IBM Lotus Extended Search

White Paper on Security, Scalability, and Performance

Version 3 Release 7

Seventh Edition, July 2002

This edition applies to Version 3 Release 7 of the IBM® Lotus® Extended Search product and to all subsequent releases and modifications until otherwise indicated by new editions.

Contents

Security

IBM® Lotus® Extended Search gives you the ability to access data from a wide range of sources. In many cases, this accessibility must be controlled to ensure that only authorized content is available to each qualified user of the system.

Lotus Domino and IBM WebSphere have proven reputations for incorporating the industry's best security measures, including access control, private and public key encryption, and digital signatures. Web application server technology uses several mechanisms that interact with each other to form the basis of a secure environment. Genuine security is possible when these mechanisms work together within such defended and trusted environments.

Extended Search is a linking technology that connects users to disparate data sources that are distributed throughout the enterprise. A broad range of search engines, each with their own security models to enforce, host these data sources. To access data sources, an Extended Search link uses the native programming interfaces of the search engine. This activity usually occurs outside the control of the Web server. Overall protection against unauthorized searching is therefore a coordinated effort between the Web server, Extended Search, and the backend data sources.

The Extended Search product employs the following kinds of security:
- "User authentication" on page 2
- "Access control" on page 5
- "Link-level control" on page 7
- "Message encryption" on page 8
- "Notes database control" on page 9

Figure 1 on page 2 illustrates how Extended Search implements different layers of authentication and access control throughout the system.

- The Web server provides the first level of control. Using its native authentication mechanisms and access controls, the Web server validates each user request and determines whether or not it can proceed.
- At the application level, additional validation is performed:
  - For requests that are received through the Web Client, Extended Search provides an exit point for enforcing enterprise-specific rules. The exit can, for example, interact with another security system or return a different user ID for use by the rest of the Extended Search system.
  - For requests that are received through the Notes Client, Extended Search can use normal Notes database access control lists (ACLs) for validation.
  - Within the configuration database, application-specific controls can restrict access to data sources. They also control which fields in a data source end users are able to search, view in a hitlist, or fetch from the data source.
- An Extended Search broker provides authentication services through a user exit. For each request received by the broker, enterprise-specific rules can validate or alter user information.
- An additional exit point is available at the link level. Enterprise-specific rules can validate the request, alter the request, or filter the search results. In addition, you can configure a link so that it connects by using a specific user ID or by using the user ID passed in on the request.
- Lastly, individual data sources may have their own mechanisms to permit or deny users access to the data source or its contents.

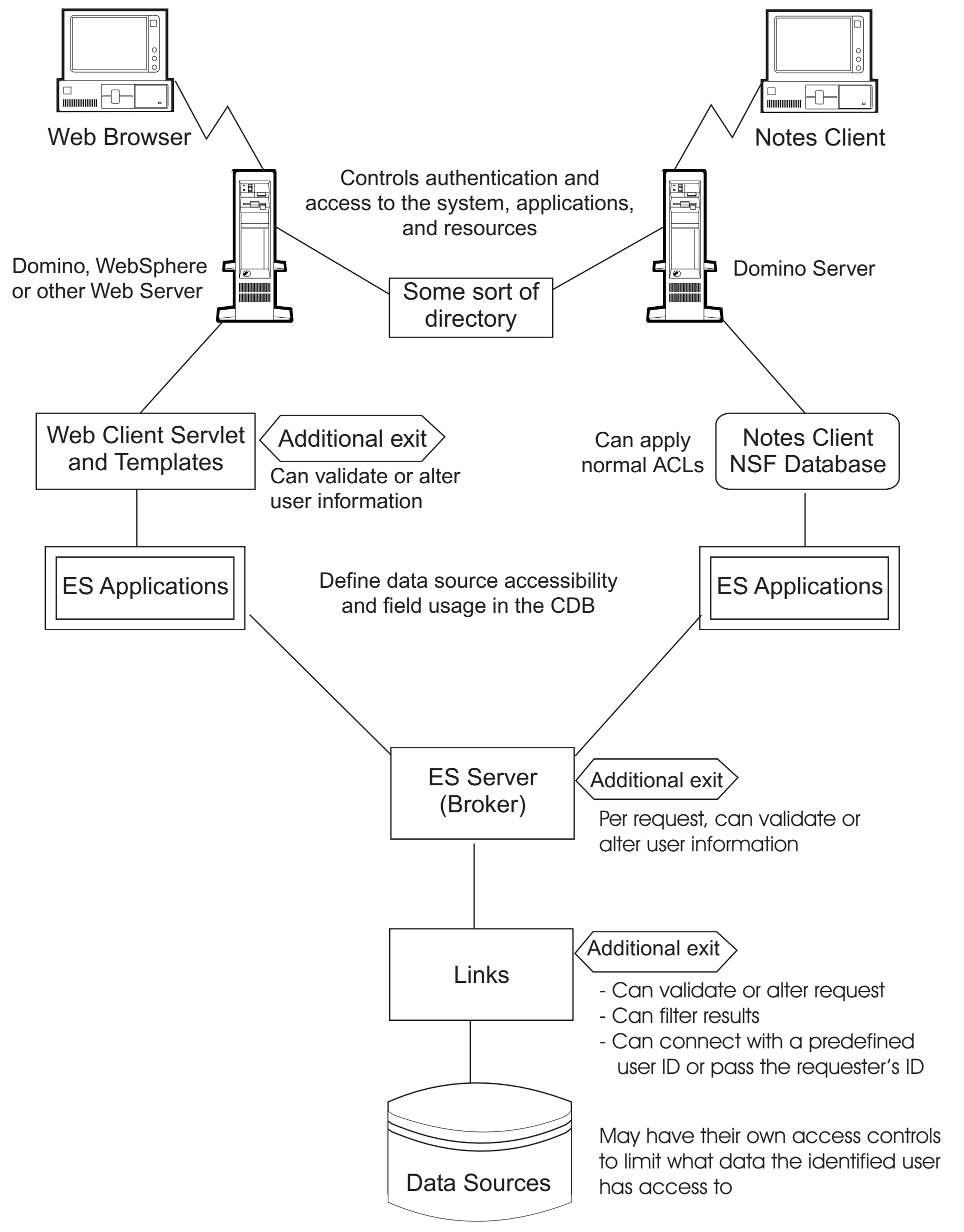
Web Browser
Notes Client
Controls authentication and access to the system, applications, and resources
Domino, WebSphere or other Web Server
Some sort of directory
Domino Server
Web Client Servlet and Templates
Additional exit
Can validate or alter user information
Can apply normal ACLs
Notes Client NSF Database
ES Applications
Define data source accessibility and field usage in the CDB
ES Applications
ES Server (Broker)
Additional exit
Per request, can validate or alter user information
Links
Additional exit
- Can validate or alter request
- Can filter results
- Can connect with a predefined user ID or pass the requester's ID
Data Sources
May have their own access controls to limit what data the identified user has access to

Figure 1. Authentication and access control

User authentication

User authentication refers to the process by which Extended Search verifies that users are, in fact, who or what they declare themselves to be. Depending on the system component that is involved, Extended Search uses different methods of user authentication:

- Authentication by the Notes Client
- Authentication by the Web Client
- Authentication by the broker

Authentication enforced by the Notes Client

User authentication is very secure when users access Extended Search through Lotus Notes client software. When used with the Notes client, the system uses private/public key encryption, digital certificates, and passwords to provide the highest degree of user authentication. Unless the user’s Notes UserID file has been compromised, this level of security assures that users accessing the search application are who they claim to be.

Authentication enforced by the Web Client

Because the Web Client is browser-based, authentication is primarily dependent on standard Web server authentication protocols. Typically, the authentication options configured for your Web server provide the first level of defense. Like Lotus Notes, the Web server can use private/public key encryption, digital certificates, and passwords to ensure that users are authorized to access Web Client applications.

The Extended Search Web Client consists of one or more HTML or JSP pages that users view with Web browsers. An Extended Search servlet runs on the Web server to preprocess the pages through HTTP-based requests from users. When you configure authentication options for the Extended Search server, you can choose to have the Web server perform no client authentication, basic authentication, or custom authentication.

No Authentication

None is the default installation value. All users with access to the server can run queries against Extended Search.

Basic Authentication

If you configure the server to perform basic authentication, the Web server will challenge the user with the standard HTTP authentication prompt for a user ID, a password and, in some cases, an authentication realm. If the user-supplied information does not exist in the server’s directory, the server will deny the request. Upon successful authentication, the user information is retained by the browser (for the duration of the current browser session) so that the user does not have to re-enter this information with each new request.

Authentication and the Administration interface

To secure the Administration interface, you need to either physically protect the server that hosts the configuration database or set up Web authentication to control access to the program. If you choose to use authentication, you must update the following environment variables to identify the Extended Search administrator before starting the Extended Search server.

- On UNIX, update these entries in the Extended Search **desStart** script:
  - ESAdminUserid
  - ESAdminPassword
- On Windows, update these Extended Search software entries in the Windows Registry:
  - AdminUserid
  - AdminPassword

If you enable user authentication in the Web server, it will impact the operation of the Administration interface when you run it as an applet (it does not impact the Administration application). For example, you will be challenged numerous times to authenticate yourself when starting and running the applet.

If you want to run the applet with authentication, you should configure separate ports on the Web server to handle different types of requests. For example, you could use port 80 to handle search requests that require basic authentication and use a non-standard port for the Administration applet. Depending on your security environment, you may not be able to access this non-standard port outside your firewall.

Custom Authentication

When authentication is set to custom, the system first performs basic authentication activities. If Web server authentication succeeds, it then calls your custom authentication user exit.

Custom authentication involves the enforcement of enterprise-provided authentication rules. You encode the enterprise-specific rules by writing a custom Java class. Information available about a request (such as the requester's user ID and password) gets passed into this exit.

Note that this exit can perform additional checks, such as interacting with another security system or a database access control list. It can also return a different user ID for use by the rest of the Extended Search system.

Extended Search does not use custom code by default:

- For information about developing a custom Java authentication class, see the discussion of *User Exits* in *Extended Search Programming*.
- For information about configuring the server to use this code, see *Extended Search Administration*.

Authentication enforced by the broker

Custom authentication can be applied at the broker level to provide additional security checks at intermediate network nodes behind the Web server. The activation of this custom authentication code is indicated through a setting in the broker's configuration data. If the code has been activated, the broker calls the customized function prior to the execution of each request.

Note that you can apply customized broker authentication to a specific Extended Search request type. For example, the identity of a user searching database content might not be of concern (no authentication), but the identity of a user retrieving documents could be.

From within the shared library, you can selectively deny or approve access to individual data sources that are identified in the search request. You also have the ability to provide alternative user ID and password mappings for discrete data sources.

Extended Search does not use custom code by default:

- For information about writing a broker authentication exit, see the discussion of *User Exits* in *Extended Search Programming*.
- For information about configuring a broker to call this exit, see *Extended Search Administration*.

Access control

Access control involves limiting what users can do once they have identified themselves. An access control list (ACL) is the most common way in which access to computer system resources can be limited. An ACL is simply a list of user names and group names. Each name is associated with a set of permissions that define the user's rights and privileges. Access control is supported through the following methods:

- Notes database ACLs and impersonation
- Web-based access control
- Data source filtering

Notes ACLs and impersonation

Domino has a very robust set of access control features implemented through the use of ACLs. Domino uses ACLs to protect Notes servers, databases, documents, and even fields within a given type of Notes document.

Extended Search users can interact with the Notes Client application to submit queries, view results, and fetch documents. As the first line of defense, you can effectively apply Domino's standard access control features to the Notes Client application. For example, it is here where you can restrict initial entry into the database, control who can submit queries, and encrypt data in the search result documents. Each level of Notes security restricts information to an ever decreasing group of users and cannot override a higher level.

Extended Search provides design templates for the Notes Client application (different templates apply to different versions of Lotus Notes). After creating an application database from a template, you can use standard Notes ACLs and other available security features to control access to Extended Search. Typically, you will define multiple client applications, each of which may have different access controls.

When a user submits a query to search a Notes data source, Extended Search does not automatically check the permissions granted to that user per the ACL defined for that data source. You can enhance security by configuring a parameter, **Impersonate**, that instructs Extended Search to compare the requesting user's name against the Notes ACL.

If the user is not on the ACL, access to the data source will be denied. Note that the user will not see error messages about this denial, but the hitlist will indicate that zero results were returned from that data source.

Before setting up impersonation, be sure you understand the following usage guidelines:

- ACL verification may degrade performance slightly.
- The Impersonate parameter uses the full canonical user name (or distinguished name, DN). You need to either define the DNs in the Domino address book, which allows users to log in with their short name while the full DN gets passed to Extended Search, or you can write an exit to handle name parsing.
- The databases for which you are setting the Impersonate parameter must exist on the same machine with the agent that services them.
- To enable impersonation, you must also enable basic authentication on the Extended Search server.
- Impersonation causes a user ID and password to be passed in, not a digital credential. However, Extended Search does encrypt the password (it is not passed in the clear from the Web server).
- If you set Impersonate as a link parameter, the system performs ACL verification against all Notes 4.6.4 and Notes 5.0 data sources. This is the most typical approach.
- If you set Impersonate as a data source parameter, the system performs ACL verification only against the data source in which you configured the parameter.

ODBC and Web Source Impersonation

You can also use the Impersonate parameter to require that a user ID and password be passed in on search requests that target ODBC sources and Web sources. In this case, Extended Search attempts to establish a connection by using the requesting user’s name and password. If the information is not valid, the user cannot search the source.

Web Client access control

As discussed in “Basic Authentication” on page 3, Extended Search servlets can also use ACLs created for Notes databases to further restrict user access through the Web. This form of access control occurs only when the options for servlet-level authentication specify both basic authentication and the name of the controlling Notes database.

Data source filtering

Data source filters further restrict access to data sources. Extended Search uses application identifiers that enable you to restrict access to particular types of users (for example, financial applications versus personnel applications).

In the configuration database, you can define any number of applications (each application has a unique name that serves as its unique identifier). You can then associate these applications with any number of data sources. Extended Search uses these associations as a filter. They allow users to access only those data sources that belong to categories associated with a given application.

Link-level control

The term agent refers to the software that implements the search logic for the various data sources to which Extended Search connects. Agents exercise access control at the link level. The amount of control depends on the type of search engine involved and whether or not your enterprise has developed a custom shared library to control access to data sources.

Link-level control is available through the following methods:

- Controlling how an agent connects to a data source
- Controlling how an agent authenticates access to a data source

Connecting to data sources

At startup, and whenever the configuration gets refreshed, an agent learns about the various search engines through the broker. In general, the agent does not perform the connection to the search engine until it receives the first query for the target data source.

Connecting to a search engine generally involves the establishment of a session with the backend source. The agent obtains the data it needs to establish a session from the configuration database as either link or data source parameters.

Extended Search implements two connection models: persistent and dynamic.

Persistent Connections

Once a session is established, the agent will maintain the session for the duration. It will not terminate the session until the server is either shut down or instructed to refresh its configuration.

This mode of operation permits an agent to service search and retrieval requests as quickly as possible without incurring the additional overhead of session management. It keeps the total number of connections low. It also reflects the number of connections relative to the number of data sources being searched, not the total number of sources that can be searched.

In this design, it is important to realize that the agent is performing as a proxy to its associated link on behalf of the user community. As far as the search engine is concerned, the agent is a user that is logged on with a unique ID (if required) as specified in the configuration database.

Dynamic Connections

For an additional level of access control, most of the links that are provided with Extended Search also support making connections at request processing time. These types of connections typically use the requester’s user ID to establish the session. This approach allows the data source to determine what data should be returned to the requesting user.

Authentication enforced by an agent

It is possible to invoke an enterprise-specific security user exit at the agent level. This custom shared library behaves similarly to the broker’s custom authentication function.

When enabled, the agent calls the custom library prior to the execution of each request to perform pre-query processing. It can call the library again after processing each request to perform post-query processing. The agent passes in all available information needed to perform authentication or access control (such as user ID, password, type of request, list of data sources, and so on). Based on this information, the shared library will either allow the request to proceed or deny the request.

In addition to basic data source filtering, the user exit can also filter the content that is to be returned. In return content (post-query) filtering, you must create a custom shared library that uses query information (such as the user ID and password, the application name, a list of document IDs, and a list of field names) to determine whether any of the documents or fields should be hidden from an end user. When enabled for post-query filtering, the agent calls the custom shared library after content has been returned in response to a request.

If pre-query screening prohibits a query from proceeding, then post-query filtering is rendered irrelevant. The shared library that contains the customized function makes use of API calls to extract information about a query and its return content. The library then uses other API calls to mark entire documents or individual document fields for exclusion from a user's view. The agent honors these API actions and removes the indicated documents or fields from the return content.

Extended Search does not use this custom code by default:

- For information about writing an agent-controlled security exit, see the discussion of *User Exits* in *Extended Search Programming*.
- For information about configuring an agent to call this exit, see *Extended Search Administration*.

Message encryption

User actions performed in a client application result in request/response pairs being exchanged with the server. This exchange is equivalent to messages that are transmitted through the network. Extended Search supports message encryption when the privacy of this message traffic is of concern. Message encryption involves the electronic scrambling of message content to prevent unauthorized viewing of sensitive data during transit.

Both user authentication and access control rely on the exchange of sensitive data, such as IDs and passwords. Documents returned in response to a fetch request can contain sensitive information, and even the results of a search can contain sensitive data. Extended Search provides two levels of message encryption:

- Encryption of message authentication data
- Encryption through the Secure Sockets Layer (SSL) protocol

Encryption of message authentication data

The first level of message encryption encrypts only the authentication data portion of a message. This method applies to all messages transmitted through an Extended Search system with the exception of those passing through servlets. This is the minimal amount of encryption provided, and it incurs the least amount of overhead because only authentication IDs and passwords get encrypted.

This level of encryption uses a 64-bit symmetric Data Encryption Standard key to encode the authentication data for each message.

Encryption through SSL

For Extended Search to use SSL for message encryption, you must properly configure the Web server for SSL. Procedures for enabling SSL vary with the Web server used. Consult the product documentation for the Web server product used in your Extended Search system.

Remember that the SSL port number that is used by the secure Web server is typically different from the standard Web server port number. Also, be aware that SSL encrypts the entire message and thus will result in a slight degradation to overall message throughput for the system.

The level of SSL support available to your applications depends on whether you use the Web Client or Notes Client application.

- If you use the Web Client, you can use the SSL version 3 (SSLv3) protocol for secure communications when searching data sources.
- If you use the Notes Client, Extended Search includes support for the SSL version 2 (SSLv2) protocol. The client LotusScript Extensions (LSX) use the HTTP protocol to communicate with the Extended Search server. This communication passes through the Web server to the Extended Search Reflector component.

  The Reflector is a very small CGI program whose only job is to forward the client request to the Extended Search server. An advantage to this approach is the ability to leverage industry-standard SSLv2 message encryption between the client LSX and the participating Web server.

  The client always initiates the SSL protocol. Therefore, you must enable SSL as a part of configuring your Notes Client applications. The Application Document contains a check box for SSL enablement.

  Note: To enable SSL with the Extended Search Notes Client application, you need the Microsoft Internet Explorer browser and Version 4.72 or later of Microsoft's WinInet Dynamic Link Library.

Notes database control

When you use Extended Search to search across multiple domains and heterogeneous data sources, you only partially realize Notes security capabilities. This situation is because Extended Search crosses the boundaries of closed systems to provide search results from multiple databases. To optimize Notes security features for Notes databases that are accessed by Extended Search, you can employ the following solutions:

- Configure multiple Notes Client applications
- Configure database-level access controls
- Configure document-level and field-level access controls

Multiple client applications

One approach for optimizing Notes security would be to create multiple instances of the Extended Search Notes Client application. You must assign each instance a unique application name and provide access to a specific set of Notes databases. Extended Search can then use standard Notes ACLs to permit users to access specific applications.

Database-level access controls

The proxy-like behavior of agents applies to Notes databases that are managed by Domino versions 4.6.3 or earlier. In this situation, the agent uses the ID file that is co-resident with the agent to search version 4.6.3 Notes databases. This ID file is usually the Server ID.

Consequently, search results may contain links to documents that the Domino Server Administrator is authorized to view but that the requesting user is not. If the user clicks the link in the hitlist to view the document, standard Notes ACLs and authentication will prevent the document from being displayed.

For databases that are managed by Domino versions 4.6.4 or higher, the Notes link can be configured to check the ACL before permitting a query against a database. When this feature is enabled, the user's name must appear in the ACL for the target database. Furthermore, the user's permission level must not be set to **Depositor** or **No Access**.

Extended Search does not check Notes ACLs by default:

- For information about how to write a custom user exit to implement agent-controlled access to databases, see the discussion of *User Exits* in *Extended Search Programming*.
- For information about how to update the configuration database to enable ACL verification for Notes databases, see *Extended Search Administration*.

Document- and field-level access controls

If you use Domino version 4.6.3 or earlier and need a more granular access control method (for example, access control at the Notes document level or field level), consider the following techniques.

- Allow all users to perform searches, but allow only selected users to retrieve complete information.

  With this approach, all users are able to see non-sensitive information in the hitlist. However, only those users with an authorized user ID are able to retrieve sensitive information within documents. This approach allows users to discover the existence of documents, even if they do not have access rights to the sensitive information within the documents. The user would presumably need to request access from the appropriate authority.

  To achieve this scenario, edit the application-specific properties for the affected data source and ensure that only non-sensitive fields are returned in the hitlist. You also need to ensure that the only way users can retrieve documents is through the use of links to Notes documents that are rendered in a hitlist abstract.

  This latter step requires you to define the fields as returnable but not fetchable in the application-specific properties for the target data source. The result is that users will be able to see public information in the hitlist. However, they will be able to open the complete document only through a Notes link with the appropriate Notes security permissions.
- Configure a two-channel application.

  Use this approach to enforce an even stricter access model when the Extended Search application identifier is not adequate protection for sensitive information within a single application instance. For example, you may have a data source that does not contain any information that can be viewed by the public.

  Using the two channel approach, two completely separate channels of communication are established: a public channel and a secure channel. You must create two instances (copies) of the Client application, and assign each instance a unique application name and list of data sources.

You must set up each instance of the Client application to communicate with a different broker and agent combination. One agent (the public channel) would use a public Notes ID. The other agent (the secure channel) would use a Notes ID that has access rights to secured data. In this scenario, public users would use one communication channel while authorized users would use a second, secured channel to access sensitive data.

When using these granular approaches, keep in mind that the agent that accesses a Notes database is using a Notes ID. You should treat the agent's ID as you would any user's ID when determining whether access to a particular document or field should be granted.

Extended Search does not check Notes ACLs by default:

- For information about how to write a custom user exit to implement agent-controlled access to documents and fields in a database, see the discussion of *User Exits* in *Extended Search Programming*.
- For information about how to update the configuration database to enable Notes ACL checking, see *Extended Search Administration*.

Scalability

Extended Search has been engineered to accommodate future scalability requirements. The following topics should provide sufficient information to help you expand the Extended Search system as the need arises. They explore different Extended Search topologies that take into account the size of an enterprise's network and user population.

Although scalability should be your primary focus, you should review this information in conjunction with "Performance" on page 25, which examines factors that can help you optimize searching in Extended Search. Generally, a system that is designed to be scalable leads to a system that also performs well. A number of factors influence both scalability and performance:

- Hardware, both the amount of memory and the number of processors.
- The number of users.
- The number, type, and speed of data sources in your domain. Keep in mind that Extended Search can never be faster than the backend sources because it must wait for the sources to respond. Because some of the sources, and the links to them, contain quite a bit of overhead, you should not install too many different types of data sources on the same server.
- Other factors beyond the control of Extended Search, such as network traffic.

Following guidelines that may help you assess your scalability needs in general, review the following ways that you can scale an Extended Search system:

- "Creating multiple client applications" on page 14
- "Separating brokers from the Web server" on page 15
- "Separating agents from brokers" on page 15
- "Setting up multiple brokers" on page 18

To help you evaluate the changes you may want to make in your environment, you should also review "Broker network configurations" on page 21.

Assessing scalability

In general, your first exposure to the Extended Search product is through installing all the components on a single machine. This arrangement is useful for demonstration purposes, exploration of Extended Search capabilities, and workgroups of a limited size.

The base scalability option is to increase the resources available on the Extended Search server. You can then adjust the configuration of server tasks as appropriate. The architecture of Extended Search allows it to take advantage of available resources to support additional workload and improve performance. However, as your system expands, there are other options you can pursue to distribute Extended Search across multiple servers.

The distributed component architecture of Extended Search offers the flexibility to scale a system according to specific requirements. It also allows an enterprise to arrange the Extended Search components in a topology that best matches its needs.

In a full production environment, where the number of Extended Search users number into the hundreds or thousands, it may be more practical to run the

Extended Search server on a different machine from the Web server machine. It may also be desirable to have the agents be co-resident with the search engines to which they connect. In this instance, the agents would be configured to run on the machines that host the data sources. Some enterprises may require a distributed processing solution to accommodate data sources or user groups that are geographically dispersed.

The Extended Search architecture supports vertical and horizontal scalability.

- Vertically, within a single Extended Search server, you can configure any number of broker and agent tasks. "Determining the optimum number of tasks" on page 30 provides guidelines for this assessment.
  - The number of broker tasks defines the number of simultaneous client requests that the server can process.
  - The number of agent tasks defines the number of simultaneous data source search 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 decide whether you want to run only broker tasks, only agent tasks, or both.
  - By having multiple Web servers, you can distribute user load. It is recommended that you use one Web server to run the Administration interface, and use at least one other Web server to handle user requests.

Scalability is an advanced topic that requires a full understanding of Extended Search concepts and components. Be sure to read the product's *Technical Overview* in *Extended Search Programming* before reading the following scalability guidelines.

Creating multiple client applications

An Extended Search server can be accessed by any number of client applications, each tending to its own user communities. When you configure Extended Search, you can configure many different instances of client applications to support the varied and specific needs of your user community. Each client application has it own application-specific definition in the configuration database.

An Extended Search Web Client application consists of HTML or JSP pages. The pages imbed easy-to-use Extended Search tags or beans that let users enter query strings, identify various sources to be searched, and set a multitude of search options. An Extended Search servlet translates the Extended Search-specific tags into corresponding search functionality and submits them to the server for execution.

The creation of multiple Web Client applications is a simple matter of creating a different set of search forms, with imbedded Extended Search functionality, for access by particular user groups.

If your users use the Notes Client application, you should create separate client application databases. With separate databases, you can partition functions (such as financial versus personnel data), enforce group filtering at the database level, and split the application processing load.

An Extended Search Notes Client application is a Notes database created from a template file (.nsf) provided by Extended Search. When you select the Notes Client to install, the template file is copied into the Domino server's data directory. See *Extended Search Administration* for instructions on how to create a Notes application database from the database template.

Separating brokers from the Web server

Continued growth in the use of Extended Search will eventually bring a broker to the limits of its machine resources. When the broker shares a machine with a busy Web server, the limits of the resources actually available to the broker are reached much sooner. A first step in improving performance, therefore, is to place the Web server and the broker on separate machines. Figure 2 shows a schematic of separating the Web server and the broker.

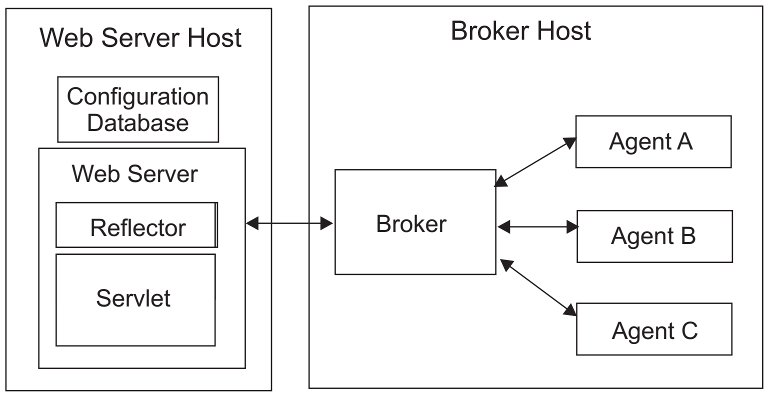

Figure 2. Running the broker and Web server on separate hosts

When you installed Extended Search, you installed Web server support on the machine that hosts your Web server. If you installed the Extended Search server on this same machine, you need to update the configuration to specify where the broker now resides. If you renamed the broker to be consistent with its machine name, you also need to update configuration data for your applications. You must ensure that each application references the correct broker name as the entry broker. This is necessary to ensure that user requests are properly redirected from the Web server to the broker's server.

For instructions on how to separate a broker from the Web server and how to configure applications to use the new broker, see *Extended Search Administration*.

Separating agents from brokers

The number of active agents plays a crucial role in determining the effectiveness of Extended Search and the response time to the end user. Consider, for instance, three users, all of whom issue queries directed at fifteen sources. Unless there are at least 45 copies of the agent task available to process these requests, at least one of the end users will experience an increased response time. The amount of time is over what would occur if only one individual were accessing the system.

Resource usage studies show that it is rarely economically feasible to support a worst case scenario. That is, you do not need to plan to have enough agents to support all users searching all sources simultaneously. Instead, you need to decide what level of support to provide.

You need to estimate how many simultaneous requests you will need to support along with the average number of sources that will be queried per request. The first value indicates the desired number of Search Manager tasks that should be running. The first value multiplied by the second value yields a value indicative of the desired number of agent tasks.

As Table 1 on page 30 illustrates, multiple simultaneous users, and a large number of queried sources, can cause the number of desired agents to quickly exceed the recommended limits for a single machine. One of the scalability upgrades available through Extended Search is that of installing agents, plus an Extended Search server to control them, onto a separate machine. Figure 3 illustrates the schematic of such a partitioning.

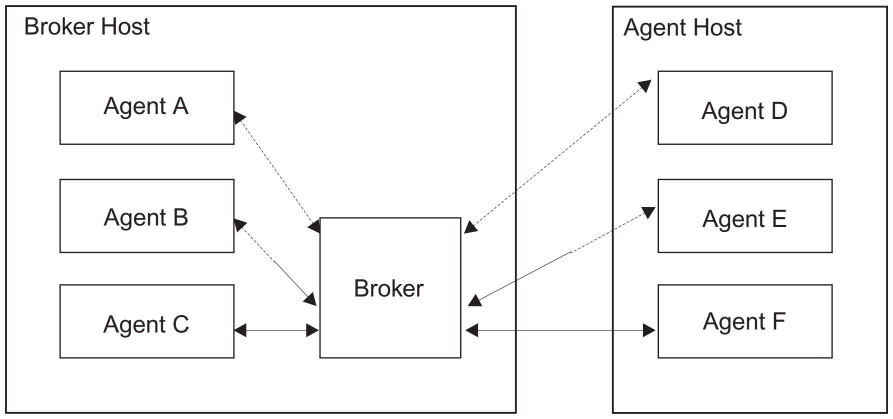

Figure 3. Running brokers and agents on separate hosts

Installing some agent tasks on another machine will not help if all of the actual sources, and their associated search engines, remain on the same broker machine. In fact, response time might actually degrade if agents on a remote host needed to communicate back to the broker's host to access their associated search engines. Accordingly, for this scaling upgrade to be effective, you should migrate or replicate some sources, but usually not all sources, on the remote host.

You should rely on performance statistics (or best guesses) when you select the sources to move, and decide how many to move. Moving a smaller number of heavily searched sources to the remote machine is likely to have a greater impact than moving a larger number of rarely searched sources.

The partitioning of agent tasks should be based on an estimate of how many sources on each machine are likely to be queried individually by ***N*** simultaneous requests. (***N*** is the number of simultaneous user requests your system is configured to support.)

The following topics discuss separating agents from brokers, moving sources to a remote host, and adding remote agents to load balance the query effort. It is possible to expand that approach by moving sources to multiple remote hosts and adding agents on all of those hosts.

- Configure remote links
- Assign unique agent names
- Partition multiple agents and assign sources to each agent

Configuring remote links

Agents are server tasks and thus need an Extended Search server to start them up, handle their communications, and provide other client-server architectural needs. Accordingly, there needs to be configuration data for the Extended Search server and its agent tasks on the remote host.

For instructions on how to set up a remote link, see *Extended Search Administration*.

Assigning unique agent names

An agent is a server task capable of dynamically linking to all of the data source types that are supported by Extended Search. The exact set of data sources is determined at initialization time through a configuration request made to the broker. The name of the agent is passed along with the request. The broker responds with a list of sources that can be serviced by an agent with that name.

Consider an example where all of the sources have been partitioned across two machines, A and B. All of the Notes sources are on machine A, and all of the File System sources are on machine B. Further assume that the agents do not have unique names (both use the default name **agent**).

The agent on machine A receives its configuration information. It learns that it is expected to search both Notes and File System sources, even though the actual file system documents are on another machine. This is of little consequence because machine A will never be sent a search request for those data sources (by way of their network addresses). However, it is important if prerequisite software for that type of link is not installed.

Access to data sources of a certain link type may depend on other product software being available on the machine where the agent is to run. For example, for an agent to access Notes data sources, then Notes software must be available on that machine. In this example, the presence of the Notes software on machine A satisfies this requirement (file system searching requires no additional software).

But now consider the agent on machine B. Because it was not assigned a unique name, it will have configured itself to search both Notes and File System sources. A problem results because machine B does not have Notes software or databases installed. The result is typically an operating system message that identifies the missing software (in this case Notes), and a subsequent failure of the agent to initialize.

Partitioning agents on a single machine

Extended Search uses both the broker name and agent name to provide the agent with information that it needs. This feature means that you can partition sources among multiple agents on the same machine. You might decide to establish more than one agent on a single machine for the following reasons:

- You may want to distribute resources by segregating a certain type of data source and have those sources serviced by one agent.
- You may need to isolate a particular source. Isolation allows you to send test queries without exposing the source to the user population.
- You may have a new type of data source that requires the development of a new discoverer, link, and grammar. Isolating the source allows you to load and test only the new shared libraries required by that type of data source.

- You may need to dedicate an agent and direct queries to a single data source, such as a data source that takes some time to instantiate, or that has security restrictions and limited access. For very slow data sources, you may want to configure a dedicated agent, start 50 copies of this agent, and allow another agent or two to handle all other requests.
- You may want to reduce the load on a source with a fixed number of simultaneous connections. By placing all the agents for the source in one set, you can grant another set of agents a larger number of server task copies. This approach may better utilize the resources of the machine.

For instructions on how to set up multiple agents on the same machine, and how to assign data sources to each agent, see *Extended Search Administration*.

Setting up multiple brokers

"Separating agents from brokers" on page 15 discusses how Extended Search supports the partitioning of sources across multiple machines. This allows the breadth and performance of your searches to be bound only by the number of machines that host the content to be searched.

You should note, however, that there is an increase in machine-to-machine communication that is incurred through this scaling method. Every source being queried, by each user, results in a separate message being sent to an agent that resides on the machine with that source. When all sources are local, this communications cost belongs entirely to the Extended Search server, and it is entirely negligible.

If a search is run against eight dozen sources, the broker will send 72 queries to the remote machines. (For example, you may have three dozen sources on one remote host and three dozen sources on another remote host.) Presumably, the broker will receive 72 hitlists in return. Each individual hitlist may be very large. If each hitlist contains the maximum number of hits, then most of that data will be discarded when the broker consolidates all of the hitlists. The broker keeps only the top entries, up to the value configured for the maximum number of hits that can be returned to the client application.

Another scaling approach is to configure multiple brokers on remote hosts with remote data sources and agents. In Extended Search, the most useful and extendable scalability method is the use of multiple brokers. Under a multiple broker schema, sources get partitioned across all of the brokers, which keeps any one broker from being overwhelmed with queries.

With multiple brokers, you can partition the user population and limit the number of queries that are handled by each broker. Additionally, broker-to-broker communication decreases the bandwidth needed (as compared to that for remote hosts without brokers). It also spreads out, as well as decreases, the effort for hitlist consolidation.

Figure 4 on page 19 illustrates a schematic for two networked brokers, both of which interact with the configuration database and a Web server. Having a single configuration database define the Extended Search domain allows system administrators to maintain configuration data at a centralized point. This reduces both the effort and the risk of mistakes as compared to a design wherein each broker has its own independently maintained configuration database.

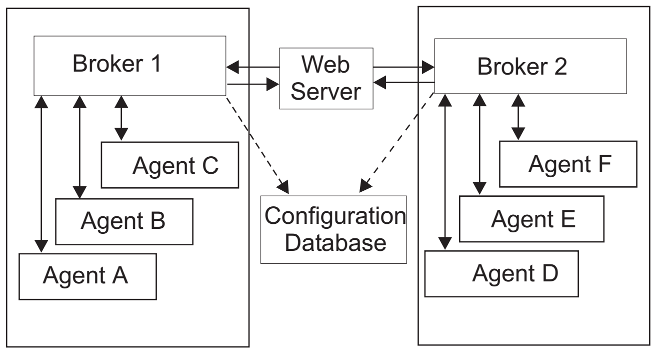

Figure 4. Dual brokers

The broker and agent tasks on each broker machine need configuration information. In addition, each broker contains an Internal Configurator task. Furthermore, all the other server tasks on each broker send their configuration request to the local Configurator task. This design minimizes the configuration changes that are necessary for scaling up to multiple brokers.

All of the individual (non-Configurator) tasks, including those for the agents, are exempt from having to know which broker machine they are running on. They simply ask their local Configurator task for the appropriate configuration data. The Configurator task running on each broker knows that it needs to forward all requests for configuration to the configuration database.

For instructions on how to set up multiple brokers, see *Extended Search Administration*.

Advantages of multiple brokers

One of the most important advantages to the multi-broker environment is that the entry broker does not have to send queries directly to every remote source. Instead, it can send a single message to the broker on each remote machine in the broker chain. The remote brokers then split the message into multiple requests for the sources (fronted by links) on that machine.

There is an even greater improvement on the return path. Instead of every source's hitlist returning to the entry broker, each broker consolidates the hitlists from the sources on its machine. Each broker then returns one message with a single hitlist to the entry broker. The entry broker only needs to consolidate a final hitlist from its own local sources, plus one hitlist from each broker in the broker chain.

Partitioning the user population

After you configure multiple brokers, you can achieve additional enhancement by partitioning end users such that the same broker is not the entry broker for the entire user population. An entry broker is the first broker (potentially in a chain of brokers) that receives the query from an Extended Search Reflector or servlet.

For each query, the entry broker performs more actions than the remote brokers. Thus, assigning all of the brokers to be entry brokers can enhance performance. Alternatively, assigning a noticeably faster machine to be the designated entry broker can accomplish the same objective.

The easiest means to partition users is to establish multiple Web servers. You assign a portion of the user population to each Web server, and have each Web server point to a different entry broker. Figure 5 illustrates a schematic for this solution with two Web servers and two brokers.

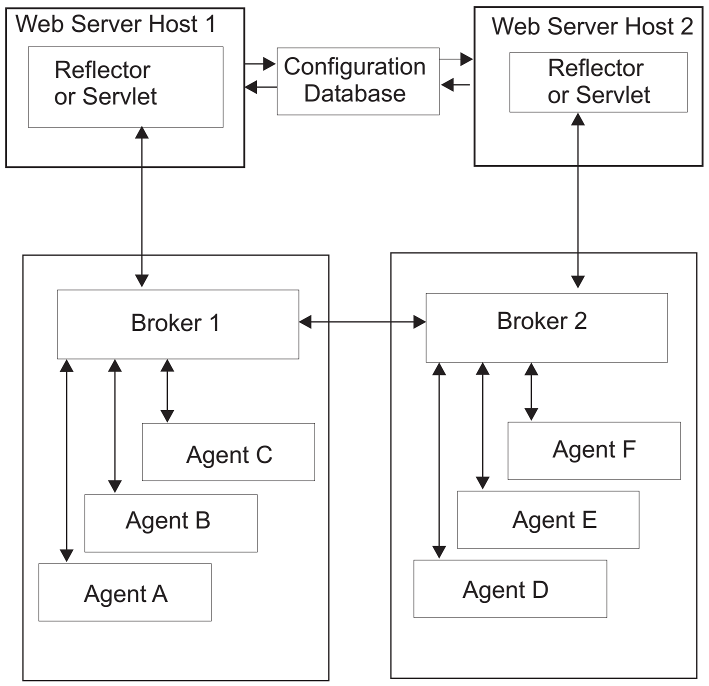

Figure 5. Partitioning users across brokers

Note that adding Web servers is not likely to be economically feasible if you do so for the sole purpose of load-balancing the brokers. However, if multiple Web servers already exist, then pointing them at different brokers is highly advantageous.

For Notes Client users, you can set up a single Web server to distribute Extended Search requests across a number of different brokers. This enables you to share the entry broker's burden across multiple brokers. To accomplish this, use the configuration file for the Extended Search Reflector to tell the Reflector about the existence of multiple brokers. The server sends each incoming message to a broker that it randomly selects from the set of specified brokers.

Figure 6 on page 21 depicts the schematic for a single Web server and two brokers, where all three entities are running on separate machines. Note that there is nothing about this schema that prevents a Web server from running on the same machine as one of the brokers it may select.

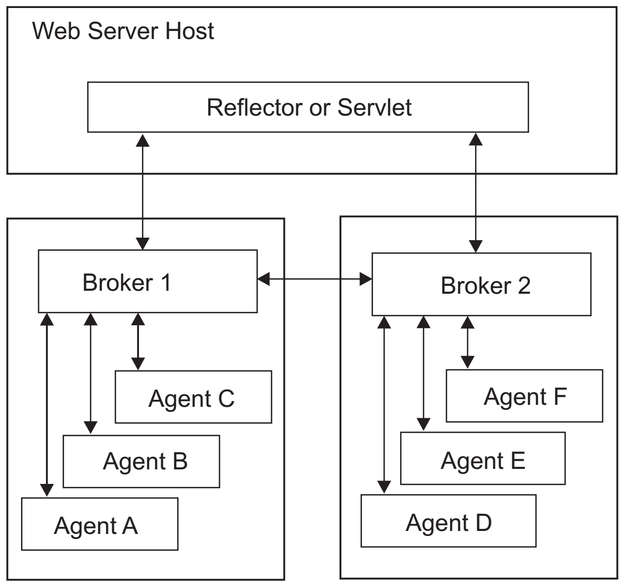

Figure 6. Using a single Web server to load-balance multiple brokers

Broker network configurations

There are many ways that you can install and configure Extended Search to satisfy the resources available in your enterprise. Some of the network configurations you might consider include the following:

- If you are a small shop, or running Extended Search in test mode, you might install all the Extended Search server components on the same machine as your Web server. In this scenario, a single broker handles all search request processing.
- To balance some of the processing load, you could separate the broker from the Web server. In this scenario, the server that hosts the broker handles all search request processing, and the Web server handles all requests for Web and configuration database processing.
- You could set up multiple brokers, and allow the same Web server to service all of them. This scenario enables you to further balance the processing load by configuring applications to use a particular broker as an entry broker. Each broker handles the search request processing for the data sources for which it is responsible. The entry broker aggregates a final hitlist for its own local sources plus the hitlists returned from the other brokers.
- You could set up multiple Web servers, and allow them to service multiple brokers. In this scenario, particular applications funnel requests through a particular Web server. The Web server distributes the requests to the appropriate entry broker, as defined in the application, and shares the workload among the various brokers.

By separating brokers and Web servers, setting up multiple brokers, and separating brokers and agents, you can scale Extended Search up to your existing hardware limits. After a certain point, which varies for each enterprise, the organization of the multiple brokers will itself become an issue that requires system administrator action.

For purposes of this discussion, Figure 7 defines a **broker unit** as a broker, its co-located sources and agents, and the users for which this broker is the entry broker. The broker unit, which is depicted as a hexagon, is the center of all normal broker and agent activities.

As depicted in the figure, there are three different network topologies for which you can configure a broker unit. These include a Fully Connected Network, a Hub Network, and a Multi-Hub Network. The following topics discuss each network configuration and the motivations for implementing each one.

To assist you as you configure brokers, the Administration interface provides tools to display your network configuration in a graphical view. After adding a server to the Extended Search domain, you can position it with an icon that best depicts how the broker on that server interacts with other brokers in the domain. You can also draw connector lines between the server icons to easily establish communication links between them. For information about adding servers and graphically connecting them, see *Extended Search Administration*.

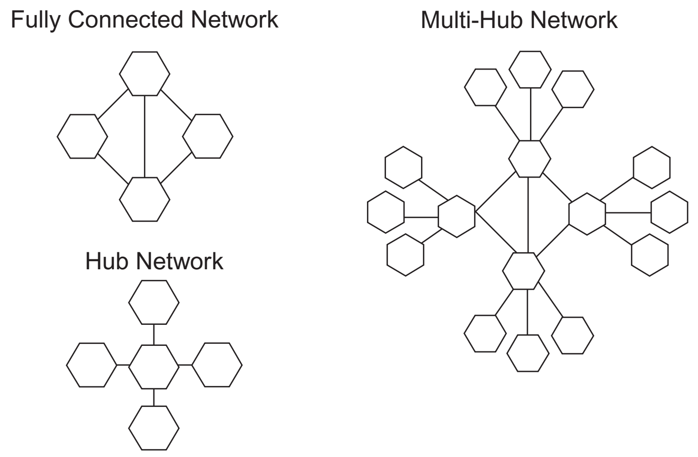

Figure 7. Fully connected, hub, and multi-hub networks

Fully connected network

The fully connected star network, as shown in Figure 7, consists of four broker units. An incoming query to any one of these brokers gets dispatched appropriately (based on the settings in the configuration database) to the other remote brokers. The remote brokers then expand the request and send individual queries to each source's co-resident agent.

Each remote broker consolidates the hitlists from its local sources and passes them back as a single hitlist to the entry broker. The entry broker consolidates its local hitlists with the hitlists returned by the remote brokers. This configuration involves a minimal amount of administrative setup.

Hub network

Another common network is the centralized hub illustrated in Figure 7. There is one hub broker unit that knows about all other broker units. Each of the other broker units knows only about itself and the hub broker. This hierarchy is most appropriate when there is one machine in the enterprise that is much more powerful than all the others. Typically, the hub broker will have no local sources.

The hub network is elegant, easy to understand, and not too complicated to set up. Its major disadvantage lies in the bottleneck of the hub broker. The hub broker must be extremely powerful to keep from being overloaded and, consequently, slowing down every request and response. Note that, among other things, the hub broker is required to do the largest part of hitlist consolidation for all queries.

Multi-hub network

The multi-hub network shown in Figure 7 on page 22 consists of a star network that comprises multiple broker units arranged as a hub. Each broker unit of that hub is the sole entry point for a number of other rim broker units. Again, it would be desirable to have the fastest machines constitute the hub broker. One design goal is to have roughly as many rim broker units connected to each hub broker unit as there are total hub broker units. The hub brokers would probably not have any local sources.

Such a hierarchy has the advantage that load sharing is almost automatic. If a query is directed against every single defined source, each broker would send out roughly the same number of requests. It would receive the same number of responses, and consolidate the same number of hitlists, as all of the other brokers. If requests are coming in evenly across all the rim brokers, then the distribution of messages will keep any of the brokers from becoming a bottleneck.

Performance

In an increasingly complex information technology (IT) environment, it is important to maintain a balance between providing users with the best technology available to help them do their jobs, and operating within the enterprise budget. The basic ingredients of the IT budget include computer hardware and software. An enterprise must combine these components in the right proportions to service end users.

It is important to plan hardware and software upgrades that account for future increases in work load:

- Hardware is a deterministic resource that is easy to quantify for planning purposes. Definitive answers include how fast the computer is, how much memory it has, or how much disk capacity it has. It is difficult to be as definitive about the software that runs on a computer network.
- Software performance stems, in large part, from a program's ability to efficiently and intelligently use a computer's resources. Even the most efficient software will, however, eventually exhaust computer resources as the number of users or the amount of activity increases.

Extended Search has been engineered to accommodate an enterprise's increased performance requirements. The following topics should provide you with sufficient information to optimize overall search performance. You should review this information in conjunction with "Scalability" on page 13, which outlines approaches that can help you expand the Extended Search system to accommodate growth. Discovering your enterprise's optimum settings is an iterative process, one that involves monitoring server resources and making appropriate adjustments.

Following guidelines that may help you assess your performance needs in general, review the following ways that you can optimize search activities in an Extended Search system:

- "Optimizing end-user performance" on page 26
- "Optimizing server performance" on page 28

Assessing performance

Before making adjustments to your Extended Search system, obtain an estimate of the domain's current performance to serve as a baseline for future comparisons.

The Extended Search Monitor allows you to identify areas where system performance can be improved. It provides a summary of search statistics that lists all of the target data sources serviced by an Extended Search server. For each data source, the Monitor provides the total number of search and retrieval requests. It also shows the average overall response time for the requests to complete.

If one data source has a relatively high average response time, then it is a good candidate for investigation. You should examine both the source database (and its search engine) and the configuration settings that define the source and its link.

The Monitor also displays server task statistics. Each type of server task can have one or more copies of it started by the Extended Search server. Increasing the copies of a task increases the number of requests for service that the server can process at the same time.

You can display the Server Tasks window in one of two views, Detailed or Summary. When you view the Server Tasks window in Summary view, you see a list of server tasks and a status light for each copy of a server task. Watching the conditions of the status lights (text and color) for a server task can reveal opportunities for system improvement.

If a task remains idle (gray), even under heavy system load, you may be able to reduce demand on system resources by decreasing the copies of that task. Conversely, if all copies of a task are busy (green), adding copies of this type of task may eliminate bottleneck conditions.

“Optimizing server performance” on page 28 describes the precise tuning of server tasks. For now, it is important to be aware of the significance of the statistics that the Monitor can provide.

Optimizing end-user performance

Many factors can impact the performance of a sophisticated search tool and application platform like Extended Search. Factors to consider include the following:

- Number of users
- Actions the users are performing
- Type and design of the client applications
- Hardware platform for both client workstations and server machines
- Operating systems for both client workstations and server machines
- Network protocols and access methods
- Server deployment topology

While search performance is, in large part, directly related to the power and performance of the Extended Search server, there are some discretionary measures that the end user can take to improve the overall response time of a search. There are also steps a system administrator can take to optimize the performance of individual servers.

When using an Extended Search client application, users may choose to submit either simple searches or advanced searches. Simple searches provide for keyword matching on all or some of the words contained in a search phrase. Advanced searches allow the formation of GQL statements of varying complexity.

Complex queries consisting of deeply nested Boolean expressions can result in the serial submission of multiple partial queries, thereby taking longer to complete than simple queries. This occurs if the target data source does not support nested operations. Under these conditions, the agent automatically compensates for this deficiency by first submitting multiple partial queries. It then calculates the union or intersection of various sets of results to create the hitlist for the entire query.

By accessing Extended Search through the Internet, users interact with servlets through a standard Web browser. When using a Notes Client, results of the search are stored in a Notes database. When using a Web Client, results are formatted as HTML and displayed directly in the Web browser. Depending on options included in the search form, users may be able to save the query and search results to share with other users.

The following techniques can help you optimize response time for your user community:

- Replicating databases.
- Setting timeout values.
- Sorting search results.

Replicating databases

One optimization technique for the Notes Client is to replicate the application on the user’s workstation. Replication is generally recommended as a way of decreasing overall response time when maintaining a continual connection to the network is not a concern. If users do not need to share queries and query results with others, then using a local copy of the Notes Client application is the most efficient choice.

When a user runs a local Notes Client application, search results get stored on the user’s workstation. Because the local application is not shared with others, the user can be assured that all application processing is dedicated to their searches only.

Setting timeout values

Users also have the ability to set workstation timeout values. Timeout values (in seconds) represent the limits that a user is willing to wait for a request to complete.

Take care when setting timeout values. You may set a certain value to accommodate a database that is notoriously slow. On the other hand, unusually long timeout values can result in very elongated response times. If you use the Notes Client application, the longest responding database or the timeout value determines the total response time, whichever occurs first.

Timeout Example

If a search request is made against 100 data sources, 99 of which performed their searches within five seconds and 1 which performed its search in 20 seconds, then the total response time will be 20 seconds. If the timeout value is 10 seconds, the results for the first 99 data sources will arrive on time. The results for the 100th data source will never arrive because all outstanding requests after the time limit, in this case 10 seconds, are ignored. If the timeout value is 4 seconds, then the timeout will occur before any of the data sources return results. The user will receive zero hits, along with a warning that the query expired.

Sorting search results

Another time saving technique for Web-submitted queries is to have the results returned the moment they become available. In this scenario, the user selects a suitable, and preferably small, hitlist page size. The system returns results as soon as one page has been filled. Thus, it is likely that the user will receive some results quickly, even if all of the data sources do not return results within the allotted timeout period.

By the time the user has examined the first page of hits, it is highly probable that the next page has been filled and returned. This technique can have dramatic effects on reduced response times when the number of sources to be searched is very large.

Sorting Example

You can modify the previous example to reflect that at least 10% of the 100 data sources will return three or more hits within 1 second. Assume that the page size for the hitlist is 5. It is likely that users will experience a 1–second response time for the first page of results and subsequent pages thereafter. When using a Notes Client application, users would need to wait 20 seconds for the server to receive and sort all of the results. When using a Web Client application, users would wait 20 seconds only if they asked for results to be sorted instead of receiving results as they become available.

In general, it is good practice to choose as sources only those that are deemed necessary for a search request. Blanket searches across all data sources may result in varying, partial, or no results depending on the system workload and the timeout values that are set for a workstation. Blanket searches also increase the total workload on the system and, as such, degrade overall performance.

Optimizing server performance

The discussions, so far, have covered the performance optimization techniques available to client applications and end users. The remainder of this discussion concentrates on adjustments that you can make to a single Extended Search server to improve overall response time. “Scalability” on page 13 describes how to use multiple Extended Search servers to improve performance when the capacity of a single machine has been exceeded.

In a single server installation, one host machine runs not only the Extended Search broker, but also the agents, the Web server (and the Reflector or servlet by which it communicates with the server), and the data sources and search engines with which the agents communicate.

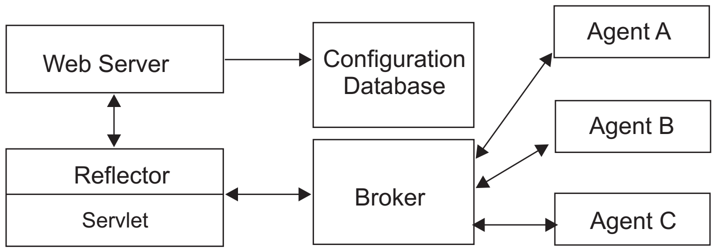

Figure 8. Components on a single host

This arrangement of components is useful for demonstration purposes, exploration of Extended Search capabilities, and small workgroups that search across a limited number of sources. For large enterprises, where the number of users and sources grow over time, a single server becomes overburdened as it attempts to meet the usage demands placed on it.

The following techniques can help you optimize Extended Search server response time:
- “Understanding server tasks” on page 29.
- “Determining the optimum number of tasks” on page 30.
- “Localizing agents with data sources” on page 31.

Understanding server tasks

The Extended Search server comprises a group of programs that work in conjunction with one another. Figure 9 depicts the server's internal schematics.

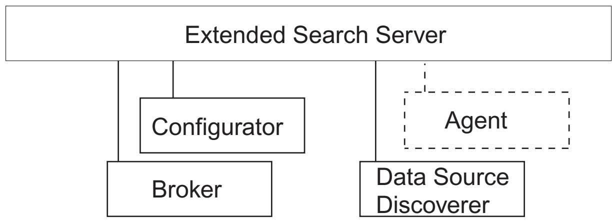

Figure 9. Constituent parts of the server

The individual programs are known as **server tasks**. A server task is an independent program that responds to requests for service. Server tasks execute under the control of the server. Other components contact them through an assigned network address (host name and port number). The Server Tasks window of the Extended Search Monitor displays the processing activity of all the tasks operating under a given Extended Search server.

Extended Search includes the following server tasks:

- The **Internal Configurator** interfaces with the Extended Search configuration database to provide configuration information to other server tasks. There should be no more than one copy of this task per Extended Search domain.

  The Configurator regulates the server tasks and ensures that the appropriate tasks get invoked in the proper order. It also translates data source names into network addresses. Sources targeted by a query are in names understandable to the end user. The agent responsible for searching a designated source is reachable through a network address. This concept of translating names to network addresses is key to being able to make scaling upgrades.
- The **Broker** performs the following key functions:
  - Distributes search requests to the appropriate set of agents, and then waits for the return of responses. The broker aggregates, sorts, and prunes the search results, and then returns them to the client application.
  - Distributes fetch requests to the appropriate set of agents, and then waits for the return of responses. It aggregates documents into a single message, and returns it to the client application.
  - Collects and consolidates messages from other server tasks, and decides which, if any, should be reported as events. You can view Extended Search messages through the Event Log Viewer in the Administration interface.
- The **Agents** perform the actual search or retrieval against a given data source. Figure 9 depicts agents in a dotted box to reflect the fact that agents can run on the same machine as the Extended Search broker, or run independently on a separate machine. See "Localizing agents with data sources" on page 31 for a discussion of running agents remotely from or co-resident with a broker.
- The **Data Source Discoverer** manages the data source discovery processes. Normally, two copies of this task per Extended Search domain is sufficient.

Determining the optimum number of tasks

You can have more than one copy of the broker and agent tasks started by the Extended Search server. Each copy of a server task is capable of processing a single request for service. Additional requests will be queued and then processed in first-in-first-out (FIFO) order. Increasing the copies of server tasks increases the number of requests for service that the system can process at the same time.

For the broker, the settings you choose determine the overall effectiveness of a broker's ability to handle requests from users. In general, if the number of server task copies increases, the number of simultaneous requests from users that a broker can process in parallel also increases. Likewise, the greater the number of agents, the greater the number of sources that the system can search in parallel.

Minimally, it is a good practice to have at least two copies in each type of server task. This provides for some automatic backup and redundancy, so that an unforeseen failure in one type of server task would not disable the entire system. The one exception to this rule is when setting the copies for the Internal Configurator. For this special server task, there must never be more (or less) than a single task in each server.

Table 1 shows the recommended copy limits for each type of server task, given the resources of the machine on which the broker is running. Note that if other full-time running programs (such as a Web server) exist on the same machine, you may need to reduce these numbers for most effective performance. Exceeding these guidelines will not break the broker. However, the additional cost in operating system overhead for each additional task will begin to interfere with overall performance.

The first column of numbers for a single 350 MHz processor with 256 MB of memory is the default configuration for a single server installation. The only exception is that a single server installation has no stand-alone agents, and thus that row is irrelevant.

Table 1. Suggested Limits on Server Task Copies

Memory	256 MB	512 MB	1 GB	2 GB
Processors	1 @ 350 MHz (default)	1 @ 550 MHz	2 @ 350 MHz	4 @ 400 MHz
Internal Configurator	1	1	1	1
Data Source Discoverer	2	2	2	2
Broker	5	15	20	30
Agent (with Broker)	15	20	30	50
Agent (stand-alone)	30	40	50	70

The total number of broker tasks determines the practical number of simultaneous end-user requests. When all of these tasks are busy, additional requests get queued until an appropriate server task becomes idle.

The speed with which search tasks can finish a query is dependent on the number of sources involved in the query. It also depends on the number of agent tasks available to search those sources, and how long it takes the data source to respond.

If you define five copies of the agent task, and define each end-user request to run against fifteen sources, then the first query would tie up all of the agent tasks for up to three search cycles before the second query could even begin to search. Meanwhile, queries three, four, and five would be in the queue waiting for their turn.

As the Extended Search server initializes, it starts the defined number of copies for each type of server task, and then listens for client requests. The installation process sets the number of copies for the respective broker tasks to default values. For a higher powered machine, it is almost certain that you will need to adjust these initial numbers.

For instructions on how to configure multiple copies of broker and agent tasks, see *Extended Search Administration*.

Localizing agents with data sources

Extended Search agents and the data source engines they connect to are generally the largest consumers of machine resources. Their consumption can vary widely from enterprise to enterprise, depending on the number and type of databases to which the agents link.

When using Extended Search to search Notes databases, it is particularly beneficial for the Notes database and the agent to be co-resident on the same machine. The performance improvement in this case is particularly large. To put it another way, the performance degradation from not making the link local to the database is very costly. Therefore, every searchable Notes database should have its associated agent installed on the same machine. If this is not possible, then you should replicate the Notes database onto a machine on which you can install a local agent.

Use of replicated databases can also alleviate potential problems. In an environment where users can query against several databases, some of which are not under your control, the owners of the databases may make changes that prevent the agents from locating the databases. For example, an administrator may move the database to another machine or change the name of the database. There may also be other enterprise applications that use these same databases and generate a lot of other activity.

You can alleviate these problems by replicating remotely hosted databases onto a local machine. Once replicated, you can maintain local copies with a constant name and address. If a database name and address must change, you should make the corresponding changes in the configuration database. For example, scalability may require you to split several replicated databases between two machines.

You can also direct the query load to these local copies. This action can improve search performance and reduce the impact of the search on other applications.

The **LNServer** data source parameter tells the agent how to access a Notes database — either locally or on a remote server. For instructions on how to configure the LNServer parameter, see *Extended Search Administration*.

Printed in U.S.A.