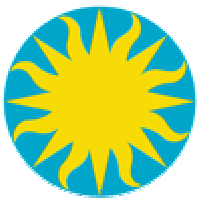

Smithsonian Advance Engineering Support

EBCA/BCAC High-Level Architecture White Paper

Smithsonian Institution

January 10, 2003

Prepared for the Director of System Architecture and Product Assurance, Office of the Chief Technology Officer

Table of Contents

Table of Figures

List of Tables

SECTION 1
INTRODUCTION

1 SECTION
INTRODUCTION

1.1 Purpose

This white paper describes a proposed High Level Architecture (HLA) to support the Electronic Biologia Centrali-Americana (EBCA) and Biologia Centrali-Americana Centennial (BCAC). Based on the available technical requirements at the time of this report, which is almost exclusively for EBCA, EBCA will be the major focus of this High Level Architecture. BCAC will be discussed whenever appropriate.

The primary purpose of EBCA is to provide a digitized copy of the 58 biological volumes of *Biologia Centrali-Americana* on the World Wide Web with a set of electronic tools and resources for biodiversity studies. With staff of NMNH providing scientific guidance, the project will serve as a model for several other major bioinformatics projects pursued by the world's leading biological repositories. Together these will speed the pace of scientific investigation into the nature of our rapidly changing biological environment.

1.2 Background

The *Biologia Centrali-Americana* is a fundamental work for the study of Neotropical flora and fauna and includes nearly everything known about the biological diversity of Mexico and Central America at the time of publication. The *BCA* was privately issued in installments between 1879 and 1915 by F. Ducane Godman and Osbert Salvin of The Natural History Museum (London). The work consists of 63 volumes containing 1,677 plates (of which more than 900 are colored) depicting 18,587 subjects. The total number of species described is 50,263 of which 19,263 are described for the first time. The contents were derived from scientific surveys and explorations conducted during the latter part of the 19th and early 20th centuries. Many of the leading biologists of the time provided specimens and descriptions for the many volumes. The illustrations are, in many cases, the only images that exist of the biota of the region and as such could be compiled for use in an electronic field guide if available in a digital and portable format.

Smithsonian Institution Libraries (SIL) has received a grant to complete the first stage of a multi-phase project that will create a model set of electronic tools and resources for biodiversity studies. The first phase consists of scanning, rekeying, and coding in eXtensible Markup Language (XML) the contents of 58 biological volumes of the out-of-print Biologia Centrali-Americana, a fundamental resource for the study of Central American flora and fauna compiled from scientific surveys and explorations conducted during the late 19th and early 20th centuries. Many of the period's eminent biologists contributed specimens and descriptions, and the accompanying illustrated plates are often the only images of Central American biota in existence. The coded text will be mapped to a database and linked to other vital biological datasets, including the National Museum of Natural History (NMNH) collections information system (the NMNH Multimedia Catalogue System), so scientists world-wide can study specimens in the

context of the published scientific record. With staff of NMNH providing scientific guidance, the project will serve as a model for several other major bioinformatics projects pursued by the world's leading biological repositories. Together these will speed the pace of scientific investigation into the nature of our rapidly changing biological environment.

The digitized contents captured in both text and image information in electronic form, organized in a database, will be accessible through the web to Smithsonian research staff and outside scholars for research and management purposes, and to the public for educational and recreational purposes.

Web application services infrastructure can be applied to address the Smithsonian's web related requirements to improve effectiveness and reduce costs. One of the primary goals and proposed benefits of this high level architecture is to leverage resulting systems and their component hardware, software, and staff expertise across SI to the benefit of all the units involved and, particularly, to SI's public and scientific customer base.

SECTION 2
SYSTEM REQUIREMENTS AND CONSTRAINTS

2 SECTION
SYSTEM REQUIREMENTS AND CONSTRAINTS

2.1 System Requirements

This section is not intended to capture the full details of the requirements for EBCA/BCAC, but instead capture the major requirements that are architecturally significant.

Following are a list of major requirements:

- The system shall support web-browser clients for basic search and retrieval.
- The system shall support complex XML documents and image files in TIFF, GIF, PDF, and HTTP format.
- The system shall provide linking to other remote biological databases for selected specimen including but not limited to the following
    - Specimen collections in large biological repositories such as Smithsonian's National Museum of Natural History through the interface provided by KE EMu.
    - The Smithsonian Institution Research Information System (SIRIS) online catalogs.
    - The Global Biodiversity Information Facility (GBIF, see www.gbif.org) aggregation services. EBCABCAC will be a participant node to supply content to the aggregation services, as well as a client node to make use of GBIF's services.
    - Other partner sites including Tropicos from Missouri Botanical Garden (www.mobot.org), The Natural History Museum, London (www.nhm.ac.uk), CONABIO in Mexico (www.conabio.gob.mx), and INBio in Costa Rica (www.inbio.ac.cr).
- The system shall allow admin users to configure, add, delete, or modify links to remote biological data sets.
- The system shall allow for tag-specific indices and allow end users to search and retrieve on predefined XML tags.
- The system shall support occasional additions, corrections, and changes to the EBCA data with proper authorization.

2.2 System Constraints

Following are the major constraints for the system from architecture perspective:

- The system should fit in the technical infrastructure of Smithsonian Institution based on the Technical Reference Model and the Strategic Technology Plan.
- The system should conform to the security architecture of the network in Smithsonian Institution.
- Technologies, products, tools, or methodologies should conform to the Technical Reference Model unless waiver has been obtained from the Architecture Review Board.

- Whenever possible and appropriate, the information technology procured should be non-proprietary, off-the-shelf products.
- The system shall be built on industry standard technologies with open interfaces (e.g. XML and J2EE).
- The system shall support Unicode and to certain extent, Spanish.
- The system shall support Section 508 standard (www.section508.gov) to provide accessibility to people with disabilities.
- The system does not support the full Z39.50 (www.loc.gov/z3950) compatibility.

SECTION 3
PROPOSED SYSTEM ARCHITECTURE

3 SECTION
PROPOSED SYSTEM ARCHITECTURE

3.1 Objectives

The objectives of the proposed high-level architecture are to identify technologies available in the market place and establish a solid technical foundation and framework for the proposed web application services infrastructure that supports the requirements for EBCA/BCAC. This will enable successful development of a system to meet the high-level requirements.

For further information on project-specific and detailed functional requirements, please refer to the draft technical requirement document.

3.2 Proposed System Overview

The proposed system overview is described from two different points of view – Section 3.3 focuses on the logical view of the application architecture. Section 3.4 provides a broader view of the system with a focus on the physical model (e.g. hardware infrastructure) and the external environment (e.g. network infrastructure and security infrastructure).

3.3 Application Architecture

The diagram in Figure 3-1 depicts the logical architecture of the proposed EBCA/BCAC system.

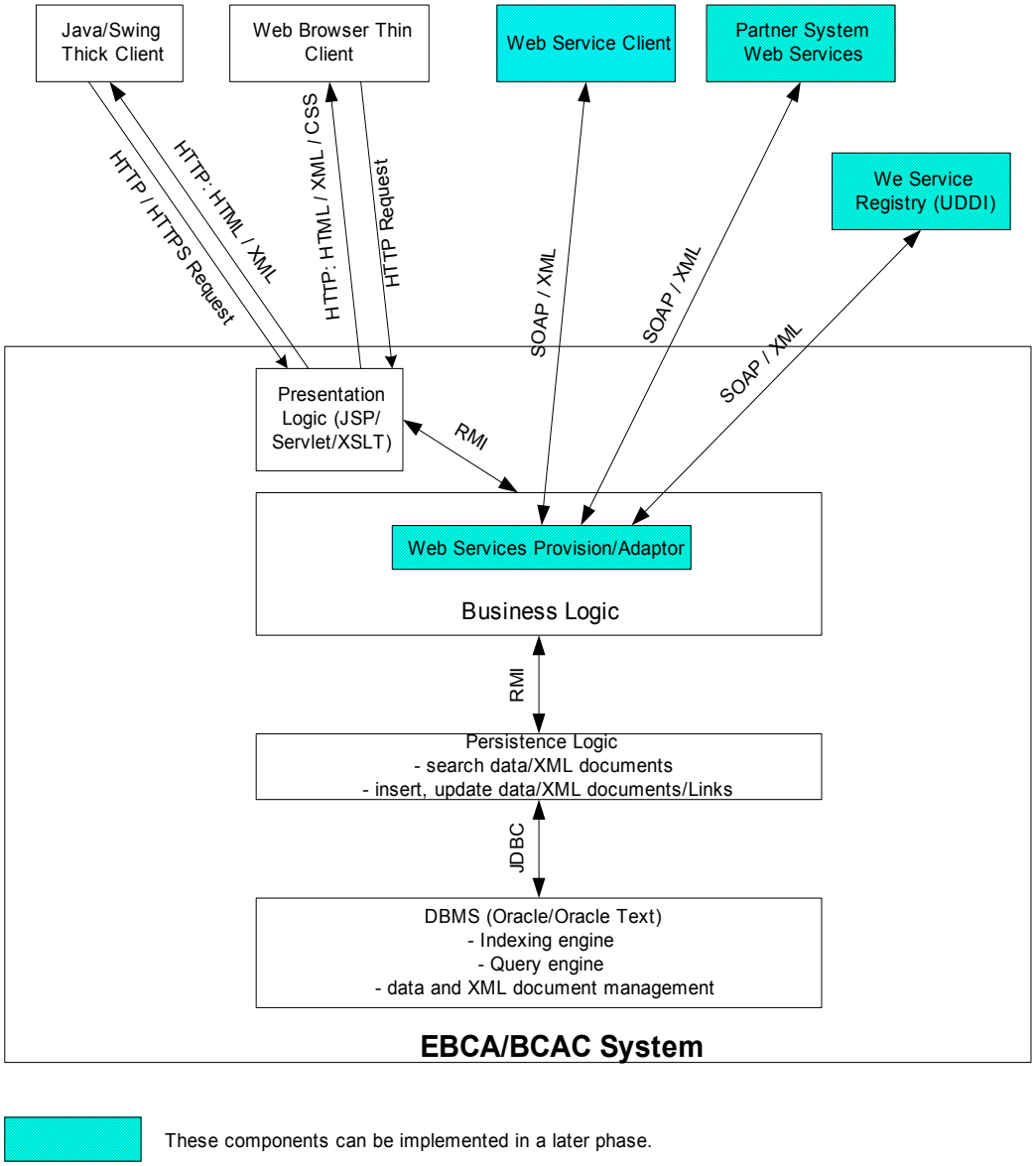

Figure 3-1. Logical Architecture of the EBCA/BCAC System

The EBCA/BCAC system is a multi-tier web-based application, which allows for more flexible deployment architecture to support large user load, maintain availability, and apply different level of access control. EBCA/BCAC will be built on top of an industrial-strength web application server (such as WebLogic Application Server) and database management system (DBMS) such as Oracle 9i. Three major logical tiers, Persistence Logic tier, Business Logic tier, and Presentation Logic tier, handle distinct responsibilities in the overall architecture.

Persistence Logic

The Persistence Logic tier is responsible for connecting to the DBMS to insert, update, or retrieve data or XML documents to and from the database. It translates the data and document

need from the Business Logic tier into database access commands and deals with the idiosyncrasies of the underlying database management system. It provides a set of well-defined interfaces for the upper tiers to use. It sends back either data or well-formed XML documents to the Business Logic tier.

The Persistence Logic tier takes advantage of the indexing and text searching capability provided by the DBMS or its related products to support a versatile search for end users. Optimization for performance purposes can happen in this tier, in conjunction with the underlying DBMS.

Well-defined design patterns such as Data Access Object (DAO) may be applied in this tier. Java Database Connectivity (JDBC) is used to connect to the DBMS. If the XML documents are stored in pieces in the database, this tier will assemble an XML document from the data elements and send the XML document back to the upper tier.

The relationship encapsulated by the XML schemas will be implemented through XML document structure, or through data model design. A DBA can help in validating or fine-tuning the performance of the data storage solutions.

Business Logic

The Business Logic tier provides a set of capabilities to directly implement the end user requirements.

To support specific user search requirements, the Business Logic tier will provide interfaces to allow for different kinds of search capabilities, such as search by taxon name, collectors, etc., or searching a classification system(i.e., browsing a taxon tree), or browse the volumes in the BCA series. What data are needed for each search or browse is defined in the interface of this tier.

This tier may also provide capabilities for loading the BCA data into the system. It will provide the necessary capability for adding, updating, and deleting of links to external web services, systems, or URLs.

This tier also makes available some or all of its capabilities through Web services. Based on the requirements, it will publish the selected functionalities in Web Services Definition Language (WSDL) to a registry and fulfill any Web Service requests via SOAP from clients.

If needed, this tier will act as a Web Service client to activate remote Web Services on a partner system, bring back the result, and present it through the Presentation Logic tier.

The Business Logic tier supports Java Remote Method Invocation (RMI) or Simple Object Access Protocol (SOAP) based on XML.

EBCA is mainly concerned with loading the contents of books in the BCA series and presenting the contents to researchers and public. The nature of data manipulation, after the initial data are loaded, will mainly be read-only. The link maintenance capability is the major requirement for data update. The data update capability will be accessed only by a small number of people

(internal Smithsonian staff and possibly a small number of external people that may have access to the database through a VPN). Access control is not a critical issue. Due to the large variety of external links, defining a flexible framework where links can be easily maintained and updated becomes very important.

For BCAC, even though the requirements have not been fully elaborated, the vision is to allow researchers to contribute data to the system through a public network. The contributions from researchers from around the world will go through a staging process. SI staff and a limited number of non-staff, who may be able to access the database through a VPN, can review and approve the contributions before they are published. Access control will become critical. The scale of contributions is expected to be fairly small when compared to industry standard. Nonetheless, transactional control will become more important as user contention may arise with concurrent updates to data.

Presentation Logic

The Presentation Logic tier presents the result to users in a user-friendly format, in HTML with Cascade Style Sheet (CSS), or in XML with XSLT. It keeps track of user HTTP sessions through cookies or URL re-writing. User authorization to access certain functionalities will be control in this tier.

The application, especially the EBCA part, is mostly stateless. If there are requirements to keep track of user profiles and support personalization, this tier will be responsible for supporting them. A user's search history can be supported through the HTTP session. Any state information that needs to go across multiple sessions, i.e. connecting to the server at different times, will need to be stored in the database.

Section 508 (http://www.section508.gov/) under the Rehabilitation Act is supported through the user interface design for the Presentation Logic tier, to provide accessibility to people with disabilities. If necessary, the Presentation Logic tier is also responsible for supporting multiple modes of access, such as regular web browsers, wireless device interfaces, or PDAs.

Even though web browser is the preferred client to access the system, Java Swing thick client can also be supported by the Presentation Logic tier, if necessary, to provide more sophisticated user experiences and efficiency of manipulating data.

3.3.1 Linking Architecture

As stated in the technical requirements, linking to remote heterogeneous biological data sets is crucial to biodiversity studies. However, maintaining the links can be laborious and error prone if the links are static URLs and maintained manually.

The external systems EBCA/BCAC link to may not exist yet; or if they exist, they may not have been built with the intention of cooperating with each other; or EBCA/BCAC may not be aware of those external systems. How to discover and locate external data sets and link to them is an issue.

The architecture solution to the linking problems calls for classification of the existing and potential future external systems and approaching each category of linking in a manner that allows the links to grow over time.

There are four major categories of linking the system needs to support:

- Forwarded HTML link
- Standard HTTP request
- Proprietary client
- Web services

The system provides a **linking management** module which allows admin users to define, add, delete, modify, or configure links through a Web interface. It also provides **client adaptors** which will be used in run-time to perform the actual linking and lead the users to the linked contents or interface where user can search for contents.

3.3.1.1 Forwarded HTML Links

Forwarded HTML links redirect the users' requests to a separate window with a search interface that is hosted on a partner site. Users enter a new site and search for related information. This is the most primitive way of linking.

These links are static. The discovery and maintenance of these links is mostly through a manual process. Defining the relevancy of these links to the tagged fields in a XML documents, or the whole XML documents themselves, is also manual. The whole process is error-prone. This kind of link needs to be limited in the future.

The system should provide management interfaces for admin users to define and maintain these links and associate them with related fields or documents. These links should be handled at the page level if possible. Configuration is trivial for these links since they are static. Since these are just jump-off links where the presentation of results is handled by external systems, no client adaptors need to be built to support them.

3.3.1.2 Standard HTTP Request Links

Standard HTTP requests allow the system to pass key information through an HTML link and send a request to a remote partner web server. The search resulta will be displayed on a separate window for users, presenting the resulting document as returned by the remote system.

These links have similar characteristics as the forwarded HTML links described above. These links are static. The discovery and maintenance of these links is mostly through a manual process. Defining the relevancy of these links to the tagged fields in an XML documents, or whole XML documents themselves, is also manual. The whole process is error-prone.

Most partners do have Web presence and support these kinds of links. These links comprise the majority of the links the system maintains in the near future. With strong partnership, EBCA/BCAC can also request the partners to add features to their systems and make the new features available through standard HTTP requests.

The system should provide management interface for the admin users to define and maintain these links and associate them with related fields or documents. Relevant parameters needed for the requests should be configured with the links so they can be passed in run-time. Since these are just jump-off links where the presentation of results is handled by the external systems, no client adaptors need to be built to support them.

For remote sites that need to access the functionalities of EBCA/BCAC, the system supports standard HTTP request. For example, the SIRIS system can provide a link on its search result set which leads the user to the browsing by volume function on EBCA to access the contents in the BCA series.

3.3.1.3 Proprietary Client Links

For external systems that do not have a Web-based interface, special client adaptors may need to be built so the system can connect to external proprietary systems, make the request through the proprietary protocol and interface, retrieve the data in either XML or proprietary format, and then present the data to the end user in a proper format.

So far we have not identified any of such proprietary partner systems. These category of links need to be limited.

The system should provide management interface for the admin users to define and maintain these links and associate them with related fields or documents. These links are in general a request back to the system which in turn uses the appropriate client adaptor to retrieve the result sets from the external proprietary systems and present them back to the end users.

3.3.1.4 Web Services

The system can act as a client to a remote partner system Web Services to retrieve data as part of its contents to be presented to the end user through Web Service client link. Similarly, the system can also act as a service provider by publishing its Web Services described in Web Service Definition Language (WSDL) to a well-known registry, and responding XML based SOAP requests.

For the Web service client links, the system acts as a Web service client to the remote partner system which provides searches as Web services and publishes the services in a UDDI registry. The system will look up the service through the registry, and then uses the Web service description information to access the provider site to bind to the service and retrieve the information in XML. The system then presents the result to the users in its own format.

Web service is a service-oriented architecture based on XML, SOAP, WSDL, and UDDI. It provides the technical infrastructure and standards to enable XML-based enterprise integration,

content aggregation, workflow management, and cooperation between organizations. It enables a system to publish and provide services independent of its implementation. Please see Appendix A for more details about Web services.

With Web services, the system can translate a link into a Web service request to a remote site, obtain the results in XML format, and present the results to the end users.

The system should provide management interface for the admin users to define and maintain these links and associate them with related fields or documents.

Adoption for Web Service and UDDI Registry

There have not been many Web services or UDDI registries deployed for production systems yet, especially in the biological taxonomy community. From architecture perspective, Web service is the technology of choice for one of the two "legs" of the system (HTTP based links and Web service based application-to-application integration).

Currently even though the public registries hosted by IBM, Microsoft, HP, and SAP have been up and running and support production-class operations since May 2001, the actual business transaction on them has been limited. Private registries, which are managed by big corporations, consortiums, or specific industries, have been in pilot phase since early 2002. With the UDDI standard transferred to OASIS and the release of UDDI v3, implementation of registries has gain a lot of momentum. Although Web service itself is still evolving, it is very promising for application integration, internal to an organization as well as outside suppliers and partners.

There is no production-class registry for data similar to EBCA for the moment. However, the EBCA/BCAC system is built to last for many years to come, and by the time the system is built, adoption of Web services may have been widespread in many production systems. The system will risk being a "legacy" and playing catch-up soon after its release if it's not Web service ready.

Implementation Strategy

Web service is the preferred way to link to other system and allow other systems to link to EBCA/BCAC. The architecture from the Global Biodiversity Information Facility (GBIF, see www.gbif.org), which EBCA/BCAC will cooperate with, seems to support the Web Services paradigm well. GBIF may provide a registry for Web services, or SI can have its own registry for Web Services. The EBCA/BCAC system can publish Web services through the GBIF registry and thus allow partner systems to locate and access the Web services provided by the system. EBCA/BCAC can also act as a client to make Web service request to GBIF or other systems to make use of their services to access related information.

It may be feasible to work with GBIF to set up a pilot based on Web service. This way, the EBCA/BCAC will be a good test bed for the GBIF project.

Another possibility is to build, deploy, and test the Web service components with a partner for its system with which EBCA/BCAC will integrate. If this approach is successful, future integration can be done in similar manner, instead of being forced into building thick clients when application to application integration is needed.

Following are two patterns for Web services at the architecture level.

3.3.1.4.1 Web Services: Provider Pattern

To provide Web Services to service external system requests such as those from the GBIF aggregation servers, the system needs capability to register services and depends on a service registry to publish its services. The diagram in Figure 3-2 depicts the participants and interaction among them to fulfill the provision of Web Services. All requests and responses are carried out through SOAP based on XML.

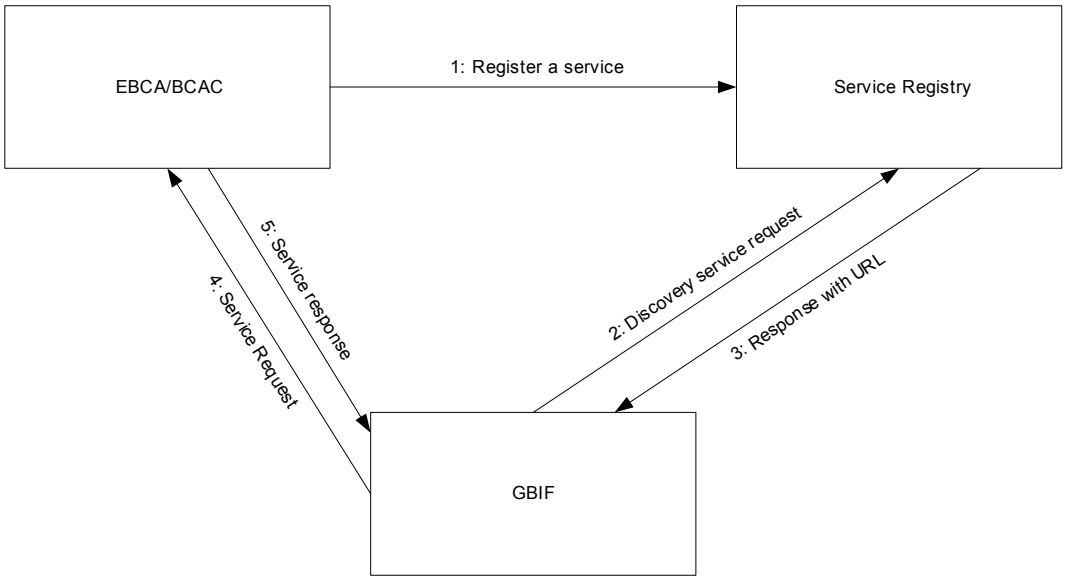

Figure 3-2. Web Services Provider Pattern

3.3.1.4.2 Web Services: Client Pattern

Figure 3-3 depicts a typical scenario to access remote Web Services. The EBCA/BCAC system discovers the availability and description of Web Services through a Service Registry based on Universal Description, Discovery, and Integration standard (UDDI). It then uses the information contained in the service description based on WSDL to locate the physical system, uses SOAP to send an XML request to invoke the remote services. It can process and present the XML result from the remote service to the end user.

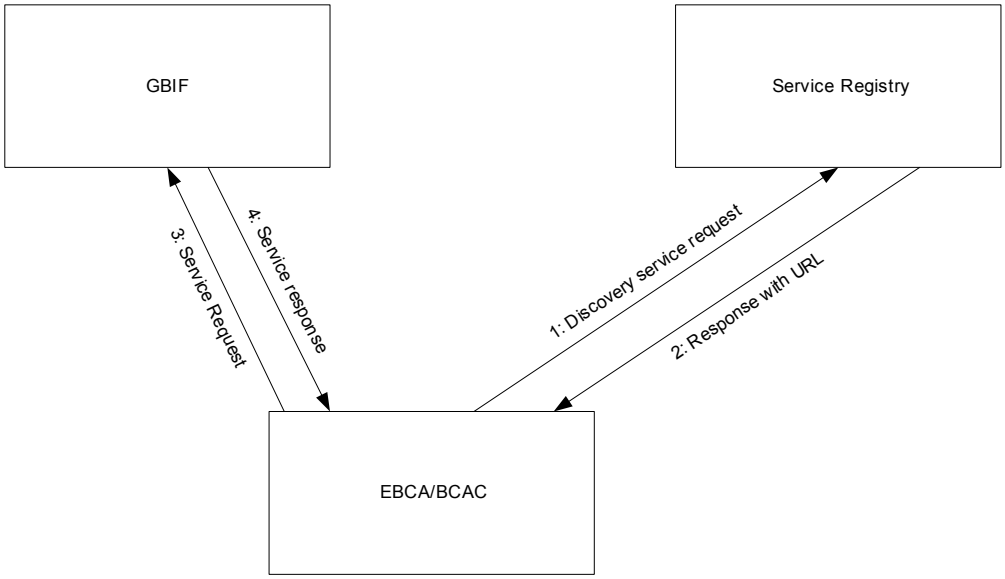

Figure 3-3. Web Services Client Pattern

3.3.2 Storing Image Files

The scanning of the volumes in the BCA series of books yield a set of image files per page in JPEG and/or PDF formats. These files can be stored in and served up by the Web servers, accessible through regular HTTP links on a page. The links are stored in the database, or with the XML documents, to allow the application to retrieve them and render on a HTTP page to end users.

An alternative way is to store the image files in the database. While this is doable, it's not recommended because this may add significant load to the DBMS, considering the amount of image files we have (about 26,000 pages for BCA).

A consistent naming convention needs to be defined in order to maintain the links efficiently and reduce errors that may cause mismatch between the file access path and the links stored in the database.

3.3.3 Persisting XML Documents

XML documents are the format of choice for information exchange. The creating and parsing of XML involves a lot of text processing which is CPU intensive. There are two main ways to support the storage of XML documents. One way is to store an XML document as a Character Large Object (CLOB). Another way is to break an XML document down into pieces and store them in an object-relational database.

The first approach maintains the fidelity of the XML document and, with proper indexing support, is searchable at its element level. This approach is suitable for static documents that are not updated often, since each update will happen at the whole document level. Once a document

is created and stored in this way, subsequent retrieval does not require XML creation because the whole XML document is retrieved intact from the database.

The second approach provides a lot of flexibility in retrieving and manipulating an XML document at the element level. It is suitable for dynamic documents where update is frequent. The overhead of retrieving multiple objects and putting them together into an XML document based on a schema can be high.

3.3.3.1 XML Schema Consideration

The XML schemas for EBCA focus on the taxonomic structure and semantics of the BCA, as well as support the bibliographic record for the individual volumes and the series as a whole (probably using the MODS schema), and mark-up of the non-taxonomic portions of BCA based in TEI-Lite for structural elements such as prefaces, introductions, tables, etc.

To describe a page context, a page break is defined for each page. The purpose of the page break is three-fold: a) to provide the user with a reference to an exact portion of the text which may be cited in other literature (e.g. you are now in Botany, volume 2, page 6); b) to allow for "drill-down" navigation so that the system can pin-point a portion of the text from a reference printed elsewhere (e.g. Botany, volume 5, page 64); and c) to provide navigational markers that will allow the system to "page forward" or "page backward" within a screen display to allow the user to sequentially through the digital representation of BCA.

Each page break will reference both the page and the volume (note: the "volume" in this instance may be a construct that does not bear any relation to any particular bound printed set of BCA). The page break will inherit the volume number and part number from the parent node. Therefore, we can search for all elements on page 5 for volume 15 by using the predicate:

```
*/pagenumber[@number='15_5'].
```

It is expected that each search will get a taxon back without the complete set of children (taxa at the next rank down in the taxonomic hierarchy). When a user clicks on a link, the relevant child section (taxon) will be retrieved. A well-defined intermediate page for user to navigate through the child taxa is needed.

It is assumed that the parent information (i.e. "volume" information) will include complete bibliographic citations (author, publisher, place of publication, etc.) contained in a MODS schema. Additionally, elements such as "chapter headings" (which would be drawn from taxonomic sections of the text would be included in the MODS bibliographic information as pseudo tables of contents. Fascicle information (e.g. printers' marks and date of publication) would also be contained in these records (as well as in the marked-up text) and available to the user.

It is assumed that the sequencing of the content is captured inherently during the re-keying process by preserving the order of the text in the books. If the individual sections are stored as

elements, the sequences need to be assigned a unique sequence identifier in order to maintain the order of elements in the original books.

3.3.3.2 Database Selection

Based on the research in Appendix B, Oracle 9i has strong support for XML data. Oracle 9i is also a proven top choice for relational data. It is recommended that Oracle 9i be the DBMS for EBCA/BCAC.

Oracle can store an XML document as a Character Large Object (CLOB). From pure data storage perspective, it's not different from storing any files of other formats. On top of that, Oracle allows for validation of the data based on a schema, indexing based on the whole document while handling the tags properly, navigation based on XPath standard, delivering data in XML format as the result from a query, and utilities to manipulate XML documents. Oracle can also take an XML schema and automatically generate necessary object-relational tables to store XML documents based on that schema in what is called XML DB Repository, which supports element level query and update and XPath standard for retrieving contents of the document, as well as provides mechanism to support relational views on the XML DB Repository through its efficient and powerful relational engine.

While Oracle 9i has strong support for XML data, its core is based on relational tables. This means it may not be able to handle the complex relationship inherent in EBCA/BCAC data. It's recommended that a prototype to validate the capability of Oracle 9i's support for the specific requirements in EBCA/BCAC. In the rare situation, if Oracle 9i cannot fulfill the need from the requirements, a native XML DBMS may need to be identified and adopted.

3.3.4 Indexing and Search Solution

Oracle 9i comes with Oracle Text, a product supporting full text search with extensive indexing and text searching capability. Oracle Text also supports index and search of XML documents.

3.3.5 Availability, Scalability and Performance

EBCA/BCAC is created to allow researchers all over the world to access the BCA information, as well as share information on new specimens, species, collections, citations, etc. As the system will be used actively by researchers around the clock, availability of the information is critical.

Availability and scalability is achieved through clustering of servers and system resources. The system is deployed in several tiers. Each tier needs certain level of redundancy to maintain the availability of the system. Clustering is much more than just adding extra servers. A cluster can provide one logical name for a group of servers so the outside clients only need to know one point to connect, and the client loads are distributed through the cluster. In some cases, the user state information is maintained across servers in a cluster. If one server goes down, user requests to that server will be automatically redirect to other servers in the cluster without the use even aware of the failure. Clustering can also allow system resources to be added at run-time without shutting the entire system down. Having the capability of adding system resources during run-

time allows the system to respond to more user traffic easily, i.e., increase the scalability of the system.

Clustering does incur communication overhead on the system which can have substantial performance penalty if not configured properly. Clustering is very suitable for mission critical systems. For EBCA/BCAC clustering may not be as critical, depending on the expected load and performance of the system.

Performance is achieved through an optimized architecture. Performance can be improved through a good logical design and a well thought out deployment architecture.

3.4 System Architecture

Figure 3-4 shows the proposed physical system architecture, which depicts the hardware, network, and security control. The network and security architecture is based on the proposed "SI Firewall Redesign High-Level Architecture". This architecture takes advantages of the existing or future technical infrastructure and services provided by the Smithsonian Institution networks.

This system must support visitors from the Internet using web browsers such as Internet Explorer and Netscape Navigator. In addition, the architecture supports visitors physically inside a museum using stationary computers, kiosks, and handheld devices such as Personal Digital Assistants (PDAs). The system components are distributed across different networks and SI subnets (e.g. the Internet, an SI perimeter network made up of multiple DMZ zones).

The data in EBCA/BCAC are not considered to be sensitive, i.e., having access to the data by potential malicious parties will not cause significant harm. In this proposed architecture, the internet application server component and database server component are put in the DMZ-2. This is to reduce the traffic and access path to the internal SINet while providing sufficient security control to protect the application and database systems. The application servers and database servers will also have ServerLock (see http://www.watchguard.com/products/serverlock.asp) product installed to protect the machines from modifying by any applications that are not designated to do so. In order to achieve high availability to support 24x7 web access, server clustering is needed. The balancer distributes the user requests to the servers in the cluster evenly to provide reasonable performance.

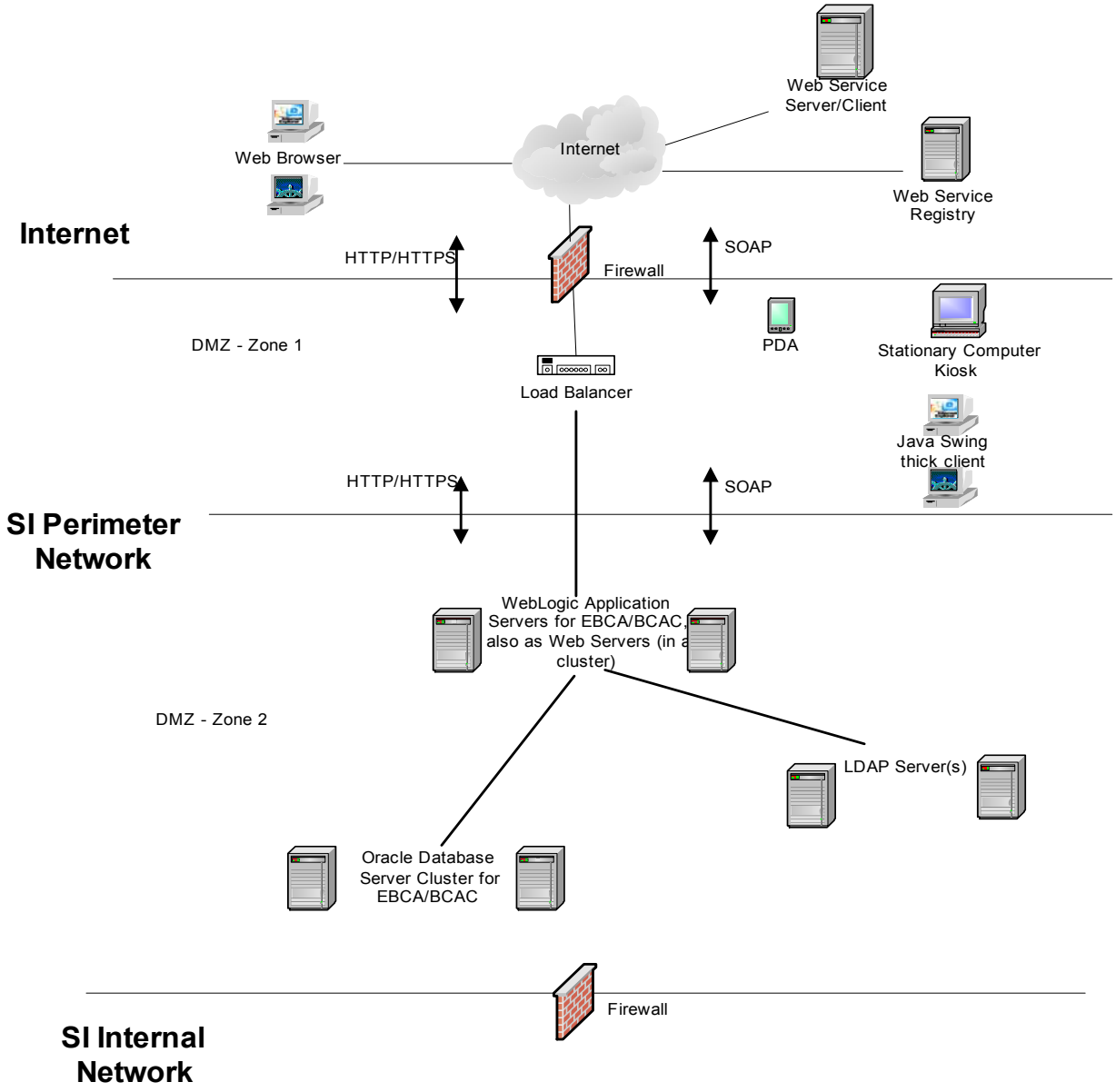

Figure 3-4. Physical View of the EBCA/BCAC System Architecture

3.4.1 Map the Logical Architecture to Physical Architecture

Figure 3-5 shows how the logical components can be deployed to the physical components, i.e., servers. Three logical tiers of components can be deployed to three tiers of servers. The recommended deployment strategy based on the requirements is to deploy all three tiers of components to one tier of physical servers. Following are some major considerations:

- The initial projected load of the system is not particularly heavy (4 to 5 internal or editorial staff, and a few hundred researchers). Two servers in a cluster can accommodate this projected load.
- Passing XML documents, which can be large, across servers, can take up bandwidth and CPU cycles due to marshaling and un-marshaling of data for RMI. By collapsing the tiers, the system is more efficient.
- Access control is quite straightforward. Only internal (or effectively internal using VPN) staff need to register and log on to update data. Public users do not need accounts, and need read-only access to information.

The logical design of the application into tiers and variety of security mechanisms provided by the infrastructure allows the deployment to have many alternatives, based on the user load, performance requirements, access control requirement, and budget constraints.

Since the data for EBCA is not sensitive. Regular backup may be sufficient for operational purposes. In this case, the application gateway may not be critical. The database server could sit in the DMZ 2, potentially combined with the application servers if budget is an issue. With such shortcuts, though, there will be impact on the performance and availability of the system.

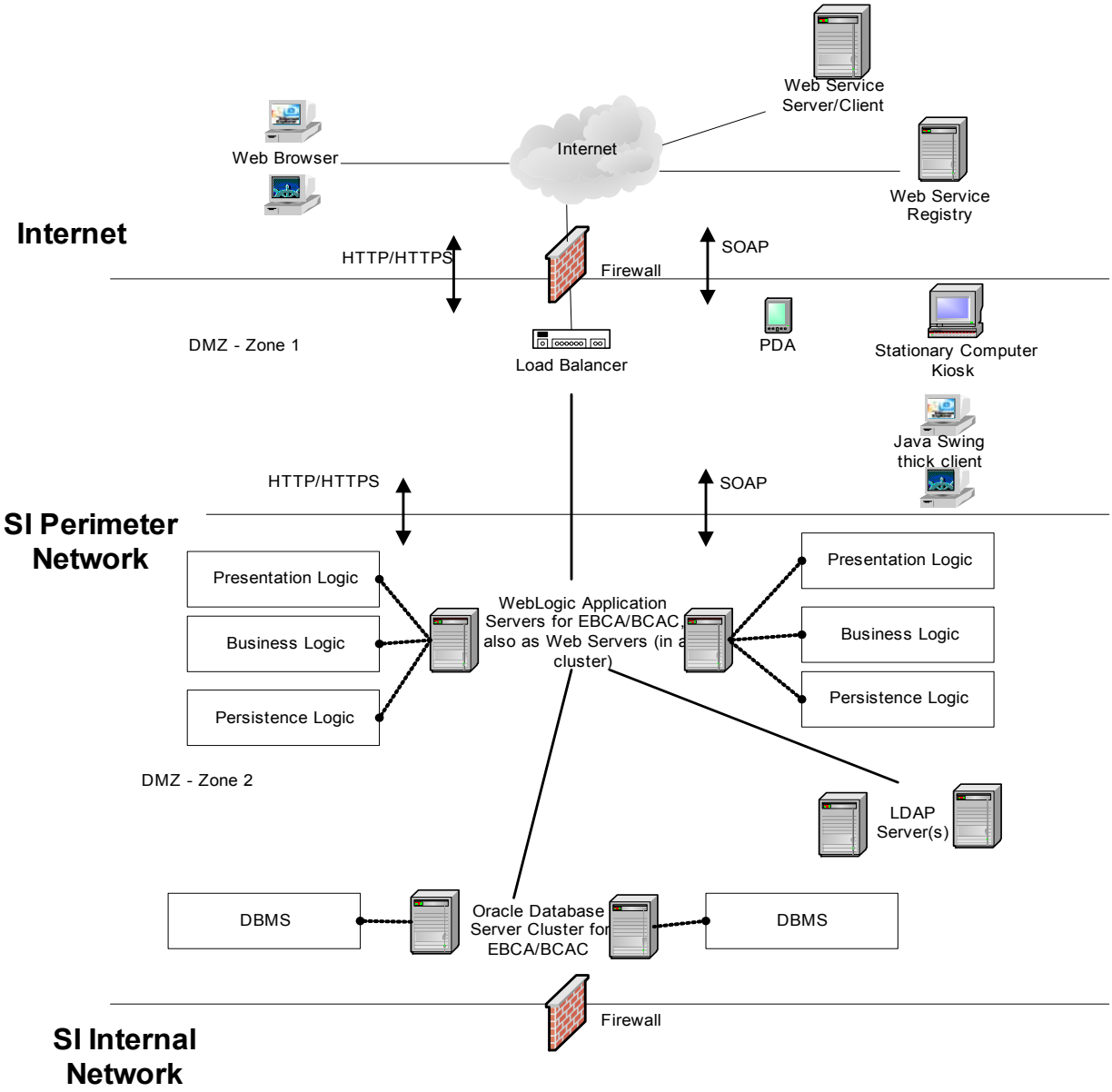

Figure 3-5. Map Logical Components to Physical Components

3.4.2 System Security Architecture

The proposed system architecture requires the existence of a supporting, SI-wide, generic security architecture. This is shown in Figure 3-6. The security components are addressed in more depth in the Smithsonian Information Technology Plan under an SI-wide security / firewall architecture. Regardless of whether the security architecture is implemented as part of an SI enterprise security infrastructure project or as part of the EBCA/BCAC project, it needs to be consistent with overall Smithsonian enterprise architecture, and implemented as a separate, but

integrated, infrastructure component of any proposed implementation of web application services.

This security architecture incorporates multiple levels of access and control, and is based on a layered protection and detection approach. It requires a security infrastructure that includes firewall servers and firewall rule sets, perimeter network(s) with multiple Demilitarized Zones (DMZs) that separate internal SINet traffic from the Internet. Intrusion detection of both network and host-based types is important. A secured and separate management zone for the management of all devices and hosts within the infrastructure is also important. Such a management zone allows logging and reporting information flow from the devices through to the management hosts, while content, configuration and policies flow to the devices from the management hosts. Network-based intrusion detection systems (IDSs) should be placed at critical network locations and host-based IDSs should be installed on critical, high risk, systems.

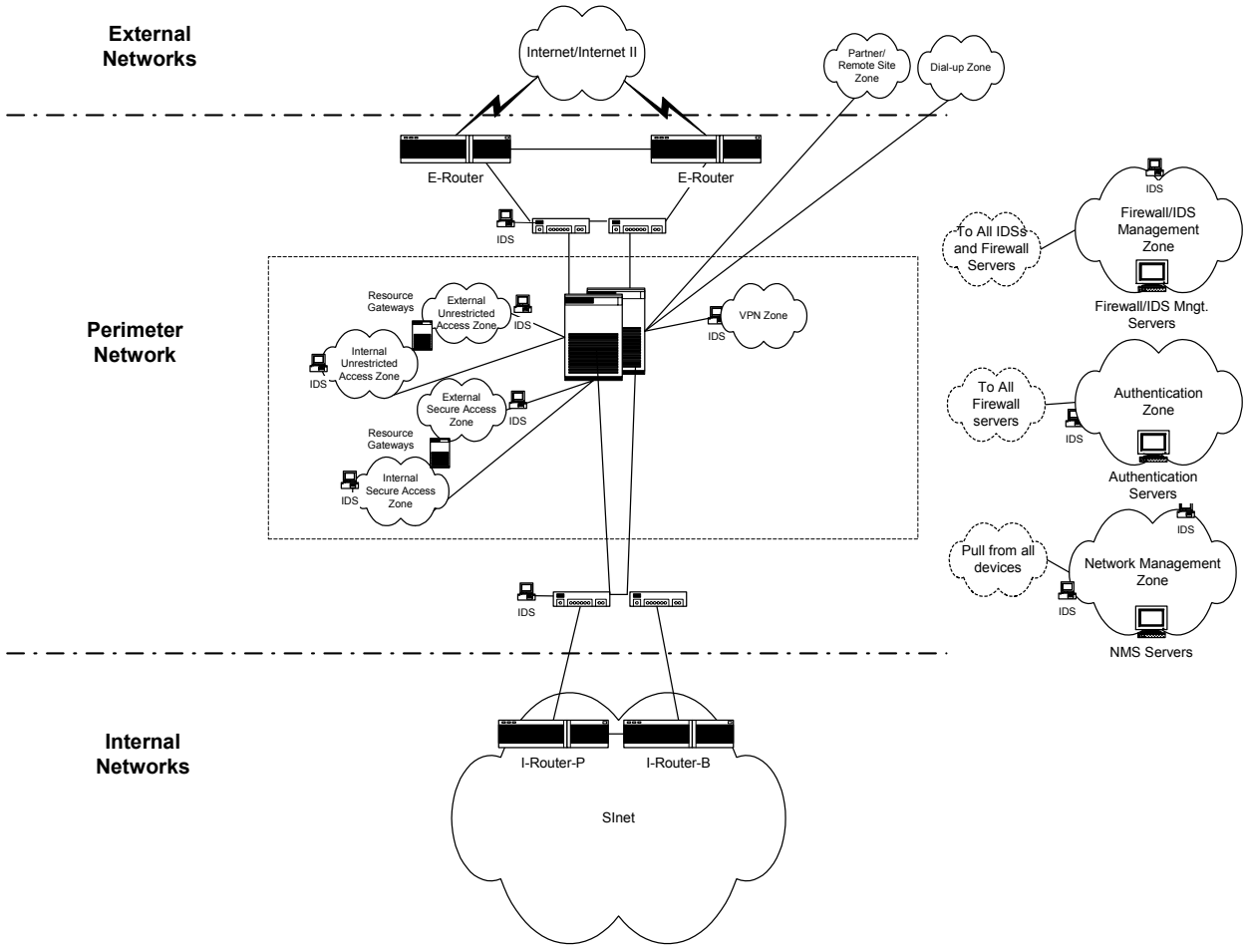

Figure 3-6. Security Architecture for Internet Access at SI

Assuming the security infrastructure is in place, the WebLogic Application Servers and Oracle database servers for EBCA/BCAC can be both put in a DMZ (DMZ-2 in Figure 3-4). Since the data are not considered sensitive, the database servers do not have to be in the internal SINet, to reduce un-necessary access to the internal network.

ServerLock (see http://www.watchguard.com/products/serverlock.asp) product will be used to lock down the servers in the DMZ zone. Running the web component on a SI Solaris Unix server with security software such as ServerLock in a DMZ with firewall and Intrusion detection will reduce security risk. Placing the application servers in the DMZ with security software such as ServerLock will limit potential compromise by external hackers within the DMZ network.

In addition, OS and database patches shall be applied on time, proper firewall rules shall be applied, and IDS shall be used actively to monitor these servers/the network.

3.4.3 Availability, Scalability, and Performance

There are several levels of redundancy which combined to provide availability, scalability, and performance.

The load balancers provide network level redundancy and load balancing. The user load is balanced at the network level first, so the traffic will not be concentrated on one server.

Application server clustering provided by the WebLogic Application Server provides failover capability. If one server in the cluster is down, the user sessions on that server can be transparently switched over to another server. Application server clustering also allows for new servers to be added to the cluster and shoulder part of the users' requests. This provides the scalability that allows the system to take on more users without bringing down the performance.

The redundancy in LDAP servers maintains availability to registered users in case one of them is down. If there is only one LDAP server, and it is down, then no registered user can log on to the server. If possible, the system can use the SI enterprise LDAP servers in the DMZ zone. The server cluster for the databases serves similar purpose. This architecture eliminates any single point of failure.

3.4.4 Storage

A storage area network (SAN) with RAID-based storage devices may be used to store both databases and file system-based data for the EBCA/BCAC. Internal or external RAID configurations may be used for servers inside DMZ zones. RAID-based storage can provide redundancies and fault-tolerant capability. For example, in one disk drive in the RAID array fails, the information on other disk drives can be used to sufficiently reconstruct all the data contents.

When SAN is used, EBCA/BCAC is just one node on the SAN, so the data is loaded from disks local to the system. SAN provides a robust solution for data backup and redundancy. The data in this project is not considered sensitive or critical, SAN storage may not be necessary.

The storage component architecture design needs to be consistent with the SI enterprise management system (EMS) and the SI system administration, backup, and recovery architecture and infrastructure which will be identified in the Smithsonian Information Technology Plan. Any storage device should be implemented as a separate (but integrated) infrastructure component within the proposed Web application services infrastructure.

3.4.5 Backup and Recovery

Any system should be integrated with a Smithsonian enterprise backup and recovery practice to backup the database(s), data in flat file systems, and system files.

3.5 Hardware, Software, Tools, and Licenses

There are many COTS products that support the web-based application and database management functions necessary for the proposed system. Assuming the system security, network infrastructure, backup and recovery solution, and LDAP are in place, the focus of the EBCA/BCAC system is on the hardware and software to run the application components and support the DBMS. Development environment and tools are also an integral part of the development effort. In order to develop and test Web services, tool support to automate the creation and deployment of Web services is very important due to the complexity of the structure and details of the Web Service Description Language (WSDL) document and the UDDI registry.

The following list of COTS software serves as a representative example of products that could be used to implement the proposed system, most preferred by the TRM. (Individual components could change in an actual implementation.)

SOFTWARE	Purpose	Capability
Solaris Operating Systems	2 for EBCA/BCAC servers, 2 for database servers, 1 for development	High performance platform to support memory management, networking, threading, file systems, etc.
Watchguard ServerLock	Server protection for the 4 production servers (2 Web/APP, 2 DBMS)	Lock down OS, application, and associated contents to prevent harmful attack or operation errors from damaging the servers.
Oracle 9i Release 2 and Oracle Text	EBCA/BCAC DBMS, for production and development.	Store and update XML data and application related information. Support indexing and query on XML data. Provide JDBC 2.0 driver to allow Java applications to use DBMS programmatically.
BEA WebLogic Application Server cluster version	EBCA/BCAC application servers and web servers, production and development.	Support standard J2EE application and Web services with UDDI registry.
BEA WebLogic Workshop	Development tool for Web Services.	Support development and deployment of Web services.
iPlanet LDAP server	To store user/visitor profile for the EBCA/BCAC	Support user login to update data.
Microsoft Windows 2000	Development desktops	Development platform to provide development environments with all development tools.
Borland JBuilder Studio	Development IDE	Integrated development environment to support J2EE coding and debugging.
RationalRose	Development modeling tool	Object-oriented design of the application components.
Rational ClearQuest	Defect tracking	Report defects during testing or operation;

SOFTWARE	Purpose	Capability
		track the defect through the bug-fixing process.
Macromedia DreamWeaver	User interface design tool	Design the visual user interface.
Rational ClearCase	Configuration management tool	Provide version control of all development artifacts such as design documents, models, code, etc.

Table 3-1. COTS Software Packages

HARDARE	Purpose	Components to Deploy
Two Sun E450 Servers	Each server hosts the Production Web server and application server for EBCA/BCAC	Solaris OS. Presentation Logic, Business Logic, Persistence Logic, Web services, WebLogic Application Server and Web Server, ServerLock
Two Sun E450 Servers	Production Database Servers. One server may be sufficient since high availability and throughput is highly critical.	Solaris OS. Oracle 9i Release 2 DBMS, OracleText, ServerLock
One Sun E450 Servers	Development server.	Solaris OS. Oracle 9i Release 2 DBMS, OracleText. ClearCase and ClearQuest.
Two Windows 2000 based Servers	To support user authentication and authorization through LDAP.	iPlanet LDAP servers.
Several Windows-based desktop PC	Application development	Windows 2000. WebLogic Application Server for development, all development and testing tools, and WebLogic Workshop.

Table 3-2. COTS Hardware

Product	Pricing Information
BEA WebLogic Application Server cluster version (3)	$17k per CPU (4 CPU each). May be free if OCIO can host the application.
BEA WebLogic Application Server development license (1)	$2000 per development license
Oracle 9i Enterprise Edition	Processor perpetual $40k
RationalRose, Professional Java Edition	Floating license: $3490 plus $698 for support
Five Sun E450 Servers	$30k each, with 4 CUPs, 4 GB memory. Large storage separately priced.
Two Windows 2000 based Servers	$20k each, with 4 CPU, 4GB memory
Several Windows-based desktop PC	$2k each, 1 CPU, 512M memory

Table 3-3. COTS Pricing

The commercial vendors and products: Solaris, SunCluster, Oracle 9i, BEA WebLogic Application Server, WatchGuard ServerLock, and iPlanet LDAP server were used as examples here because they are industry leaders, are consistent with the SI Technical Reference Model (TRM), and, in many cases, are the same products and technologies being implemented to support other related Smithsonian projects. Standardizing on products across SI would provide interoperability, scalability, reliability, and procurement, and engineering and operations support and maintenance cost and effectiveness advantages.

3.6 Implementation Strategy and Resource Requirement

Based on the requirements, implementation can be divided into the following five phases:

I. Prototype an end-to-end scenario based on canned schema and data to gain valuable insight into XML document storage, manipulation, presentation, with Oracle9i and WebLogic Application Server. Validate XML schema and linking mechanisms.
II. EBCA: storage, indexing, search, and basic linking management
III. BCAC: contribution management
IV. Web Services interface with GBIF and other partner systems
V. Wireless, PDA, and Kiosks support

Phase I can reduce technological risk and implementation risk significantly. It also provides feedback to improve and mature the schemas, as well as improve the requirement for re-keying.

Creating and maintaining links to outside system has very big operational impact. At what level the links will be defined, and how the links are maintained effectively have major development and operation impact. To reduce the risk, a prototyping effort is recommended to study to what extent the links should be supported, and the major operational issues on creating and maintaining these links.

Phase II solves the problem of XML document storage, indexing, search, and basic linking including jump-off links to other sites, or simple HTTP request links with a few key values. The XML schemas for EBCA including descriptions for taxonomy, MARC, and TEI Lite, should be defined in this phase. The scanning, re-keying, and XML-encoding of the selected catalogs will be done.

Phase III provides the functionality for BCAC and linking to the complete set of initial partners through a variety of linking mechanisms such as HTTP jump-off links, simple HTTP requests, EMu interface, etc.

Phase IV will make use of GBIF and accommodate requirements from GBIF, as well as provide Web services to any potential partners. To integrate with partner systems through Web services, the partners need to provide the services. That puts a demand in the partners to support Web services. The efforts and issues involved in the process of integrating with partner systems

through Web services have not been fully studied. Therefore a second prototyping effort is recommended to study the efforts and issues related to using Web services. This prototyping effort will also provide significant insight into the implementation of Web services.

Phase V extends the mode of access to the system to become pervasive computing to reach out to the widest research and education communities. This phase can be built together with Phase IV.

3.6.1 Life-cycle Management and Testing Automation

Iterative incremental development cycle is recommended based on the need of the project and best practice of the industry. To deliver quality software products in an iterative manner, testing early and testing frequently is key, and effective Configuration Management is of great importance for iterative development.

Tools

JUnit (www.junit.org) is a freeware tool that cab be used to build a set of automated test cases to do unit testing whenever the code has been changed. It is a regression testing framework written by Erich Gamma and Kent Beck. It is used by the developer who implements unit tests in Java.The basic idea is when a developer writes a Java class, he/she also writes the testing code to test each method in the class with JUnit. When that class is changed, the test can be re-run (regression) to make sure the change has not broken other part of the code which has not been changed. Ideally, if a set of test cases have been created to cover 100% of the code, any change to the code can be verified if the change has broken other code by running all the test cases again automated with JUnit.

Load testing tools such as LoadRunner from Mercury Interactive or TestStudio from Rational Corporation can help to verify the system performance by simulating the expected user load against the system.

Rational Suite TestStudio (http://www.rational.com/products/tstudio/index.jsp) automates functional, performance and reliability testing of complex applications and architectures. It provides testers with built-in process guidance and mentoring; integrated defect management; runtime analysis; a centralized test management tool; and ensures that the entire team works with the same data and metrics.

A competing tool from another best player in testing tools is LoadRunner from Mercury Interactive. LoadRunner (http://www-heva.mercuryinteractive.com/products/loadrunner/) is a load testing tool that predicts system behavior and performance. It exercises an entire enterprise infrastructure by emulating thousands of users and employs performance monitors to identify and isolate problems. By using LoadRunner, organizations can minimize testing cycles, optimize performance and accelerate deployment.

Rational (www.rational.com) ClearCase is one of the best CM tools in the market which provide version control of all development artifacts such as design documents, code, test case, etc. Rational ClearQuest is one of the best defect tracking tools to log a system defect and track the fixing cycle of the defect. Rational Rose is a market leading object modeling tool which supports

Unified Modeling Language (UML) and allow software designers to visually describe the object models for a system. It can also convert part of an object model into a data model.

3.7 Architectural Risks

This section identifies the architectural risks that can potentially impair the development of a successful system to fulfill the system requirements. Each risk has different level of criticality, and may need different mitigation strategy. Although the previous sections have noted some of the risks and also discussed some strategies, this section includes a more complete list of the risks for this project.

Following is the list of architectural risks:

- The XML schema, which captures the conceptual model of the taxonomic literature, may be difficult to implement. The granularity of the XML documents, i.e., the unit of contents to be stored and retrieved as XML documents, need to be defined and prototype based on the schema should be developed to verify the conceptual model.
- The richness and complexity of the conceptual model entails taxonomic data, bibliographic and citation data, and museum collections data. Whether the XML schema captures the correct relationships to allow for effective representation and navigation of the contents need to be verified through prototyping.
- While Web service is the technology of choice for integrating with partner systems and publishing services, partner systems may not be ready or have the technical support for Web services.
- The performance implication of storing XML documents in their entirety in a DBMS or breaking down the XML documents and storing the pieces in a DBMS need to be clarified.
- The system need to support indexing and search of large amount of data in a flexible way. The recommended product Oracle Text may not be capable enough to fulfill the indexing and searching requirements. Evaluation of alternative products such as Verity (www.verity.com) or EMu (www.kesoftware.com) may be needed.
- While a Web browser based thin-client is preferred, thick-client application may be needed for efficient data entry.
- The mechanism for linking to external data source may not be flexible enough to accommodate the variety of links, or may be too ad hoc to be maintained effectively or efficiently.

SECTION 4
REFERENCES

4 SECTION REFERENCES

The following sources of information were referenced in the white paper:

1 Technical Requirements for The Electronic Biologia Centrali-Americana, November 2, 2002
2 Native XML Databases and Relational Databases, eXcelon, http://www.exln.com.
3 Can schema agnostic databases boost Web services? Application Development Trends, December 12, 2002. http://www.adtmag.com/article.asp?id=6498.
4 XML-Grounded DBMSs: Short-Term Fad or Viable Market? Research Note, M-15-6059, T. Friedman, the Gartner Group, 2 May 2002.
5 When XML and Databases Collide. Research Note, TU-14-0694, T. Friedman, the Gartner Group, 3 October 2001.
6 Oracle XML DB, Technical White Paper, Oracle Corporation, January 2002
7 Oracle XML DB: Using XML Content and Data, Victor Votsch and Mark Walter, Seybold Consulting, Group, March 2002.
8 Oracle Text, An Oracle Technical White Paper, March 2002, Oracle Corporation
9 Oracle Text: Features Overview, March 2002, Oracle Corporation
10 Portlets Help 'Tame' the Delivery of Web Services, Research Note, SPA-18-3897, G. Phifer, the Gartner Group, 25 October 2002.
11 Proposal to the Atherton Seidell Endowment for Digitizing and Disseminating the Biologia Centrali-Americana. Smithsonian Institution Libraries, May 6, 2002.
12 Section 508 Standard. http://www.webaim.org/standards/508/.
13 BCA Information Analysis Report, Betty Harvey, December 15, 2002.
14 Service-Oriented Architectures Foster Real-Time Capability, Research Note COM-18-9401, W. Andrews, etc., the Gartner Group, 26 November, 2002.
15 Building Web Services with Java, Making Sense of XML, SOAP, WSDL, and UDDI, by Steve Graham, etc., SAMS Publishing, ISBN 0-672-32181-5.

Appendix A
Web Services

A.1 Overview of Web Services

Web services have gained a lot market acceptance recently and the Gartner Group predicts that by 2004 most major enterprises will use Web services. A Web service is a unit of application logic providing data and services to other applications. Applications access Web services via ubiquitous Web protocols and data formats such as Hypertext Transport Protocol (HTTP), eXtensible Markup Language (XML), and Simple Object Access Protocol (SOAP), without the need to worry about how each Web service is implemented. It uses Web Service Description Language (WSDL) to describe a service in a registry and Universal Description Discovery and Integration (UDDI) to facilitate discovery of services. It combines the best aspects of component-based development and the Web. In summary, it is a platform and implementation independent software component that can be:

- Described using a service description language
- Published to a registry of services
- Discovered through a standard mechanism (at runtime or design time)
- Invoked through a declared API, usually over a network, and
- Composed with other services.

Most Web services technologies, when applied to application integration problem, has a pattern called service-oriented architecture (SOA), as depicted in the following diagram.

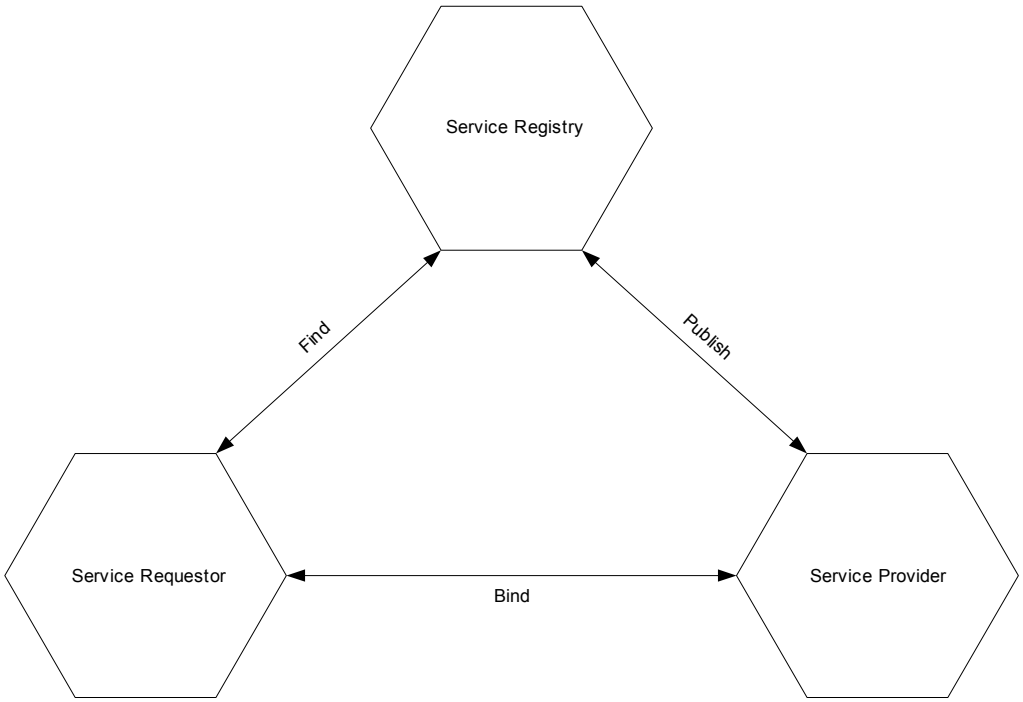

Figure A-1 Service-Oriented Architecture (SOA)

A service provider creates a service description, publishes it to one or more service registries, and receives Web service invocation messages from one or more service requestors. A service requestor finds a service description from a service registry and uses that description to bind to or invoke the Web service hosted by the specified service provider. A service registry advertises Web service descriptions published to it by service providers and allows service requestors to search the collection of service descriptions contained within the service registry.

The following diagram from the Gartner Group depicts the technologies forming the Web Services Stack.

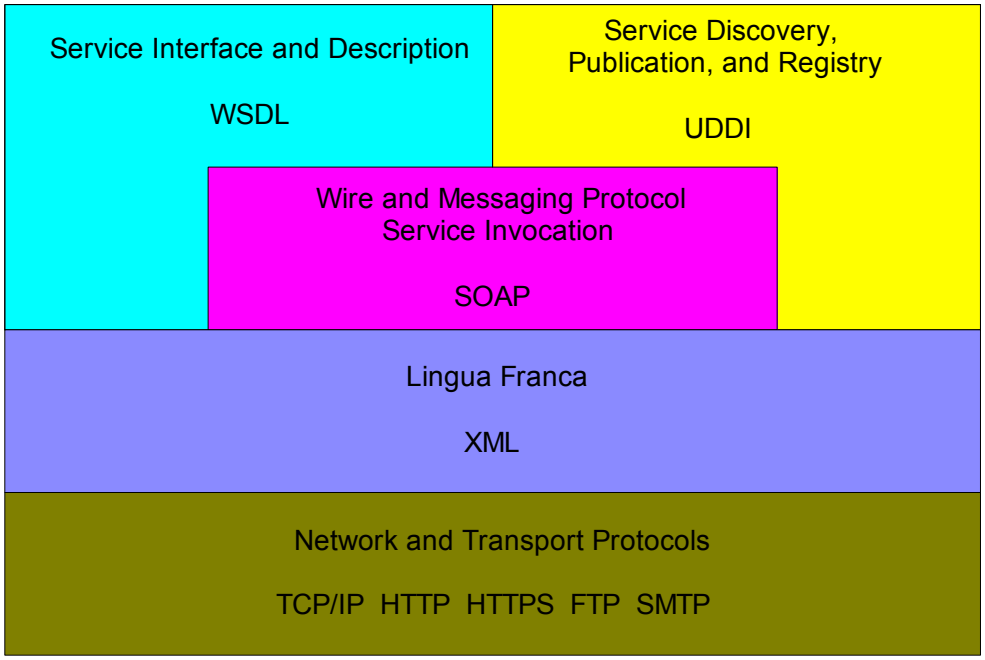

Figure A-2 Web Services Stack

WSDL provides a standard XML vocabulary and document format that describes Web services offered in a service registry. The public UDDI Registry – with sites managed by IBM, Microsoft, and Hewlett-Packard and kept in sync with each other – uses WSDL for publishing and dynamic service discovery. Private UDDI registries may or may not support WSDL.

Both WSDL and UDDI show promise in solving inter-organizational integration problems. However, according to Gartner, WSDL is still an immature and relatively incomplete technology with a variety of issues that need to be resolved, and not expected to be in business-critical systems at least until 2003. Through 2004, private UDDI implementations, which are managed by major corporations, industry association, or a group of business partners, and control who have access to them, will be adopted more quickly than public ones.

Development Web service need strong tool support. Major J2EE development environments such as BEA WebLogic Workshop provide tools for developing and deploying Web services.

While Web services or any other XML based integration technologies provide dynamic binding to services so the integrated partners are decoupled and thus create a flexible and robust partner network, there is overhead in transporting, parsing and processing XML messages. Internal to an application, the components work with each other tightly and frequently. If XML messages are used as communication format across layers of an application, the overhead of XML processing can cause degradation to the system performance, especially under heavy load. Systems with tight performance requirement should make sure this overhead is acceptable. In general, tightly coupled components should communicate through RMI or regular local Java method invocation, while loosely coupled components can base their interfaces on standard format such as XML, to hide difference in platforms or languages.

Appendix B
Databases for XML Documents

B.1 XML Database Technologies Survey

The Gartner Group classifies XML related databases and repositories into five categories (see TU-14-0694): XML-Wrapped DBMSes, XML-Grounded DBMSes, XML-Aware Repositories, XML Servers/Portals, and XML Query/Search Engines. The characteristics and applicability of each one are summarized in the following table:

XML-DB Type	Characteristic	Usage
XML-Wrapped DBMS	Map XML into columns and rows Use SQL query and index to support search	Data-centric XML serves as transport mechanism for data XML document structure is static
XML-Grounded DBMS	Proprietary XML structure is stored in its entirety Fidelity of XML document is maintained. Support new structure without modifying database schema. Use standard XML interface such as DOM or XPath Support indexing the elements and attributes Limited support for transaction and operation	Document-centric Web-publishing or general document management
XML-Aware Repository	Store in native format Support dynamic and less structured format Fully indexed and searchable Lack transaction management and admin tools Content is stored in chunks and can be put together into a complete document.	Document-centric Web-publishing or general document management
XML Servers/Portals	XML as transport and presentation mechanism for universal data access (files, DBMS, etc.) Predefined mapping to different data sources and formats No centralized DB. Integrated and accessed on the fly.	highly data-centric for data access and integration Web-based data access and e-commerce.
XML Query/Search Engine	Not a database. No storage or management of data. Extensive search on XML documents	Where search is sole functionality.

Table B-1 XML Data Storage and Retrieval Technologies

In Gartner's opinion (see M-15-6059), native XML-grounded DBMS products have only gained limited market traction and vendors' value proposition vs. RDBMS will degrade over the next

two years. Major RDBMS vendors are adding deeper XML support in their products. Unless an enterprise has a clearly defined problem that cannot be sufficiently addressed by its preferred RDBMS vendor, it may not want to invest in native XML-grounded DBMS.

Even for vendors specialized in XML database technology, they only see this technology fit in niche market. For example, Jaenicke, XML evangelist for Burlington, Mass.-based Excelon, the vendor for the XIS XML DBMS, points to a path where a combination of RDBMS and XML DBMS work side by side to solve enterprise data problems. She explained, "A backend database such as Oracle is very good for durable data, fixed schema, highly typed, highly normalized data that you're going to store for the rest of your life. An XML database, which is schema agnostic, is much better tuned to manage active data. And what we mean by active data is data that is being actively updated, actively audited, actively processed."

B.2 Oracle9i Database Release 2 XML Support

With release 9.2, Oracle significantly adds to XML support by introducing a high-performance XML storage and retrieval technology that fully absorbs the W3C XML data model into the Oracle server, and provides new standard access methods for navigating and querying XML. This set of features in the Oracle server is called the **Oracle XML DB**. With it, you get all the advantages of relational database technology and XML technology at the same time.

Following are some major XML features in Oracle XML DB:

- Store and manage both structured and unstructured data, under a standard W3C XML data model.
- Complete transparency and interchangeability between the XML and SQL metaphors. Better management of unstructured XML data through piecewise updates, indexing, search, multiple views on the data, managing intra-document and inter-document relationships and so on.
- Repository functionality inside the Oracle database – foldering, access control, FTP and WebDAV protocol support with versioning – thus enabling applications to retain the file abstraction when manipulating XML data brought into Oracle.
- Automatic identification and parsing of XML schemas
- Navigation that adheres to the XPath standard by W3C. Developers can write queries that specify an XPath location, and update individual elements or attributes of XML documents, without checking out the rest of the document.
- Very high performance and scalability for XML operations
- Use XMLType to store data either in LOB with Text index to maintain the original accuracy of the XML document, or object-relational storage with B-Tree index to maintain DOM fidelity and achieve excellent DML perfromance.
- Oracle Text full-text indexing engine indexes XML text documents stored in the Oracle9i Database.

From Seybold Consulting:

“The technologies that must be part of this next-generation infrastructure include an integrated repository for data and content; multiple, nonproprietary methods of access; and support for the

Web family of standards (HTTP, FTP, WebDAV, SOAP, etc.) to handle content and data exchange among servers and client productivity tools. The integrated and optimized XML handling provided by Oracle XML DB addresses the need for a standards-based, scalable infrastructure to support increasingly complex uses of XML."

B.3 Oracle9i Database Release 2 Search Support

Oracle9i Database Standard and Enterprise Edition include the Oracle Text product, which provides integrated full-text retrieval technology.

Oracle Text uses standard SQL to index, search, and analyze text and documents stored in the Oracle database, in files, and on the Web. Oracle Text can perform linguistic analysis on documents; search text using a variety of strategies including keyword searching, context queries, Boolean operations, pattern matching, mixed thematic queries, HTML/XML section searching, etc. Oracle Text excels at mixed queries, i.e. those that involve text as well as structured relational attributes.

For XML documents, Oracle Text offers advanced within-section searching:

- Search for a document that contains search terms within an XML-tagged section
- Nested section search
- Search within attribute values
- Map multiple tags to the same name
- Path searching, section existence (a subset of XPath)
- Ability to search for content and structure at the same time

Oracle Text can render search results in various formats including unformatted text, HTML with term highlighting, and original document format. Oracle Text supports multiple languages and uses advanced relevance-ranking technology to improve search quality.

The Knowledge Base in Oracle Text can be extended to support search by thesaurus. This feature can be used to support common name search in EBCA/BCAC.