*S*ystem & *H*uman *R*esource *L*ocator

Sherlock

Requirements

Revision A

August 23, 1998

1 Introduction

1.1 Purpose, Scope, and Field of Application

1.1.1 Purpose

The purpose of this document is to specify the requirements of Sherlock, a graphical, client/server-based, platform-independent online query tool for identifying and locating enterprise resources through an enterprise directory.

The trade journals and in-flight magazines would have you believe that communicating in today's corporate environment has been facilitated with the advent of voice mail, electronic mail, web servers, cellular phones, and pagers. Ironically, while it is technically easier to get in touch with a colleague, it becomes a onerous task finding out *how* to best reach that colleague. Coupled with the proliferation of operating systems and computing platforms, all this technology seems to contradict the claims of trade journals and in-flight magazines: Leaps in technology are making it tougher to communicate. We need something that independent of operating systems and platforms, intuitive, useful, very fast, and capable of scaling to the growth of our organization. An example of how we might do this is below.

Using a browser, we want to be able to search information on a colleague. We want to be able to get e-mail addresses, telephone number, extension, and maybe even a pager number, cellular phone number, or personal web site. We should also be able to get his mailing address, mail stop, his job title, the name of his workgroup and division, and his supervisor's name. In case we want to book a meeting or do lunch, I want to be able to view his calendar/personal scheduler.

From there, we should be able to click on his supervisor's name and get the same sort of info. We should be able to click on his e-mail address and automatically be able to type a new mail message. We should also be able to click his pager and send either a numeric or alpha-numeric page.

For that same person, we want to see where his office is located . We want to be able to look at a birds-eye view of the facility in which he works and see an X in the general vicinity of his office. Furthermore, we want to be able to zoom in on that X and see one or two levels of granularity of the vicinity, such as the numbers of nearby offices, conference rooms, copier machines, videoconference equipment, even restrooms!

Completing this thought, we want to be able to look at our colleague's office itself. We want to see in which corner he sits. From there we can see devices, such as a telephone, computer, and a printer. When we move our cursor over any of these devices, we receive pop-up information about that device such as its IP address, machine name, brand and model. Not only can we determine the brand and model of these devices, we can also see to which wall jacks these devices are attached. From those wall jacks, we can also determine what rack number to which it corresponds back in the wiring closet. From that rack number, we can ascertain to which port on which hub/switch the device is attached.

We would also like to extend this beyond personnel. We should be able to use this same browser-based tool to relationally search for anything: a printer by name, or a person by a jack number, or an IP address by office number. Important too, is that we should be able to view a map of any item that is associated with an individual, a location (facility, office, room), and/or a device.

From that map, we should be able to click on any conference room and see if it is equipped with a speakerphone or live data jacks. Moreover, we should be able click on "Reserve this Room" to send an e-mail message to Facilities to reserve the room for a particular date and time.

And all this information is kept in a general-purpose enterprise directory. What we need is a server agent that makes all that information meaningful and wraps it up in a slick, intuitive interface for a platform-independent client. This is what we would like to call Sherlock.

Not only is Sherlock helpful, but it supports internationalization, i.e., multiple languages, such as English, Finnish, French, and Spanish.

1.1.2 Scope

The intent of this document is to provide students of CS406/CS407 with the necessary specifications to fully design, test, and implement Sherlock. This document will also serve as a means for tracking progress against the schedule provided by Tellabs and Dr. Mathur.

1.1.3 Field of Application

Once designed and tested, Sherlock will be implemented as an important part of Tellabs intranet infra-structure with positive implications for roughly 6,000 end-users and administrators both, across 40-plus different locations around the globe.

1.2 Revision History

Revision	Date	Description
AP1	7/6/98	Initial version
AP2	8/14/98	Update of server description
AP3	8/21/98	General updates, formatting
A	8/23/98	First official version

1.3 References and Attachments

LDAP World™ provides current information on the status of the LDAP specifications, availability of LDAP products, and deployment of LDAP-based directories. It is a free and unsupported service provided to the Internet community by Innosoft International, Inc. *http://www.critical-angle.com/ldapworld/*

X.500 Directories, hosted by NEXOR Ltd, Nottingham UK. *http://web.nexor.co.uk/users/cjr/x500.html*

LDAP, RFC-1777 (March 1995). *http://www.innergy.com/RFC/rfc-1777.html*

Timothy A Howes, The Lightweight Directory Access Protocol: X.500 Lite (July 27, 1995). http://www-leland.stanford.edu/group/networking/directory/doc/ldap/ldap.html

Timothy A Howes and Mark C Smith, A Scalable, Deployable, Directory Service Framework for the Internet (April 28, 1995). *http://info.isoc.org:80/HMP/PAPER/173/html/paper.html*

Web site for the Official Guide to Programming with Perl CGI.pm. Software is entirely free. *http://www.wiley.com/compbooks/stein/distribution.html*

1.4 Definitions, Acronyms, and Abbreviations

Abbreviation	Definition
LDAP	**Lightweight Directory Access Protocol**
CLDAP	**Connectionless Lightweight Directory Access Protocol**
IT&S	**Information Technology & Services**
OS	**Operating System**
HTTP	**Hypertext Transmission Protocol**
HTML	**Hypertext Markup Language**
GUI	**Graphical User Interface**
URL	**Universal Resource Locator**
IP	**Internet Protocol (refers to addresses of networked devices)**
CD-ROM	**Compact Disc-Read Only Memory**
JPEG	**Joint Photographic Experts Group (refers to image files)**
GIF	**Graphics Interchange Format (refers to image files)**
NPA-NXXX	**Numbering Plan Area (refers to telephone numbers)**

2 REQUIREMENTS OVERVIEW

Electronic directories come in many different forms, designed for many different purposes. All types of directories have a common characteristic, which is that they hold information about objects. Objects can be almost anything about which one would want to store and retrieve information, such as persons, organizations, computer applications (on-line services), and network components.

Today, the key driving force behind general-purpose enterprise directories is for providing a central corporate repository for commonly and widely used information. This includes information about employees of the enterprise for example white pages data (email addresses, phone numbers) and information enabling access to services (printers, computers, buildings). In addition, authentication, encryption and digital signatures are frequently required for secure communications. Directories were designed to support PKI (Public Key Infrastructure) which provides these facilities.

Providing a directory using open directory standards enables a wide range of enterprise applications to interface with a single enterprise directory.

When reviewing this document, unless a feature has been explicitly drawn out, feel free to make assumptions as to its look, feel, and functionality. Creativity is encouraged; be sensitive to the constraints it might impose, particularly on the timeline. Where assumptions have been made, be sure to note them and be prepared to discuss at next meeting.

Also understand that requirements are subject to change, sometimes at the most inconvenient times.

3 SYSTEM AND ENVIRONMENTAL REQUIREMENTS

The requirements in this document define the highest level specifications for Sherlock. This includes platform and source language restrictions, as well as certain implementation details of the core and peripheral features.

This document is not intended to be a complete design of Sherlock. Rather, it ought to be viewed as a framework in which this system will be designed, tested, and implemented.

3.1 Platform Requirements

3.11 Server

In the server environment, Tellabs IT&S supports the operating systems of Sun (SunOS 4, Solaris 2.x) as well as Microsoft NT Server (NT Server 3.51 and 4.0). Since remote administration of Sherlock is a requirement and NT Server is not robust in its remote login capabilities, it is preferred to build Sherlock on a supported and current Unix operating system, e.g., Solaris 2.x, Linux 2.0.x.

3.1.2 Client

In its client environment, Tellabs is moving to cross-platform and platform-independent software solutions. As a result, it is required that the client operates from any SSL-capable browser.

3.2 Source Language Requirements

Sherlock may be implemented in variety of languages, restricted only by those which have wide commercial support. Some recommended languages are Java, Perl 5, Tcl/Tk, HTML, and C.

All source code must be delivered to Tellabs at the conclusion of this project. In addition, it must be able to be modified, compiled (if appropriate), and executed.

3.3 Timing and Speed Characteristics

With regard to timing and speed, Sherlock must be able to handle multiple, simultaneous client queries. All client queries should be processed (results returned) under ten (10) seconds, with a moderate server load, network traffic notwithstanding. Ten seconds has been deemed the longest any Tellabs user should have to wait for anything!

Sherlock must be able to update swiftly, under a moderate load, from its parent LDAP server. In addition, Sherlock must offer a configurable timing option for scheduling updates from the parent server.

4 CORE FEATURE REQUIREMENTS

4.1 GUI Requirements

4.1.1 GUI Overview

The look of the GUI is less important than the functionality. For all administrative and end-user functions, a browser-based, SSL-capable, GUI is mandatory.

4.1.2 Browsing (Sherlock server and client)

All system configuration options must be browser-based, SSL-capable, and graphical. Pull down lists and buttons should be intuitive and clearly labeled. For a good example, refer to Netscape’s commercial server products; all servers have a browser-based admin GUI.

A screen presents a directory hierarchy to the user, which can be followed intuitively by clicking on URLs. All screen data is automatically generated from the directory.

4.1.3 Presentation of Entries

A directory entry can be presented as desired. As an example, e-mail addresses can be mapped to mail URLs; jpeg/gif images can be displayed; related entries can be presented as hyperlinks, and so on.

Regardless of the browser, both administrative and end-user presentation should still be useful and be pleasing to the eye.

4.1.4 Navigation

Both hierarchical browsing and direct access should be supported. A portion of the window, be it the title bar or side-bar, must be updated to reflect the current location of the user's directory access, in relation to the top-most portion of the global directory.

Windows should also offer "hot links' which are defined for direct access to specific directory entries, such as commonly-accessed organizational entries.

User-selected directory entries must be accumulated so that a user need not browse the entire hierarchy to return to a recent point in the directory space.

A separate "history" list of recently accessed entries must also be maintained.

4.2 Directory Service Requirements

4.2.1 Open Client/Server Access

To build a general-purpose infrastructure, open to a wide range of applications, open access is required. Widely supported protocols for these are LDAP, CLDAP, HTTP, and X.500 DAP.

4.2.2 Robust and Secure

A directory solution will be considered mission-critical. As such, data must never be lost or corrupt. Since data contained within the structure is sensitive, appropriate security must be used.

4.2.3 Fast

Since applications will be depending on directory access, access has to be very fast. The server must be able to support large numbers of concurrent connections and be highly-available.

4.2.4 Scalable and Distributed

In order to support the size and scale of Tellabs, the directory must provide replication and distribution.

4.2.5 Open Data Management

Ties into 4.2.1 above. Multiple applications operating in multi-platform environments will require application-specific tools for management.

4.3 Administrative Requirements

4.3.1 Customization

Views of information, as presented by Server, should be easy to customize using a common scripting language. This would enable support of different presentation requirements for different user communities, such as a "restricted view" for users on an extranet.

Mapping of directory attributes to specific HTML elements may vary based on the need of a distinct user community. Server scripting mechanism should support this.

4.3.2 Authentication and Login

Users should be able to login to directory via the Sherlock server.

Users should be able to customize their login entry, rather than have to use LDAP/X.500-compliant Distinguished Names.

4.3.3 Modification

Server must support data modification by users for specific portions of their own data. Options to modify data will only be presented when Server determines whether user has “modify rights.”

4.4.4 Forms-based Searching

Server must be configured to use a forms-based approach to searching for anything. The form that is presented is controlled by a simple script. That script should control the layout of the form and the user selection of options used to search the directory.

4.4 Elemental Requirements

4.4.1 Overview

This list is not exclusive. The following are the main “buckets” with their respective contents i.e., related items.

4.4.2 Personnel

Firstname Middle Initial Lastname, extension number, phone number (+1 npa-nxxx), e-mail address, mailing address, title, department, division, office number (based on office coordinates), device name(s), map.

4.4.3 Devices

Device name, device family name, description, IP address, facility name, jack #, map

4.4.4 Facilities

Facility name, facility address, facility photo, facility contact, map

4.4.5 Offices

Office numbers (based on office coordinates), description, facility, jack #s, map

4.4.6 Jacks

Jack #s, description, rack #s, port #s, map, office (based on office coordinates), device name, device family name, IP address (if applicable), extension number (if applicable), telephone (+1 npa-nxxx if applicable).

4.4.7 Ports

Port #s, description, rack #s, jack #s, device name, device family name, IP address, , IP address (if applicable), extension number (if applicable), telephone (+1 npa-nxxx if applicable).

4.4.8 Maps

Facility name, facility address, office views (zoom 2x/4x), and office numbers (based on office coordinates).

4.5 Map Requirements

4.5.1 Overview

Maps are not to be considered an afterthought of Sherlock. They will be looked to "pack the punch" with regard to Sherlock's functionality for end-users and administrators alike.

4.5.2 Bitmapped

Images should be as "lightweight" as possible. They also must be printable. Sherlock should be able to accommodate a map as if it were any other object.

4.5.3 Meaningful

Maps must make sense; grid systems should be well-coordinated; the appearance of maps should be deemed "very good."

4.5.4 Facilities

Facilities should be coordinated via a simple indexing system that makes them unique, yet scalable to accommodate growth.

4.5.5 Offices

The generic term "office" will be used for any room that has a numeric designation within a facility. Office numbers will be pre-assigned by facilities. However, some sort of grid or other suitable numbering scheme must be developed that (a) works within Sherlock and that (b) will scale. Once accomplished, each office will be assigned coordinates. Coordinates will accompany office information on the Server.

4.5.6 Overlays

Overlays will accompany maps and be used to designate personnel, jacks, and office locations. Overlays will be color-coded, intuitive, and meaningful. Overlay elements will support "mouse-overs" so that relevant information about an element pops up on the screen. A textual view (non-pop-up view) will also accompany the overlay for purposes of printing.

5 PERIPHERAL FEATURE REQUIREMENTS

5.1 Administrative Add-ons

5.1.1 Support for vCard and vCalendar

An enhancement of Sherlock is the ability to generate a vCard representation of all personnel. This vCard will be suitable for placement into a contact manager or personal address book. Similarly, support of vCalendar on Sherlock will allow support of an enterprise calendaring scheme. Such a scheme would make possible, via Sherlock, browsing of a person's public calendar information and integrated support of calendar requests, such as scheduling a meeting.

5.1.2 Specialized Tools

Specialized applets could be launched from a hyperlink to accomplish things ranging from looking up *only* the phone number or e-mail address of a person, to scheduling a meeting electronically without user intervention.

5.2 Functional Add-ons

5.2.1 Shared Address Books

Use of Sherlock to support shared address books which would allow users to easily share telephone numbers and e-mail addresses of a common set of people, like customers or suppliers.

5.2.2 Business Partners

If a business partner (or merger/acquisition target) operates a directory based on open protocols, it is possible to link the two together. If both directories use a common schema and are a part of a global naming structure, rapid integration would be possible. This would enable very efficient operations.

5.2.3 Receptionist

An applet for those who "always gotta answer the phones" or "deal with visitors." Quite simply, an operator could perform extension number lookups using Sherlock.

At the same time, a receptionist could identify the person being visited. Ideally, the person being visited might even have this visitor booked in his shared schedule. This enables an automatic visitor check-in system.

For security purposes, Sherlock can be used to provide information about staff for security verification, e.g., photographs or "mother's maiden name." This would be particularly helpful in an office where "hoteling" is prevalent: Allowing identification of staff and assignment of resources.

5.2.4 Human Resources (HR)

Beyond just basic information about employees, HR could use Sherlock to handle more-structured information, such as employee numbers, Social Security numbers, home address, and holiday information.

Employees could then verify recorded holidays and change home address information. Managers too, could have access to information about employees where appropriate.

5.2.5 Resource Access

Both physical access to the building and online access could be integrated through Sherlock, allowing Facilities and IT/IS to manage access to through the same interface.

Furthermore, Sherlock would provide the management of single sign-on and general access. Future technologies such as smart cards, certificates of authority, and public key information could be managed and served via Sherlock.

5.2.6 Enterprise Application (SAP) Integration

At some point, it would be advantageous to have tight integration of SAP with Sherlock for workflow and other office automation functions, such as Purchase Requests, Expense Reports, Travel Requests, and Employee Benefit Programs, e.g., 401(k), Employee Stock Purchase.

5.3 Media Requirements

5.3.1 Overview

Thus far, Sherlock presumes that a network connection exists between the server and client. However, there are instances when Sherlock must function in a connectionless or standalone environment.

5.3.2 CD-ROM

Sherlock ought to be transportable to CD-ROM. A disk could be burned with all the features and functionality normally available through a network connection. All the data on CD-ROM would be encrypted. A user or administrator would simply insert the CD-ROM, fire up his browser, authenticate to use the CD, and have all the information (current as of the date of the disk's creation) at his fingertips.

This would presume that the contents that comprise Sherlock server and its data components do not exceed the 650 MB of storage available on disk.