div>
    </view>
  </Structure>
</CGM>
```

The response above describes the physical structure of a document. Because no view was specified in the request, the default view was returned.

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Structure&ver=1.0&identifier=cul.math/00640001&view=v123-b
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="Structure" ver="1.0" identifier="cul.math/00640001"
           view="v123-b">http://some.cgm.server/script</request>
  <Structure ver="1.0">
    <identifier value="cul.math/00640001"/>
    <view id="v123-b" label="Chapter Listing" default="0">
      <div id="a123" type="maindocument" order="1" label="Entire Monograph">
        <div id="a123-1" type="front" order="1" label="Front Matter">
          <div id="a123-1-1" type="page" order="1" label="Page i">
          <div id="a123-1-2" type="page" order="2" label="Page ii">
          <div id="a123-1-3" type="page" order="3" label="Page iii">
        </div>
        <div id="a123-2" type="chapter" order="2" label="Chapter I">
          <div id="a123-2-1" type="page" order="1" label="Page 1">
          <div id="a123-2-2" type="page" order="2" label="Page 2">
          <div id="a123-2-3" type="page" order="3" label="Page 3">
          <div id="a123-2-4" type="page" order="4" label="Page 4">
          ....
          <div id="a123-2-36" type="page" order="36" label="Page 36">
        </div>
        <div id="a123-3" type="chapter" order="3" label="Chapter II">
          <div id="a123-3-37" type="page" order="1" label="Page 37">
          <div id="a123-3-38" type="page" order="2" label="Page 38">
          <div id="a123-3-39" type="page" order="3" label="Page 39">
          <div id="a123-3-40" type="page" order="4" label="Page 40">
          ....
          <div id="a123-3-77" type="page" order="41" label="Page 75">
        </div>
        ...
      </div>
    </view>
  </Structure>
</CGM>
```

This response describes a non-default view of the identified document. The document view is structured into a front section and following chapters, with individual page images available within these structures. ("..." indicates where additional pages and chapters have been omitted in the example for brevity).

Verb History

Modification of Dienst verb Structure, version 2.0. CGM has removed all format related functions from the Dienst Structure.

Earlier NSF-DFG note: For `Structure`, we discussed replacing the unnumbered DIVs, which can nest, with numbered DIVs (e.g., DIV1, DIV2, etc.). Further, `Structure` should use real ID values (i.e., only NAME characters, beginning with an alpha character), and need to have a sense of relationship between layers (e.g., what shows first, GIF or PDF?) and a repository's desire to suppress or hold back a layer (e.g., don't show the OCR without this caveat, or don't show it until you've shown other layers, and then only with this caveat).

Get Formats

Verb name: Formats
Verb version: 1.0 beta
Modification Date: 2003-03-18
Required arguments: identifier
Optional arguments: version, div, view
MIMETYPE of response: text/xml

Verb Description

Returns a structured response indicating the formats available for the various structural components of a document.

Arguments

- **`identifier`** :: [required] the document identifier
- **`version`** :: specifies the document version for which the format information is requested. If omitted, it provides information about the latest document version available (see ListVersions).
- **`div`** :: specifies the id value of a div component provided in the Structure response. Use to request format information for a specific structual component. [Is there a default then? Highest level div of document?]
- **`view`** :: specifies the the view of the document for which format information is requested. If omitted, the response provides information about the default view. Specific views are requested using the view id value, obtained using the `ListViews` verb. [Some uncertainty how view works with Formats, and whether it's necessary/desirable. Currently not in response, but should be if we keep it.]

Response

Each response includes a single high-level `divReq` element.

- **`<divReq>`** :: The `div` for which format information has been requested becomes the `divReq` or root div element in the response. The `divReq` element has the same attributes as a Structure response `div` [except order?]:
  - `id` :: [required] an ID value for this division, usable as is in Disseminate and other requests. This is not defined as an XML ID.
  - `type` :: if any of the following `div` types are present, they must be identified as such with the type attribute [open issue--coordinating this with possible docstruct values used in enhance Search and ListDocstruct]:
    - `maindocument`
    - `front`
    - `body`
    - `back`
    - `chapter`
    - `section`
    - `page`

    - `order` :: [required] the order of this `div` among its siblings. Values are positive integers, beginning with 1. [this doesn't make any sense here, since there's always only one divReq. Do away with order here?]
    - `label` :: a displayable label for this division. These are applied by the local repository and should make sense in a navigational context.

  The `divReq` element may contain (and must contain at least one of) the following:
    - zero or more `format` elements for the complete `divReq`;
    - zero or more `formatSeq` elements;
- **`<format>`** :: an empty element with attributes describing a single format available for its parent element, which may be `divReq`, `div`, or `formatDiv`. The `format` element has four attributes:
    - `type` :: [required] from a controlled vocabulary of format types.
    - `mime` :: [required] the mime-type of the format.
    - `size` :: [optional?] in bytes [or KB?].
    - `label` :: [required] a displayable label identifying this format for use in a user interface.
- **`<formatSeq>`** :: an element that contains one or more `div` or `formatDiv` elements. Each `formatSeq` gives a format representation of the entire `divReq` [so, formatSeq can occur only directly under divReq? Not within some nested hierarchy?]. The `formatSeq` element has one attribute:
    - `label` :: a displayable label identifying this group of formats, for use in a user interface. [ListBinders and ListEncoding may argue for an ID value on formatSeq??]
- **`<div>`** :: a structural division within `formatSeq` that contains one or more `format` elements. If `div` is used, as opposed to `formatDiv`, it is assumed that the same structural division is also represented in Structure verb responses. `div` has four attibutes, corresponding exactly to `div` attributes in a Structure Verb response:
    - `id` :: [required] an ID value for this division, usable as is in Disseminate and other requests. This is not defined as an XML ID.
    - `type` :: if any of the following `div` types are present, they must be identified as such with the type attibute:
        - `maindocument`
        - `front`
        - `body`
        - `back`
        - `chapter`
        - `section`
        - `page`
    - `order` :: [required] the order of this `div` among its siblings. Values are positive integers, beginning with 1.
    - `label` :: a displayable label for this division. These are applied by the local repository and should make sense in a navigational context.
- **`<formatDiv>`** :: a division within `formatSeq` that contains one or more `format` elements. If `formatDiv` is used, as opposed to `div`, it is assumed that this division is not structural--not represented in Structure verb responses--but rather indicates a division peculiar to a Formats verb response. This element has four attibutes:
    - `id` :: [required] an ID value for this division, usable as is in Disseminate and other requests. This is not defined as an XML ID.
    - `type` :: [makes no sense to use the div types above here. What would make this useful? Do we need it at all?]
    - `order` :: [required] the order of this `div` among its siblings. Values are positive integers, beginning with 1.
    - `label` :: a displayable label for this division. These are applied by the local repository and should make sense in a navigational context.

Error Codes

- **`idDoesNotExist`** :: the value of the `identifier` argument is unknown or illegal.
- **`badArgument`** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Formats&ver=1.0&identifier=cul.math/00640001
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="Structure" ver="1.0" identifier="cul.math/00640001">
           http://some.cgm.server/script</request>
  <Formats ver="1.0">
    <identifier value="cul.math/00640001"/>
    <divReq id="a123-a" type="chapter" order="1" label="chapter 1">
      <format type="PDF.600" mime="application/pdf" size="5500000"
              label="Entire Document in PDF"/>
      <format type="OCR" mime="text/plain" size="412000" label="Un-proofed
              OCR text"/>
      <formatSeq label="Sequence of pages">
        <div id="a123-1" type="page" order="1" label="i">
          <format type="GIF" mime="image/gif" size="63488" label="Page
                  image"/>
          <format type="PDF.600" mime="application/pdf" size="54272"
                  label="Page in PDF"/>
          <format type="OCR" mime="text/plain" size="4096" label="Un-proofed
                  OCR text"/>
        </div>
        <div id="a123-2" type="page" order="2" label="ii">
          <format type="GIF" mime="image/gif" size="63488" label="Page
                  image"/>
          <format type="PDF.600" mime="application/pdf" size="54272"
                  label="Page in PDF"/>
          <format type="OCR" mime="text/plain" size="4118" label="Un-proofed
                  OCR text"/>
        </div>
        <div id="a123-3" type="page" order="3" label="1">
          <format type="GIF" mime="image/gif" size="63488" label="Page
                  image"/>
          <format type="PDF.600" mime="application/pdf" size="54272"
                  label="Page in PDF"/>
          <format type="OCR" mime="text/plain" size="4616" label="Un-proofed
                  OCR text"/>
        </div>
        ...
        <div id="a123-103" type="page" order="103" label="100">
          <format type="GIF" mime="image/gif" size="63488" label="Page
                  image"/>
          <format type="PDF.600" mime="application/pdf" size="54272"
                  label="Page in PDF"/>
          <format type="OCR" mime="text/plain" size="4580" label="Un-proofed
                  OCR text"/>
        </div>
      </formatSeq>
      <formatSeq label="PDF Chunks" >
        <formatDiv id="chunk1" type="chunk" order="1" label="Pages 1-30">
          <format type="PDF.600" mime="application/pdf" size="1620000"
                  label="Multi-page PDF"/>
          <format type="OCR" mime="text/plain" size="110000" label="Un-proofed
                  OCR text"/>
        </formatDiv>
        <formatDiv id="chunk2" type="chunk" order="2" label="Pages 31-60">
          <format type="PDF.600" mime="application/pdf" size="1650000"
                  label="Multi-page PDF"/>
          <format type="OCR" mime="text/plain" size="120000" label="Un-proofed
                  OCR text"/>
        </formatDiv>
        <formatDiv id="chunk3" type="chunk" order="3" label="Pages 61-90">
          <format type="PDF.600" mime="application/pdf" size="1650000"
                  label="Multi-page PDF"/>
          <format type="OCR" mime="text/plain" size="120000" label="Un-proofed
                  OCR text"/>
        </formatDiv>
```

```
        <formatDiv id="chunk4" type="chunk" order="4" label="Pages 91-103">
          <format type="PDF.600" mime="application/pdf" size="550000"
                  label="Multi-page PDF"/>
          <format type="OCR" mime="text/plain" size="40000" label="Un-proofed
                  OCR text"/>
        </formatDiv>
      </formatSeq>
    </divReq>
  </Formats>
</CGM>
```

Verb History

Modification of Dienst verb Formats, version 4.0.

Get Formats - original take

Verb name: Formats
Verb version: 1.0 alpha
Modification Date: 2003-03-03
Required arguments: identifier
Optional arguments: version, div, view
MIMETYPE of response: text/xml

Verb Description

Returns a structured response indicating the formats available for the various structural components of a document.

Arguments

- **`identifier`** :: [required] the document identifier
- **`version`** :: specifies the document version for which the format information is requested. If omitted, it provides information about the latest document version available (see ListVersions).
- **`div`** :: specifies the id value of a div component provided in the Structure response. Use to request format information for a specific structual component. [Is there a default then? Highest level div of document?]
- **`view`** :: specifies the the view of the document for which format information is requested. If omitted, the response provides information about the default view. Specific views are requested using the view id value, obtained using the `ListViews` verb. [Some uncertainty how view works with Formats, and whether it's necessary/desirable. Currently not in response, but should be if we keep it.]

Response

Each response includes a single high-level `divReq` element.

- **`<divReq>`** :: The `div` for which format information has been requested becomes the `divReq` or root div element in the response. The `divReq` element has the same attributes as a Structure response `div` [except order? The point is, this div is the same as the Structure response divs. The divs below are not really Structure divs.]:
  - `id` :: [required] an ID value for this division, usable as is in Disseminate and other requests. This is not defined as an XML ID.
  - `type` :: if any of the following `div` types are present, they must be identified as such with the type attribute [open issue--coordinating this with possible docstruct values used in enhance Search and ListDocstruct]:
    - `maindocument`

        - front
        - body
        - back
        - chapter
        - section
        - page
    - order :: [required] the order of this div among its siblings. Values are positive integers, beginning with 1. [this doesn't make any sense here, since there's always only one divReq]
    - label :: a displayable label for this division. These are applied by the local repository and should make sense in a navigational context.

  The divReq element may contain (and must contain at least one of) the following:
    - zero or more format elements for the complete divReq;
    - zero or more formatSeq elements;
- **<format>** :: an empty element with attributes describing a single format available for its parent divReq or div element. The format element has four attributes:
    - type :: [required] from a controlled vocabulary of format types.
    - mime :: [required] the mime-type of the format.
    - size :: [optional?] in bytes [or KB?].
    - label :: [required] a displayable label identifying this format for use in a user interface.
- **<formatSeq>** :: an element that contains one or more div elements. Each formatSeq gives a format representation of the entire divReq. The formatSeq element has one attribute:
    - label :: a displayable label identifying this group of formats, for use in a user interface..
- ** ** :: an element within formatSeq that contains one or more format elements. This element has four attibutes, corresponding to div attributes in a Structure Verb response [these divs *seem* like the divs from Structure response, but they won't be in all cases. So, should we call them something else, like formatDiv? Larger question, how do we connect Structure response info to Formats response info, if the divs in one don't match up with the divs in the other???]:
    - id :: [required] an ID value for this division, usable as is in Disseminate and other requests. This is not defined as an XML ID.
    - type :: if any of the following div types are present, they must be identified as such with the type attibute [but these aren't the same types as in Structure, so this doesn't make sense. Compare type=chunk in example]:
        - maindocument
        - front
        - body
        - back
        - chapter
        - section
        - page
    - order :: [required] the order of this div among its siblings. Values are positive integers, beginning with 1.
    - label :: a displayable label for this division. These are applied by the local repository and should make sense in a navigational context.

Error Codes

- **idDoesNotExist** :: the value of the identifier argument is unknown or illegal.
- **badArgument** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Formats&ver=1.0&identifier=cul.math/00640001
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
  <CGM>
    <responseDate>2002-10-02T19:20:30Z</responseDate>
    <request verb="Structure" ver="1.0" identifier="cul.math/00640001">
             http://some.cgm.server/script</request>
    <Formats ver="1.0">
      <identifier value="cul.math/00640001"/>
      <divReq id="a123-a" type="chapter" order="1" label="chapter 1">
        <format type="PDF.600" mime="application/pdf" size="5500000"
label="Entire Document in PDF"/>
        <format type="OCR" mime="text/plain" size="412000" label="Un-proofed OCR
text"/>
        <formatSeq label="Sequence of pages">
          <div id="i1" type="page" order="1" label="i">
            <format type="GIF" mime="image/gif" size="63488" label="Page image"/>
            <format type="PDF.600" mime="application/pdf" size="54272"
label="Page in PDF"/>
            <format type="OCR" mime="text/plain" size="4096" label="Un-proofed
OCR text"/>
          </div>
          <div id="i2" type="page" order="2" label="ii">
            <format type="GIF" mime="image/gif" size="63488" label="Page image"/>
            <format type="PDF.600" mime="application/pdf" size="54272"
label="Page in PDF"/>
            <format type="OCR" mime="text/plain" size="4118" label="Un-proofed
OCR text"/>
          </div>
          <div id="i3" type="page" order="3" label="1">
            <format type="GIF" mime="image/gif" size="63488" label="Page image"/>
            <format type="PDF.600" mime="application/pdf" size="54272"
label="Page in PDF"/>
            <format type="OCR" mime="text/plain" size="4616" label="Un-proofed
OCR text"/>
          </div>
          ...
          <div id="i103" type="page" order="103" label="100">
            <format type="GIF" mime="image/gif" size="" label="Page image"/>
            <format type="PDF.600" mime="application/pdf" size="54272"
label="Page in PDF"/>
            <format type="OCR" mime="text/plain" size="4580" label="Un-proofed
OCR text"/>
          </div>
        </formatSeq>
        <formatSeq label="PDF Chunks" >
          <div id="chunk1" type="chunk" order="1" label="Pages 1-30">
            <format type="PDF.600" mime="application/pdf" size="1620000"
label="Multi-page PDF"/>
            <format type="OCR" mime="text/plain" size="110000" label="Un-proofed
OCR text"/>
          </div>
          <div id="chunk2" type="chunk" order="2" label="Pages 31-60">
            <format type="PDF.600" mime="application/pdf" size="1650000"
label="Multi-page PDF"/>
            <format type="OCR" mime="text/plain" size="120000" label="Un-proofed
OCR text"/>
          </div>
          <div id="chunk3" type="chunk" order="3" label="Pages 61-90">
            <format type="PDF.600" mime="application/pdf" size="1650000"
label="Multi-page PDF"/>
            <format type="OCR" mime="text/plain" size="120000" label="Un-proofed
OCR text"/>
          </div>
          <div id="chunk4" type="chunk" order="4" label="Pages 91-103">
            <format type="PDF.600" mime="application/pdf" size="550000"
label="Multi-page PDF"/>
            <format type="OCR" mime="text/plain" size="40000" label="Un-proofed
OCR text"/>
          </div>
        </formatSeq>
      </divReq>
    </Formats>
  </CGM>
```

Verb History

Modification of Dienst verb Formats, version 4.0.

Disseminate Content

Verb name: Disseminate
Verb version: 1.0 beta
Modification Date: 2003-03-03
Required arguments: identifier, format-type
Optional arguments: version, div, view, binder?, encoding?
MIMETYPE of response: Dependent on dissemination requested.

Verb Description

Request a dissemination of a digital object. The characteristics of the dissemination that can be requested (the arguments to the `Disseminate` verb), are determined by the responses to the `Structure`, `Formats`, `ListViews`, and `ListVersions` verbs for the specific document. The response is a MIME-typed byte stream.

Arguments

- **`identifier`** :: [required] the document identifier
- **`version`** :: specifies the document version for which the format information is requested. If omitted, it provides information about the latest document version available (see ListVersions).
- **`div`** :: specifies the id value of a div component provided in the Formats response [or Structure???]. Use to request dissemination of a specific structural component of the specified document. [Is there a default then? Highest level div of document?]
- **`view`** :: specifies the the view of the document for which a dissemination request is made. If omitted, ???. Specific views are requested using the view id value, obtained using the `ListViews` verb. [As with Formats, it's unclear how view works here, and whether it's necessary/desirable. Doesn't really make sense as is. That is, if there's a div requested, then it may conflict with view. I'd like not to use it unless there's some compelling reason to.]
- **`format-type`** :: [required] specifies the MIME-type of the content in the dissemination. The list of available formats for a document is indicated by the response to the `Formats` request. Note that the value supplied for the argument is not a mime-type, but the format "type" indicated in the response to the `Formats` request.
- **`binder`** :: [included for now; do we keep this?] specifies the encapsulating binder for the dissemination. The MIME type of the stream disseminated to the requesting client is then the output of the binder application. A `binder` argument is *required* if the dissemination contains multiple logical objects (e.g., multiple page images in `image/tiff`). The list of Binders supported by a repository is available through the ListBinders protocol request.
- **`encoding`** :: [included for now; do we keep this?] specifies the compression encoding scheme that should be applied to the dissemination. The `encoding` applied is reflected in the HTTP `Content-encoding` header returned from the `Disseminate` request. The list of Encodings supported by a repository is available through the ListEncodings protocol request.

Response

The response to Disseminate is the requested content, directly. [do we consider wrapping content in xml, with something like base64 encoding?]

Error Codes

`idDoesNotExist` :: the value of the `identifier` argument is unknown or illegal.
- **`badArgument`** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Disseminate&ver=1.0&identifier=cul.math/00640001
        &div=chunk2&format-type=PDF.600
```

Response

The response returns a PDF file associated with a division identified as "chunk2", part of the specified document.

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Disseminate&ver=1.0&identifier=cul.math/00640001
        &div=i2&format-type=GIF
```

Response

Returns a GIF image of this division. Note that we are depending on divs from the Formats verb response, and not the Structure response. Is this what we want??

Verb History

Modification of Dienst Disseminate verb, version 1.0. Our position on the current CGM verb is that the record-oriented components (metadata) are to be replaced by OAI verb `GetRecord`, and that Disseminate needs to continue for handling the actual content.

Removed from Dienst verb description: "The MIME type of the byte stream is either the MIME type of the specified `content-type` or, if a `binder` is specified, the MIME type of that `binder`."

Removed from above, this was a definition of argument div (from jpw?): " narrows the dissemination request to a specific structural component (e.g., a *chapter*) of the specified view of the document instance. The supplied argument must be one the available `divs` of the view (indicated by the output of the `Structure` request) paired with a value that is an identifier of one of those `divs` (again indicated by the output of the `Structure` request). For example, if a `Structure` request indicates that there is a `div` of type `chapter` with identifier `1`, then this argument could by `chapter=1`. Note that the `div` argument can not be used in combination with the `pageimage` argument."

Also removed from argument section: "<viewID>,where `<viewID>` is one of the available views for the document instance. The resulting dissemination, if there is no `pageimage` argument, is then the specified view of the document instance."

List Views Available

Verb name: ListViews
Verb version: 1.0 beta
Modification Date: 2003-02-17

Required arguments: identifier
Optional arguments: none
MIMETYPE of response: text/xml

Verb Description

Returns a structured list of the possible views available for a single document.

Arguments

- **`identifier`** :: [required] an identifier of the document for which a list of views is requested.

Response

One view, but only one, must be identified as the default view.

- **`<view>`** :: an empty element, one per view, with three attributes:
  - `id` :: [required] an id of the view (XML ID).
  - `label` :: [required] a displayable, short label describing this view. These are applied by the local repository and should help end users choose an appropriate document view.
  - `default` :: [required for default view] possible values are `1` or `0`. If not included, `default="0"` is assumed.

Error Codes

- **`idDoesNotExist`** :: the value of the `identifier` argument is unknown or illegal.
- **`badArgument`** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=ListViews&ver=1.0&identifier=cul.math/00640001
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="ListViews" ver="1.0" identifier="cul.math/00640001">
           http://some.cgm.server/script</request>
  <ListViews ver="1.0">
    <identifier value="cul.math/00640001">
    <view id="v123-a" label="Page listing" default="1" />
    <view id="v123-b" label="Chapter and Sections" />
  </ListViews>
</CGM>
```

Verb History

New to CGM. No corresponding Dienst verb.

Stage II Verbs

Not critical, but not optional?

- DescribeVerb
- ListVerbs

Describe Verb

Verb name: DescribeVerb [this needs work yet]
Verb version: 1.0 alpha
Modification Date: 2003-01-01
Required arguments: value
Optional arguments: none
MIMETYPE of response: text/xml

Verb Description

Returns a structured response that contains a list, where each element of the list provides information on a version of the specified verb that is supported by this service.

Arguments

- **`value`** :: [required] name of the verb to be described.

Response

The following information may be provided at the verb or version level.

- **`description`** :: , description of the verb or a specific version
- **`note`** :: , information pertaining to the verb or a specific version

Each element of the list contains the following information:

- **`version number`** :: of the verb.
- **`arguments`** :: , a list of the names of the fixed and keyword arguments, if any, accepted by the verb in that version.
- **`example`** :: template of request to this repository, with fixed argument indicated in brackets
- **`returns`** :: , optional, contains information about response format.

Note that a service may implement more than one version of a verb.

Error Codes

-

Examples

Request

```
protocol=CGM&verb=DescribeVerb&ver=1.0&value=Formats
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
  <DescribeVerb ver="1.0">
    <verb name="Formats">
      <description>Returns a structured response indicating
        the formats of disseminations available for this document
      </description>
      <versions>
        <version id="1.0">
              <example>http://www.umdl.umich.edu/cgi/b/broker/broker?
                                protocol=CGM&ver=2.0&

verb=Formats&identifier=identifiervalue></example>
           <arguments>
             <required>
               <arg name="identifier" />
             </required>
             <optional>
               <arg name="version" />
               <arg name="view" />
             </optional>
           </arguments>
         </version>
      </versions>
    </Verb>
  </Describe-Verb>
```

Verb History

Modification of Dienst verb Describe-Verb, version 2.0.

List Verbs

Verb name: ListVerbs
Verb version: 1.0 beta
Modification Date: 2003-03-24
Required arguments: none
Optional arguments: none
MIMETYPE of response: text/xml

Verb Description

Returns a structured response containing the name and version number of each CGM verb supported by this repository. Different versions of the same verb should be included, within separate `verb` elements.

Arguments

Response

The List Verbs response contains a single, repeatable element.

- **`<verb>`** :: an EMPTY element with two attributes:
  - `name` :: the verb name.
  - `ver` :: the version of the verb.

Error Codes

- **`badArgument`** :: the request includes illegal arguments, is missing required arguments, or contains

invalid argument values.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=ListVerbs&ver=1.0
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="ListVerbs" ver="1.0">
           http://some.cgm.server/script</request>
  <ListVerbs ver="1.0">
    <verb name="Disseminate" ver="1.0" />
    <verb name="Display" ver="1.0" />
    <verb name="Formats" ver="1.0" />
    <verb name="Search" ver="1.0" />
    <verb name="Search" ver="2.0" />
    <verb name="Structure" ver="1.0" />
    <verb name="ListViews" ver="1.0" />
    <verb name="ListVerbs" ver="1.0" />
  </ListVerbs>
</CGM>
```

Verb History

Dienst verb, version 2.0

Optional Verbs

Optional verbs:

- ListBinders
- ListEncodings
- ListVersions
- Terms

List Binders Available

Verb name: ListBinders
Verb version: 1.0 alpha
Modification Date: 2003-03-18
Required arguments: none
Optional arguments: none
MIMETYPE of response: text/xml

Verb Description

Returns a structured list of the types of binders available in this repository. [In this repository? or for a specific document, or even a specific div within a document? Does this make any sense at the repo level?]

Arguments

Response

Error Codes

- **badArgument** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=ListBinders&ver=1.0
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="ListBinders" ver="1.0">
           http://some.cgm.server/script</request>
  <ListBinders ver="1.0">
    <binder type="tar" />
  </ListBinders>
</CGM>
```

Verb History

Dienst verb, version 1.0

List Encodings Available

Verb name: ListEncodings
Verb version: 1.0 alpha
Modification Date: 2003-03-18
Required arguments: none
Optional arguments: none
MIMETYPE of response: text/xml

Verb Description

Return a structured list of the types of encodings available in this repository. [same questions as with ListBinders--should this be aimed at documents or document divisions rather than the entire repo?]

Arguments

Response

Error Codes

- **badArgument** :: the request includes illegal arguments, is missing required arguments, or contains

invalid argument values.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=ListEncodings&ver=1.0
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="ListEncodings" ver="1.0">
           http://some.cgm.server/script</request>
  <ListEncodings ver="1.0">
    <encoding type="gzip" />
  </ListEncodings>
</CGM>
```

Verb History

Dienst verb, version 1.0

List Document Versions Available

Verb name: ListVersions
Verb version: 1.0 alpha
Modification Date: 2003-03-18
Required arguments: identifier
Optional arguments: none
MIMETYPE of response: text/xml

Verb Description

Returns a structured response describing the current versions available for the requested document.

Arguments

- **`identifier`** :: [required] the document identifier for which version information is requested.

Response

- **`<version>`** :: its attribute `value` is the version number, a positive integer, used when requesting a specific version in Structure, Formats, and Disseminate verbs. The element contains two subelements:
  - **`<date>`** :: the date the version was last modified, expressed in the ISO 8601 format YYYY-MM-DD.
  - **`<comment>`** :: a comment or description.

Error Codes

- **`idDoesNotExist`** :: the value of the `identifier` argument is unknown or illegal.

- **badArgument** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=ListVersions&ver=1.0&identifier=cul.math/00640001
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="ListVersions" ver="1.0" identifier="cul.math/00640001">
           http://some.cgm.server/script</request>
  <ListVersions ver="1.0">
    <identifier value="cul.math/00640001">
    <version value="2">
      <date>2002-03-18</date>
      <comment>Author revised version.</comment>
    </version>
    <version value="1">
      <date>1999-10-01</date>
      <comment>Original published version.</comment>
    </version>
  </ListVersions>
</CGM>
```

Verb History

Modification of Dienst verb List-Versions, version 1.0.

Terms of Use

Verb name: Terms
Verb version: 1.0 alpha
Modification Date: 2003-03-18
Required arguments: identifier??
Optional arguments: none
MIMETYPE of response: text/xml

Verb Description

Returns terms of use and rights information [for a document, or entire collection? I'm assuming for a document here, for now. Some rights information is available at the repository level via OAI Identify].

Arguments

- **identifier** :: [required] the document identifier for which terms of use information is requested.

Response

Error Codes

idDoesNotExist :: the value of the `identifier` argument is unknown or illegal.

- **badArgument** :: the request includes illegal arguments, is missing required arguments, or contains invalid argument values.

Examples

Request

```
http://some.cgm.server/script?protocol=CGM
        &verb=Terms&ver=1.0&identifier=cul.math/00640001
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<CGM>
  <responseDate>2002-10-02T19:20:30Z</responseDate>
  <request verb="Terms" ver="1.0" identifier="cul.math/00640001">
           http://some.cgm.server/script</request>
  <Terms ver="1.0">
    <identifier value="cul.math/00640001">
    <terms>
      <date>2002-03-18</date>
      <comment>This digital material is made available by Cornell
         University Library, and is to be used for personal or research use
         only. Any other use, including but not limited to commercial or
         scholarly reproductions, redistribution, publication, or transmission,
         whether by electronic means or otherwise, without prior written
         permission of the Library is prohibited.</comment>
    </terms>
  </Terms>
</CGM>
```

Verb History

New to CGM?

Proposed Search verb ver. 2.0

This is a proposal for a Search verb ver. 2.0. Search ver 1.0 allows searches on bibliographic fields and full-text, but the full-text "region" is treated as a single structure (in effect, there is only one structure, maindocument). Thus, the only full-text boolean query a user can construct in ver. 1.0 is one for two terms occurring within the same maindocument (which in our case is an entire monographs). We recognize that this limits query precision and thus the usefulness of full-text searching.

However, abstracting a search protocol so that substructural regions can be identified across native document types is not trivial. We have struggled, particularly, with the distinctions between concurrent structures (especially logical and physical), but there are plenty of other problems, partially masked by the uniformity of our document types (monographs). A further complication is how a query service "understands" and makes use of structural relationships (parent|child, etc.) in order to construct meaningful searches, when these relationships may be different at each local repository. For example, one repository may have chapters with child elements of pages while another has chapters with no, or different, child elements. The query service is faced with significant data gathering and interpretation demands, merely to build an accurate search query form.

The basic premise of Search ver. 2.0 is that we go ahead and push 'abstraction' further, to include not only search query mechanics (as we've done in ver. 1.0) but document structures as well.

Search ver. 2.0 overview

Search ver. 2.0 would allow for documents with four abstract structural elements (or docStruct values), two of which are required (meaning that searches within these structures must be supported):

- maindocument [required]
- div-high
- div-mid
- div-low [required]

The local repository maps native document structures to abstract structures. Communities may have strict guidelines for such mapping, although it could be left to local decisions. For this project, I imagine a mapping to look something like this:

- maindocument: the entire document
- div-high: structures such as TEI front, body, back
- div-mid: chapters, sections, miscellaneous divisions
- div-low: page, paragraph, illustrations, charts (i.e., page or "sub-page" elements)

Again, mapping all of these is not required. If you have no high level document structures, you don't use the docStruct value "div-high". If your maindocument is an article, you'd most likely offer maindocument and div-low as the only available structures. If you've got a lot of mid-level divisions (chapters, sections, divs within divs), you'd decide which one to map to div-mid. Multiple mappings could be allowed (div-mid mapping to local chapter and section and whatever), but would mean more local expense (and perhaps redundant searching?).

The verb ListDocStruct would list the abstract document structures supported by a repository.

A parent|child|sibling relationship is assumed among the abstract document structures.

dwr, 2003-03-20

CGM Search - RPN Notation

This document describes the RPN specification of CGM Search verb arguments.

RPN (Reverse Polish Notation) provides a way to order boolean operations in such a way as to avoid the need for parenthesis. In our context the boolean query is decomposed and arranged according to the highest precedence operations first with operators applied in a postfix manner.

A simple mathematical example:

Example: 3 * (1 + 2)

May be represented in RPN notation in different ways:

RPN: 1 2 + 3 *
RPN: 3 1 2 + *

Both are equivalent. RPN uses the concept of a stack for applying operators to operands. In the second example the numbers 3,1,2 are on the stack and the '+' operation uses the top two operands on the stack, 1 and 2, leaving the stack with 3 3. An operation removes two operands from the stack and returns the result to the top of the stack.

The Search request allows arguments of the form fieldN, valueN, and opN. Our desire is to use RPN to present arguments to the Search verb.

The following are basic rules of RPN calculations:

1. Evaluation occurs from left to right.
2. Operands precede operator.
3. Operation results in operand.

For simple queries the mapping to verb arguments is straight forward.

Example: (title="math" AND title="Losung") OR author="hilbert"

As RPN: title="math" title="Losung" AND author="hilbert" OR

(Note that RPN eliminates the need for parenthesis.)

op2=AND op3=OR
field1=title field2=title field3=author
value1="math" value2="Losung" value3="hilbert"

In this scheme operator N operates on values N and N-1. For more complex queries operands may build up requiring repeated operators to be applied in succession. In the case where an operator occurs alone it operates on the previous two operands.

Example: (title="math" AND title="Losung")
OR (author="hilbert" AND title="tenth")
As RPN: title="math" title="Losung" AND author="hilbert"
title="tenth" AND OR

op2=AND op4=AND op5=OR
field1=title field2=title field3=author field4=title
value1="math" value2="Losung" value3="hilbert" value4="hilbert"

Note that op5 does not have a field5/value5 defined. It operates on the operands that result from op2 and op4.

Translating queries to RPN is straightforward. Create a graph of the query with low precedent operators (OR) high in the graph and high precedence operations (AND). Parenthesis increase the precedence of an expression. In the last example the tree looks like:

```
OR
/ \
AND AND
title/math title/Losung author=hilbert title=tenth
```

Evaluation of the RPN notation results in a single result set such that the number of operators is always one less than the number of field/value pairs. RPN expressions that contain more than N-1 operators are invalid.

For certain operators the order of operands is significant. For the ANDNOT, INCLUDING, and WITHIN operators the order of operands is the same as in the non-RPN query expression. The NOT operator is the binary ANDNOT form of NOT and not the unary form.

Example: title/math ANDNOT title/Losung As RPN: title/math title/Losung ANDNOT

David Fielding, 2002-12-13