IBM Lotus Extended Search Technology & Architecture

Highlights

The lack of a centralized index reduces storage requirements and costs

The use of native data systems preserves data integrity

The ability to search multiple sources in parallel results in faster response times and consolidated search results

An easily scalable architecture supports horizontal and vertical growth

A multi-layered security model supports enterprise-specific rules

National language support can help capture the global marketplace

Whether you are looking to provide a single point of communication via a “corporate portal,” entering the world of e-commerce, or implementing an enterprise-wide knowledge management initiative, you are confronted with the challenge of information access and integration.

You are intrigued with today’s web search engines, at their ability to search a world’s worth of information in seconds, and wonder why such a technology cannot be applied in your organization.

To solve this problem, Extended Search employs a strategy that uses your existing data management software to access all of your organization’s information, wherever it may be located, with no need to create a central index.

By presenting a common view of the information domain, Extended Search shields users from the complexities of dealing with so many different search interfaces and the logistics required to access backend systems. Users use a familiar web interface, and enter a single query, to retrieve information from throughout the enterprise.
What is Extended Search?

Extended Search, as the name implies, is a product that "extends" the standard capabilities of other search products by allowing you to search and retrieve all types of information, not just data that is managed by the hosting software.

Using a standard web browser, you can easily and quickly locate and review information contained in thousands of data repositories that are spread throughout the organization. These repositories may be of varied content and structure, and they may be geographically dispersed throughout the world.

With a single query, you can search internal and external web sites, full text indexes, Microsoft indices, document management systems, e-mail systems, file systems, relational databases, LDAP directories, and the full complement of Lotus databases — simultaneously.

Think of Extended Search as a brokered agent working on your behalf: it searches each of the data stores in their native form, yet returns results in a single aggregated results set.

Extended Search shields you from this data source diversity. You interact with a single function-rich interface that transparently distributes your search to potentially thousands of data stores.

Results, which can be presented in a variety of ways, are typically returned in a web page; URLs enable you to go directly to documents of interest or download attachments.

You can save and share searches and results with other users. By scheduling searches to run on a repeated basis, you can keep current with changes in your knowledge banks, tap realms of expertise that are spread throughout your enterprise, and ensure that timely and relevant information is communicated to the people who need it when they need it.

With Extended Search, you experience more than what you have come to expect from a typical search application. You discover that your reach has been extended further into the enterprise, and that you can access information not easily attainable before.
Extended Search Technology

Our Approach

Almost all search solutions involve the use of an index that catalogs the information to be searched. Without an index, the time to perform the search would be greatly exaggerated – much like trying to find a book in a library without a card catalog. But while most search solutions require you to re-index your information into a new index, Extended Search leverages your current data management investment.

The information in your enterprise probably exists in many shapes and forms. It is distributed throughout the enterprise and managed by the most appropriate software for the task at hand. For example, you may use a SQL application to tap into relational databases or a Domino application to access Lotus Notes databases.

Extended Search taps into this mesh of data sources and modes of data access, and treats the existing collection as one virtual index with a single point of entry.

Users need not know specific details about any of the backend data sources. Because you have complete control over the front end, you can present users with point-and-click search options and a means to type simple search words. You can also give experienced users free access to formulating complex queries in a syntactically precise query language.

Regardless of how a query gets submitted, Extended Search translates it into the native search languages of the target data sources and uses search and retrieval methods that are native to each data source to return results.

Not all backend data sources are created equal, however, and they can sport widely differing capabilities (such as full text indexes vs. relational databases vs. e-mail systems).

Extended Search attempts to minimize these disparities whenever possible and achieve more than the least common denominator effect.

For example, Extended Search might combine two or more operations against one data source to achieve what can be done in a single operation for a different source (more on this later).

This approach enables users to search multiple data sources, of differing types and formats, in parallel with a single query. The design offers several advantages:

- Mechanisms of the native database or search engine preserve the integrity and currency of the information.
- Because the duplication of data into a centralized index is not necessary, it reduces overall storage requirements and eliminates the cost of re-indexing resources.
- Data remains in close proximity to the work that is being performed.
- The respective data sources and search engines can perform operations in parallel, thereby improving search response times.
This approach also offers greater scalability than those built around a centralized index. As the number of documents in a domain grows, so does the time it takes to index the information. It is not uncommon for the indexing of large domains to take several days or weeks and, at some point, become impractical.

In this situation, the power of Extended Search can complement the depth of the index: first, stabilize the index at its maximum capacity, and then use Extended Search to search the index in conjunction with other data stores. With this approach, you have the option of indexing only what you feel needs to be indexed while using Extended Search to search both indexed and non-indexed sources.

Connecting new sources to the search domain results in little overhead being incurred by Extended Search, and it is insignificant when compared to the expense of the actual search operations themselves. Because the search is executed on one or more machines that already exist in your network – using existing search operations – the incremental cost of connecting new sources is negligible.

Of course, newly connected data sources, not easily accessible before, may experience more computer usage as a result of their increased search activity.

Providing a Common Ground

The approach just described relies on a strategy for accessing all of your information wherever it happens to be located by using your existing data management software. With this strategy comes a greater likelihood of encountering systems with different capabilities. There are general areas in which most systems vary with respect to search and retrieval:

- Search language – how the query is expressed
- Data model support – how the data is organized, related, and represented

Application program interface (API) support – how information access is performed

Extended Search offers a common ground that enables system implementers to work in a complementary manner within these areas.

Note: Security is an obvious differential between systems and was intentionally excluded from the list. It will be covered in greater detail in a later section.

Common Search Language

Nearly all data management systems employ a grammar or query language of some kind to express the criteria of a search. These grammars can vary widely depending on the structure and composition of the data.

In free text systems such as the web, for example, a search is generally expressed as a list of keywords. Additional notations are used to express boolean conditions (and, or, not) or positional information, such as specific words that must occur within the same sentence or paragraph.

If the data is highly codified and structured, the grammar may be more parametric and support fielded operations (for example, the value of the Quantity field is greater than 100).

Contrast the simplicity of the first example below, which illustrates a typical keyword query specified in a web search engine interface, with the syntactically precise format of a query expressed in Structured Query Language (SQL).

Sample Web Query:

RUN + BASEBALL - RACE - STOCKINGS

Sample Structured Query Language (SQL) Query:

```sql
SELECT * FROM products WHERE Quantity > 100
```
Sample SQL Query:
WHERE ( (BODY LIKE %RUN%) 
  AND (BODY LIKE %BASEBALL%) 
  AND NOT (BODY LIKE %RUN%) 
  AND NOT (BODY LIKE STOCKINGS%)
)

Generalized Query Language

Clearly, it is impractical for a user to know the syntax of the entire set of search grammars used in an extended search. It is much more practical to let the user express the query in a single common search language. Extended Search offers just such a language, which it refers to as the Generalized Query Language (GQL).

GQL represents the culmination of years of experience with numerous types of grammars. Think of it as the superset of search grammars from which most queries can be expressed.

All Extended Search queries are expressed internally in GQL format (with the exception of “pass through” queries which will be discussed shortly).

When the GQL expression arrives at a specific data source, Extended Search transforms it into the native query language of that source. This translation process maps the lexical elements of GQL to equivalent elements of the native grammar.

While the fullness of the GQL grammar is supported by all components of Extended Search, not all backend search engines are capable of supporting all elements of the GQL grammar. When Extended Search encounters these disparities, it tries to compensate by combining two or more native operations to achieve the same effect.

Only after Extended Search has exhausted all alternative methods will it ignore the non-supported portion of the expression. Under these circumstances, you can instead pass through a query that does not get translated.

Pass Through Queries

Expressions identified as “pass through” queries skip the translation step and run “as is” against the target data source. This allows you to specify a query in the native grammar of the target data source. You must take care to ensure that pass through queries are directed only towards those data sources that can understand and execute the query.

Pass through queries, when combined with GQL, provide all of the convenience and flexibility you need in a search grammar. You could, for example, issue a GQL query against hundreds of data sources of various types and then, based on the information returned, issue a specific pass through query to retrieve additional meta data about each item in the result set.

Common Data Model

Just as search grammars can vary with each dissimilar backend system, so can the data models used to organize and store information. The data model used by a particular data management system is typically designed for the class of applications it serves. This determines the amount of structure and granularity found in its information.

For example, free text systems tend to use a loosely structured document model with low data granularity.

A document may consist of a few fields (such as title, author, and body) but its text remains free in form and unstructured.

By comparison, information can be highly structured, such as that found in relational databases. Here, data is organized into rows and columns that can be related in any number of ways, which results in high data granularity.

Extended Search has normalized this diverse set of data models into a single model, one that is easy to understand and usable by most search applications. The search application designer need only contend with this one conceptual model of the data and not be confounded by the many.
The illustration to the left presents the Extended Search common data model. As you can see, it is closely aligned with a traditional document model. In this model, a document can be made up of one or more fields. A collection of one or more documents comprises a searchable source. One or more sources can be grouped into searchable categories. Categories can further be assigned to a search application (for example, a personnel vs. financial application).

The model itself does not attempt to achieve a full union of all the backend data models but rather to provide a flexible form into which all models can map most of their concepts. It is an easy process to define this assignment to Extended Search, and there are many wizards available to assist in the task.

**Mapped Fields**

Typically a source maps to a database in the conventional sense but can just as easily be mapped to a web site, a directory in a file system, or a node in an LDAP hierarchy. Likewise, a document maps conveniently to those systems that are text based but could just as easily represent an instance of data, such as a row in the table of a relational database.

A common problem encountered when relating data sources of different types is the mismatch in field labels. For example, an author’s name might be labeled `AUTH_NAME` in one data source and `CREATOR` in another, and yet be represented as three fields (such as first name, middle initial, and last name) in another.

An important feature of the common data model is the ability to define **mapped fields**. A mapped field is a composite of one or more native fields.

To resolve the ambiguity in the previous example, you could define a single mapped field with the label `AUTHOR`. You could then map this field to one or more native fields in each of the data sources that support the semantic of author’s name.

The benefits of this feature are compelling when used in the search expression. A user needs only to specify the mapped field in the query. Extended Search will automatically associate the mapped field to the correct native fields on the backend. This approach greatly simplifies the query expression, and provides greater benefit as the number of data sources increases.

Not only do mapped fields help in simplifying the search expression, they can also be used to simplify the processing of search results.

Suppose, for example, that an application is to display a barometer that is calibrated to reflect the age of a document (that is, more recent documents would have a higher barometer reading). If the “date created” field were labeled differently in each of the backend sources then the application would need an exhaustive set of conditional statements, one for each discrete date field name.

But, with mapped fields, only one mapped “date created” field needs to be identified for retrieval and subsequently tested. By referencing this single mapped field, the result would contain the correct date value as it correlates to the source’s native date value.

**Note that dates can vary by more than just name but also by format (such as mmdyy vs. yyyyymmdd). Date translation macros are available to assist in the conversion of dates to a common format.**
Of course there are times when diversity in the results set is desirable and can be easily supported by the common data model. For example, if the result came from a personnel record in an LDAP directory, you might want to return the person’s name, job title, and contact information.

On the other hand, if the result came from an e-mail system, you might want to return the date, subject, and author. These semantically different data fields can be selectively retrieved using their native source field identifiers. When the native field label is too cryptic for display purposes (for example, $Doc_Abstract), then a mapped field could again be used as an alias for the native field name (for example, Document Abstract).

**Common API**

So far we’ve talked about the benefits of a common data model and a generalized query language – both necessary components of a heterogeneous search strategy. But the question remains: How do I interact with all of these different backend systems to search and retrieve their managed information?

The API to each backend system can vary dramatically by calling methodology, syntax, semantics, and programming language.

Extended Search offers a single interface executing searches and retrieving results. Herein lies the strength of Extended Search and one of the greatest benefits to the application designer.

The functions issued through this common interface are translated automatically by the system into the native methods of the backend system – much like the process of translating GQL into the native search grammar.

Extended Search caters to each of the different mechanisms on the backend in the best way possible to achieve optimum parity between all data sources while respecting the operating policies of each native search API. By using the published API of the hosting software, the integrity and security of the data source is preserved.

To achieve this communicability, an Extended Search link module is created for each type of backend data source and is used to encapsulate all of the native API calls required for that type.

Extended Search comes with a broad set of link modules that connect you to most of the industry’s common data management systems. If your data management system is not part of this standard set, you can develop a custom link module by using the provided link toolkit.

An agent-based technology is used to apply these link modules to their respective data sources. Agents work on behalf of your brokered search and are generally transparent (links and agents will be discussed in more detail later on). From a designer’s perspective, you are interfacing with one searchable backend system through a single API.

**API Modes**

The Extended Search common API is available in two forms.

The first and most flexible is the Java bean interface. With this method the beans are combined with the power of the Java programming language to develop a broad range of simple to highly complex and specialized search and retrieval applications.

The product demonstrates the use of these beans in a sample web application that uses JavaServer Pages (JSPs).

For those organizations not skilled in the Java programming language, the product provides a set of HTML-like tags that allow you to embed Extended Search functionality into new or existing web pages.
These easy to use tags empower your web master and enable users to enter queries that specify a variety of search options. You can embed Extended Search tags anywhere within a web page – they do not interfere with the surrounding HTML tags.

**Search Results**

Extended Search provides great flexibility with regard to how search results get returned. As part of configuring the system, you can determine which fields a user is allowed to query, view on the results page, and retrieve from the data source.

The user is presented with a single, consolidated page containing results from multiple sources. The list is pre-pruned for relevancy, thus ensuring that the user sees the best matches first.

To retrieve a document, users typically select a result item (identified by a URL), and the web browser renders the content in accordance with the MIME type set for the document. For example, the browser might use Microsoft Word to render documents that have the file extension .doc).

In some cases, such as with documents stored in file systems or in ODBC databases, Extended Search will explicitly retrieve the requested document.

**Scalability**

The distributed component architecture of Extended Search offers the flexibility to scale a system according to changing requirements. It also allows the Extended Search components to be arranged in a topology that matches your environment, enabling you to blend IBM AIX, Sun Solaris, Windows 2000, and Windows NT platforms as needed.

The architecture supports vertical and horizontal scalability:

- Vertically, within a single Extended Search server, you can configure multiple instances of server processes to influence the number of simultaneous requests that the server can process.

- Horizontally, with multiple machines, you can set up additional Extended Search servers and additional web servers. For each Extended Search server, you can determine the types of server tasks you want to run. By having multiple servers, you can distribute and balance the processing load.

**National Language Support**

Extended Search is fully enabled for national language support. As installed, it is enabled to support 42 primary locales and 62 secondary locales, and it can be easily changed to include additional primary and secondary locales.

Designed to support a wide variety of data sources that exist on Windows and UNIX platforms, it allows data in these sources to be stored in a variety of code pages (note, however, that all data stored in a single data source must use the same code page).

The system also supports a variety of HTML-based search applications in different locales within one Extended Search installation. JSP-based search applications are limited only by the support provided by the web application server.
In Closure

To confront the challenge of information access and integration, search engines can play a prominent role. They are not, however, the total solution.

We talked about how re-indexing information that already exists has its limitations and may only be practical when applied to a subset of the information. This is actually the case with today’s web search engines whose coverage of the World Wide Web is actually twelve percent and declining.

The approach used by Extended Search offers a unique solution to this problem. It relies on a strategy that uses existing data management software to access all of an enterprise’s information wherever that information may be located.

This technology protects your investments in existing hardware and software and preserves the integrity and security of the data. Because the data is accessed where it is and not copied and re-indexed, search results remain as up-to-date and current as the backend dictates.

An important feature of this approach is the common ground it offers to end users. Users have no need to learn a complex query language, nor do they need to know anything about how to access the sources they want to search.

With Extended Search, users express their queries in a single language and use a single interface to search and retrieve information throughout the enterprise.

Extended Search offers a truly brokered approach to search and retrieval, much like a stock broker who handles transactions for buying and selling different company shares on your behalf.

The next section describes in greater detail this brokered architecture and shows how seamlessly you can merge it into your existing IT architecture.
The huge economic growth experienced toward the end of the century has been attributed to the advances made in information technology and to those companies that invested in it.

Because the technology was (and still is) changing at break neck speeds, companies frequently invested and then re-invested their IT budgets into ever evolving products to manage their information.

More often than not, the result has been islands of information distributed throughout the organization – highly specialized for the task at hand, but not easily accessible on an enterprise wide basis.

Even if a company were able to dictate a common IT policy for the enterprise, it could be thwarted by the single acquisition of a company that employs a different IT architecture.

Through its ability to access a highly diverse set of information sources where they are, with existing software, and within your IT infrastructure, the architecture of Extended Search is specially designed for such mixed environments. This section discusses the architecture of Extended Search in detail, and demonstrates how easily you can deploy Extended Search throughout your organization.

A Multi-Tiered Design

The Extended Search system employs a four-tiered architecture. Messages start from search applications in the first tier and proceed consecutively through subsequent tiers to the backend.

In most cases, the backend is a third-party data source to which Extended Search is connected but it can also be the Extended Search configuration database (CDB), a backend that is managed by DB2.
Message flows between the tiers can be divided into two basic categories:

- **Run Time messages**, shown above the dotted line in the preceding diagram, are messages usually issued by the user community to perform searches and retrieve documents.

- **Administrative messages**, shown below the dotted line, are issued by the Administrator and result in updates to the configuration database.

Run time messages can be submitted either through a standard web browser or a Lotus Notes client program. Administrative messages are always submitted through the Extended Search Administration interface.

The horizontal bars in the diagram indicate the consecutive components through which each message must flow during its journey from the first tier through the fourth tier and back again. Each of these components is described below, starting from the right side of the diagram and moving to the left.

### Links and Translators

Extended Search links are the software modules that encapsulate the native API calls for search and retrieval to a specific data management system. They contain all of the required data structures, programming objects, and procedural logic necessary to interface with the backend data system.

A link module is uniquely assembled to support (at a minimum) four callable methods that typically exist in all data management systems:

- Methods to connect to and disconnect from the host system
- Methods to search content and retrieve data from the system

The link module performs a null operation for those methods that are not supported by the backend source.

For example, a file system search does not support the concept of connecting and disconnecting.

Extended Search translators are the software modules responsible for translating the incoming GQL expression into the native search grammar of the backend data system. They, too, contain all of the required data structures, programming objects, and parsing logic necessary to generate a syntactically correct search expression.

In some cases, the same translator module applies to several different backend systems, as is the case for the SQL translator and the many varied systems that support the standard SQL grammar.

Extended Search comes with a broad set of link and translator modules that enable you to connect to most of the industry’s common data management systems. If your data system is not contained in this standard set, you can develop a custom link or translator module by using an easy-to-use toolkit provided with the product.

### Agents

Extended Search agents are programs that respond to search and retrieval operations targeted against a particular data source. The agent loads the appropriate link and translator modules when a request against a specific data source type is first made. The agent then calls upon these module libraries for translation (XLAT), connect, disconnect, search, and retrieval operations.

The following diagram illustrates the interaction of the agent with a given backend system.

For search operations, an agent will sort the results set by relevance rank and then truncate the set to the maximum number of hits, as specified in the original search request. This sorting and subsequent pruning of the list of hits is an important precursor to aggregation, which will be discussed shortly.
Agents can reside on the same machine as the data source (recommended) or use a data source’s remote APIs for access. More than one copy of an agent can run on a single computer to handle concurrent search and retrieval requests. An agent can be dedicated to a single data source, a group of sources of a particular type, or a range of sources that have a mixture of link types.

**Brokers**

Extended Search brokers are intermediary components that exist between the requestors of service and the agents that actually perform the service through the backend. They function as special purpose resource coordinators designed to manage the multitude of searches generated from a single request – as caused by a category search for example. The following diagram illustrates the functionality performed by an Extended Search broker.

A broker typically performs these tasks:

- Validates the request.
- Expands categories to obtain a list of the data sources available to the application and resolves the source addresses. (Label 1)
- Distributes queries to agents for efficient, parallel searching. (Label 2)
- Aggregates and optionally sorts search results that are returned by the various agents into a single search result set. (Label 3)
- Caches search results for subsequent paging operations. (Label 4)
- Issues requests to agents to retrieve source documents for the user (note that in most cases, the web browser uses the URL returned in the results list to retrieve the document).
- Honors timeouts and response options.

The degree of responsiveness can vary dramatically from a large set of backend systems contributing to a single request. Some data management systems respond faster than others, and some not at all – possibly due to out of service conditions. To account for this situation, brokers are designed to communicate asynchronously with their agents.
This design allows a broker to not be dedicated to any one particular backend data source, and it enables the user to assign a timeout value to the request. When a timeout threshold has been reached, the broker returns whatever results have been compiled up until that point.

Additional options let you control how the broker returns results. Two such options are to return the results when they are available or after they have been sorted.

If you specify the “When available” option, the broker will return the results in the order that the sources respond to the query. This approach provides a fast way to see the results of your search, but there is no guarantee that the first results you see will be the most relevant results.

If you specify the “Sorted” option, the Broker will collate all the results, sort them according to additional options you specify, and eliminate duplicate references before returning the results to you. This approach usually takes longer than obtaining results as they are available, but the results may be more relevant to your query.

**Multiple Brokers on Multiple Machines**

To support performance and scalability, a given Extended Search domain can contain multiple brokers set up on multiple machines.

This ability to establish a hierarchy of brokers, along with the ability to set up agents co-resident with the sources they support or to dedicate agents to particular sources or types of sources, provides Extended Search with endless flexibility with regard to changing and expanding environments.

Under a multiple broker schema, sources get partitioned across all of the brokers, a design that prevents any one broker from being overwhelmed.

For example:

In a single broker environment, a search that targets six dozen sources would result in 72 queries being sent to the remote machines and 72 sets of search results being returned to the broker.

If each result set contains the maximum number of results, most of the data will be discarded when the broker consolidates and aggregates the data for the list being returned to the requestor (the broker prunes the results and keeps only the top items, up to the maximum number allowed by the search application).

With multiple brokers, an entry broker sends a single message to brokers on remote machines. The remote brokers then split the message into multiple requests for the sources (fronted by agents) on their respective machines.

Instead of all result sets being returned to one broker, each broker consolidates, aggregates, and prunes the results returned by its agents, and then returns just a single list – containing the top hits – to the entry broker. The entry broker only needs to create a final results set from its own local sources and the consolidated lists returned by the remote brokers.

This design enhances overall performance (less bandwidth is needed for broker-to-broker communication as compared to that needed to communicate with hosts that lack brokers) and it allows new sources, regardless of location, to be easily integrated into an existing domain.

**Configuration Database**

Brokers and agents obtain information about the resources from the Extended Search configuration database. This database contains information about data sources and how they should be searched. It also stores network addresses, saved queries, saved search results, and data that was downloaded by a web crawler.
You can easily update information about your network topology, data sources, and search applications by using an intuitive Administration interface. This interface also provides the gateway through which you can run discovery (discussed below), view error message and event data, schedule queries, and work with saved queries and search results.

Several wizards facilitate common configuration activities. The wizards enable you to easily export and import data between domains, design the format and content of search result sets, specify data source search and retrieval parameters, and configure mapped fields.

Note that a simple refresh action will disseminate changes you make in the CDB throughout the Extended Search domain. The only time you need to restart the server is when you update configuration data for the server itself.

Discovery

To add data sources to your domain, Extended Search provides a collection of discoverers, programs that load the CDB with default information about a data source. The discovery process automatically configures field and parameter information for each new data source. Later, using the Administration interface, you can designate which fields you want to enable for search and retrieval operations.

Extended Search comes with a broad set of discoverers that enable you to quickly incorporate many of the industry’s common data management systems. Like links and translators, if your data system is not contained in this standard set, you can develop a custom discoverer by using the Extended Search toolkit.

Monitoring

To help you collect statistics and fine-tune the system for performance, Extended Search includes a Monitor, and tool that enables you to observe server activity through a graphical user interface. The Monitor is packaged as a standalone C++ program and as a Java applet that you can launch from within the Administration interface. This feature enables you to make adjustments and refresh the system without having to restart the server.

The Monitor can run independently of the broker, and be started and stopped any number of times, without affecting work being done by the Extended Search server. Because it can run remotely, you can quickly check the status of various servers from a location other than the host console.

Environment

Because Extended Search is designed to use existing software to search for information and retrieve data from wherever it exists throughout an organization, it must integrate well with the existing IT infrastructure.

To this end, an Extended Search domain supports a mixed topology. Extended Search server components (brokers, agents, and so on) can reside on IBM AIX, Sun Solaris, Microsoft Windows 2000, and Microsoft Windows NT platforms, and you can mix the component topology as needed to satisfy the requirements of your operating environment.

As shown in the following illustration, users can submit requests through a web browser or a Lotus Notes client - interfaces that they are already familiar and comfortable with. This design allows Extended Search to provide a distributed search across many different data repositories through a single, efficient, and easy to use point of access.
All user requests get sent to the web server, which in turn forwards the request to the appropriate Extended Search broker. The broker, in turn, contacts the agents needed to carry out the request and search the various target sources.

When access is through a web browser, information about the search (what sources to search, how to search them, and how results should be returned) is determined by the HTML or JavaServer Pages that define the search application.

When access is through Notes client software, information about the search is stored in a search application database, which can either exist on a Domino server or be replicated down to the user's workstation.

Note that Extended Search uses the industry-standard Hypertext Transfer Protocol (HTTP) to invoke the appropriate servlet for processing requests. This approach enables search applications to use many web server-related features such as support for socks, proxies, and secure sockets layer (SSL) technology.
Customer Scenario

NEWCOM Corporation is a typical company with a headquarters office that consists of several departments and sales offices that are located in different locations.

Although sales were expanding at an unprecedented rate, the Marketing and Sales team of NEWCOM Corporation realized it had a problem. The ad-hoc, research-intensive means that sales reps relied on to gather product and customer information could no longer keep pace with the company’s changing business needs. Details were getting lost and unacceptable amounts of time were being spent tracking down information that, had it been at their fingertips, could have been used to close deals within minutes.

To compound the problem, sales reps at one location were often unable to access information managed in other locations, which meant that customer solutions were not being shared and re-used to the company’s advantage.

Yes, sales were expanding. But how many potential sales were being lost? And how much service time was being spent in an effort to maintain customer satisfaction? NEWCOM Corporation realized it needed to take immediate action.

To preserve continued growth, employees needed to be able to search a diverse collection of data stores quickly and easily, readily identify relevant documents, access information from any location, and share results with other team members. The product chosen to solve these problems: IBM Extended Search.

Leverage Existing Infrastructure

Cost was a major consideration for NEWCOM Corporation. It had a tremendous investment in its current database suite and did not want to incur the expense of indexing the information or purchasing additional equipment to store it in a common format. Because Extended Search can tap into virtually any type of data source by using calls native to that source, NEWCOM Corporation could leave its existing infrastructure just as it was.

Before Extended Search, team members had to query each repository in turn to pull together the various pieces of information they needed to build a complete picture. Now, a single query can search a DB2 support database, a Notes requirements database, competitor web sites, and other local or remote, structured or unstructured, legacy or highly dynamic sources in parallel.

Intuitive Interface

NEWCOM Corporation could not afford to spend a great deal of time training its employees to use a new system. The HTML and JSP templates provided by Extended Search enabled the company to produce custom search applications with minimal effort.

What’s more, because they had complete control over the front end, they were able to design interfaces that used concepts readily understood by the sales reps.

They were also able to streamline the query process by targeting specific sources or categories of sources that were most likely to return relevant information, enabling even the most casual user to obtain meaningful results simply by entering simple search phrases.
Of course, sales reps were free to search different sources and specify different search options. The more experienced users could also enter queries in a generalized query language, and take advantage of the precision such queries could render.

Before returning results, the system ranks the findings by relevance, which means the sales reps can get right to the documents that satisfy their needs.

**Competitive Advantage**

One of the sources regularly searched is a web site that contains information about and links to competing products. NEWCOM Corporation chose to take advantage of the web crawling features in Extended Search in order to track changes in the marketplace.

Step 1: schedule the crawler to crawl the site every day and save the data to a file system. Step 2: Define a query that targets the file system data source, save the query, and then schedule it to run every night.

On a daily basis, any number of sales reps can review and act on the search results. For example, a feature being touted in a competitor’s product forum might suggest a new requirement for a NEWCOM product. While viewing the search results, the sales rep can easily e-mail the item of interest to a product manager who can make that call.

**Growth and Acquisition**

As NEWCOM Corporation continued to enjoy expanded sales, it also began to acquire smaller companies that complemented its product line. With these acquisitions came a new realm of potential problems: new and different databases to absorb, more employees to support, varying needs for security, and more geographically dispersed business centers. It found that Extended Search more than adequately met these challenges.

**Links to New Data Sources**

In many cases, NEWCOM Corporation found that it could use one of the many link modules provided by Extended Search to simply add new data sources to the existing search domain.

All an administrator needed to do was discover the source, specify options for how the source should be searched and how results should be returned, either add the data source to an existing category or create a new category for sources similar in purpose or content, and mark the category for inclusion in a new or existing search application.

All of these steps were easily accomplished by using the intuitive Java-based Administration interface. A simple refresh action disseminated changes throughout the search domain. The sales reps submitting queries were often unaware that new sources were being targeted until they saw the search results.

For certain proprietary databases, NEWCOM Corporation realized that it needed to provide custom links in order for Extended Search to communicate with a data source in its native search language. Using the rich application programming interface, the company created a discoverer, link, and translator that enabled data to be searched and retrieved.

Once developed, an administrator would use the custom code to add the new data sources to the system in the same transparent manner as those added through the use of predefined links.
Scalability

With so many new data sources to search, and so many new users to search them, the distributed component architecture of Extended Search offered NEWCOM Corporation the flexibility to scale the system according to its changing needs.

The ease with which search applications could be created made it easy to distribute the processing load. Each application targets those data sources most likely to return information relevant to an area of interest. Employees simply choose the application they want to use from a front-end portal.

When first installed, the Extended Search broker co-resided with the web server. Over time, and with increased use, it became apparent that improved capacity and performance could be achieved by separating these processes, enabling each to devote more resources to respective tasks. The ability to install just a web server component or just a broker component, and indicate the new configuration through a single update in the Administration interface, made this task a breeze.

One of the most extensible features is the ability to configure multiple brokers on different servers. By placing brokers on remote hosts with the data sources they service, NEWCOM Corporation ensured that no single broker would be overwhelmed with queries. Users are automatically partitioned by virtue of the broker associated with the search application.

Setting up multiple web servers and configuring them to service multiple brokers enabled the company to achieve even greater load balancing. In this case, each web server distributes requests to an entry broker (as defined in the search application) who in turn shares the workload among various brokers.

For many of the newly acquired data sources, NEWCOM Corporation determined that it was more efficient and effective to install only the agent component on the machines that host the sources. A central broker funnels requests to the various remote agents, according to the data sources targeted by the search application.

Some sources that required time to instantiate and had strict security restrictions were co-located with faster, more readily accessible sources. NEWCOM Corporation found that it was easy to partition multiple agents on the same machine so that an agent that was servicing the secure, slow source would not bog down queries against the fast sources.

Security

With an ever expanding collection of disparate data sources spread throughout the world, NEWCOM Corporation had concerns over how well it would be able to maintain a trusted environment. The multi-layered security model enforced by Extended Search helped put those concerns to rest.

Extended Search uses the web server as its first line of defense. The web server can use private and public key encryption, digital certificates, and passwords to authenticate users and ensure that only authorized users are able to access search applications.

At the application level, access controls can be specified through the Administration interface. Not only can access to data sources be restricted, but an administrator has complete control over which fields a user can search, view in a results set, or retrieve.
An application-level user exit enabled NEWCOM Corporation to enforce enterprise-specific rules. In this case, the exit communicated with a secure database to confirm the user’s identity, and then returned a different user ID to be used within the Extended Search domain. This feature proved useful because not all IDs are appropriate for all sources. Thus, for example, a Notes ID would not be blocked from accessing an ODBC database.

At the broker level, another custom user exit enabled the broker to selectively deny or approve access to individual data sources named in the search request.

At the agent level, additional security rules were imposed. An exit further validated the user prior to processing the search request. Before returning results, the exit filtered the content and the constraint values and, if necessary, deleted items that failed to pass this post-processing inspection.

As a final guarantee, Extended Search works with the individual data sources, each of which may have their own security mechanisms to permit or deny access to content.

Remote Access

With offices spread throughout the world, NEWCOM Corporation needed to ensure that all sales reps were able to submit search requests and work with results regardless of location or time zone. Because Extended Search provides a web interface, it easily satisfies this requirement.

Whether working at home, at the office, or from some hotel room in Barcelona, employees can access the URL for the search application and submit queries.

Furthermore, employees discovered that they could take advantage of the ability to save and schedule queries.

This feature enabled them to review search results and benefit from someone else’s work without having to spend time composing and submitting a query themselves.

Global Presence

By better managing its information assets and making information immediately accessible to its employees, NEWCOM Corporation continued to grow and eventually established itself as a company with a world-wide presence. Extended Search, through its support for national language, helped the company tap into the global marketplace.

Search applications can be created in many languages, which makes it easy for employees to locate relevant materials. A sales rep in Germany, for example, may prefer to focus on market trends and events reported through German sources and not be inundated with search results that reflect the U.S. marketplace.

National language support also makes it easy for IT groups to set up and maintain networks of web servers, brokers, and agents to support locale-specific needs.

Knowledge Management

Once NEWCOM Corporation found an efficient and accurate way to search for relevant data, it began looking at ways to transform many pieces of information into a cohesive knowledge base. Through its ability to store data in a format that is compliant with international standards for data exchange, Extended Search helped the company achieve its goals.

When saving a query, users can choose to save the search results to a file system. This feature enables Extended Search to operate as an intelligent, query-driven crawler, automatically retrieving content of different formats on a scheduled basis.
Once stored, a document can later be rendered in its native form or be analyzed and indexed by another product, such as Lotus Discovery Server.

Examples of how NEWCOM Corporation benefits from this feature include:

- Employees are able to directly search the results whenever they submit a query against the file system.

- The results set was trapped and fed it into a ticker tape applet. Now, whenever users log on, they see a streaming list of “hot docs”. By clicking on an item as it scrolls by, they can access the document and stay current with the latest information.

- An administrator added file extensions to the stored data, thus enabling each document to be rendered by its native viewer or by a web browser. For example, adding a .doc extension enabled Microsoft Word to open the document.

In most cases, administrators add the extension .html, which enables a browser to display the document regardless of whether the native application is available. This feature makes it possible for everyone to view Lotus Notes documents, for example, even if Notes is not installed or locally available.

- A custom application imports the stored result data into other applications. The flexibility offered by this feature enables NEWCOM Corporation to massage the data according to how it intends to put the knowledge to use:

  - In some cases, a decision is made to crawl and index the data, thus making it meaningful for data mining and knowledge management purposes. This approach enables employees to rapidly access relevant information by using keywords that were associated during the indexing process.

  - In other cases, the data is made available without first indexing it. When storing a document, Extended Search can create an XML wrapper that identifies the original document and the location of the document as stored in the file system. The document identifier, which typically takes the form of a URL, enables users to access the original source of the document, not just the version that was stored when the search results were saved.

A Success Story

Faced with ever expanding volumes of data, the employees of NEWCOM Corporation were thrilled to discover that a single tool, which they could access anywhere at any time, enabled them to tap securely into a myriad of sources and retrieve meaningful information. The IT group was pleasantly surprised at how little effort was required to set up Extended Search initially, and how little time and investment was needed to extend the search domain as new data sources were added to the corporate domain.

With facts at their fingertips, employees found that they were able to spend much more direct time with customers, an activity that proved to be directly proportionate to a steady increase in sales. While responding more rapidly to new opportunities, they could also provide better and more timely information to the existing customer base, an attentiveness which translated into even more sales.

In addition to improved productivity, employees found that they were able to better track and react to events in the constantly changing competitive marketplace. As the customer base grew more geographically dispersed and international in flavor, the ability to share and re-use information enabled deals to be closed in record time.

IBM Extended Search met the challenges posed by NEWCOM Corporation and provided a solution that more than satisfied the company’s expanding business needs.