A Practical Implementation of Single Sign-On using LDAP for Web-Based Applications

Meng-Hsien Lin, Shih-Chang Wang, Der-Jeng Juang and San-Wei Sun

Internet & Multimedia Application Tech. Lab
Telecommunication Laboratories
ChungHwa Telecom Co., Ltd.
Taoyuan, Taiwan

Address of Correspondent:

Shih-Chang Wang
Internet & Multimedia Application Tech. Lab
Telecommunication Laboratories
ChungHwa Telecom Co., Ltd.
Taoyuan, Taiwan

TEL: (886-3) 424-5340
FAX: (886-3) 424-4470
E-mail: purewang@cht.com.tw

Abstract

Traditionally, users have to sign-on to multiple systems, each of which may involve different usernames and authentication techniques. Consequently, such web applications force users to mange multiple user names and passwords daily and cause great inconvenience. In contrast, with single sign-on, users authenticate themselves just once to access information on any of several systems. Single sign-on has the following advantages: convenience to the user, improved security, administrative convenience, and reduced expenditure. *LDAP*(Lightweight Directory Access Protocol), the clear directory standard, leverage a single, master directory that owns all user, group and access control information. In this paper, we propose a practical web single sign-on implementation in *CHT*'s *EIP*(Enterprise Information portal) system for integrating web-based information systems together. This approach is based on *HTTP* cookie and the unified *LDAP* server. Compared with the existing methods, our approach has the benefits:1)simple to be implemented, 2) it is not necessary to modify the *LDAP* server's configuration substantially, 3) integration of the existing web-based systems and *LDAP* server, 4) no extra relational database is needed.

Keywords: *HTTP, Cookie, Single Sign-On, LDAP, Web-Based Applications, EIP, Portal*

Workshop: Workshop on Computer Networks

I. Introduction

Today, directories exist in a multitude of applications ranging from operating systems, asset management systems, *PBX*'s, badge security systems, and *HR* systems to email and database applications. The cost of implementing and administrating these disparate, proprietary directories is great because each one must be managed independently. This adds enormous administrative burden and cost to already strained *IT* budgets. Rather than pouring more money and resources into managing these systems, *IT* departments are requiring that applications support *LDAP*(Lightweight Directory Access Protocol), the clear directory standard[1-2, 14-15]. *LDAP*-compliant systems leverage a single, master directory that owns all user, group and access control information. This directory server becomes the central repository providing user, group and access control information to all applications on the network(see Fig. 1). Unified directories eliminate redundancy which lowers management costs. In addition, unified directories ensure that applications can run within and without of an organization so that partners, customers and vendors may participate in network applications where appropriate[7].

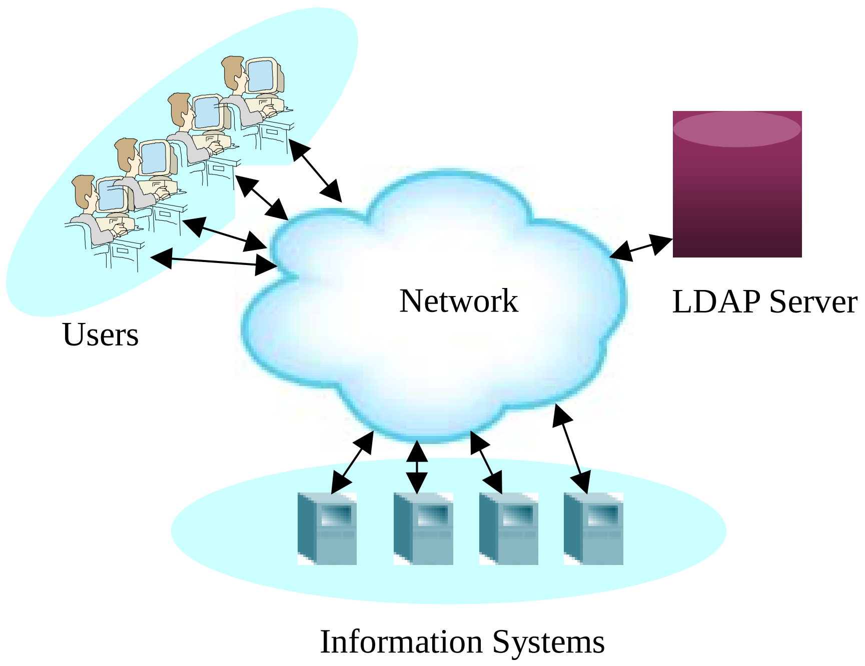

Fig. 1: An illustration for the unified *LDAP* server cooperating with information systems.

A directory is a hierarchical collection of objects and the attributes of the objects much like the subdirectories of a file system and the files contained in the subdirectories. Objects can have various attributes and numbers of the same attributes, unlike the columnar structure of a *SQL* database's "table". Generally, directory servers are typically optimized for a very high ratio of searches to updates. Generally, *LDAP* has the following characteristics: standard, unique naming, easy and fast searching, distributed architecture, and referral mechanism.

In *CHT*(ChungHwa Telecom Co., Ltd, Taiwan), we have built more than 35,000 entries of the employee and organizational-unit information into a Netscape *LDAP* server[16] successfully and provide directory services(for example, authentication, search and so on.) to more than 30 *CHT*'s various information systems. In order to facilitate the system integration onto the *LDAP* server, we also have developed the *generic high-level C/C++/Java/COM APIs*(Application Programming Interface)[9-10] for existing systems to connect the *LDAP* server easily.

More and more enterprise information systems are designed as the web-based models due to the convenience of *WWW*(World Wide Web). With the integration of web-based desktop and applications, the number of services a typical user can access has grown multi-fold. It is no longer acceptable to enter one's username and password so many times daily. *WSSO*(Web Single Sign-On) is necessary to provide users a single identification and authentication mechanism among web-based systems. In *CHT*, the *EIP*(Enterprise Information Portal) system is designed to ingrate all web-based systems in an office into a single web site. Obviously, *WSSO* is one of the most important functionality in the *EIP* system. As the name says, *WSSO* users log on once and the system takes care of the rest for them.

Many solutions about *WSSO* were proposed in the literature[3-5,8] and commercial products[11-13]. However, additional overheads are generally needed to complete the *WSSO*, for example, the *PKI*(Public Key Infrastructure), the password storing, the domain cookies[3,6] and so on. In this paper, a *WSSO* approach based on *LDAP* is given. The proposed novel *WSSO* scheme needs little overhead, and now is implemented in *CHT*'s *EIP* system. Furthermore, an implementation scenery is also described in this paper.

The organization of this paper is as follows. In Section II, the basic concept about *WSSO* and *EIP* is given. The proposed *WSSO* procedure is shown in Section III. In Section IV, the *WSSO* implementation in *CHT* is provided. Finally, conclusions are given in Section V.

II. Preliminaries for WSSO and EIP

The use of *LDAP* is gaining popularly in the world. *LDAP* provides the benefit of unified information and quick response. Recently, the famous newspapers and magazines publish that many enterprises or businesses have adopted *LDAP* server to manage the employee's information and to achieve the single account while accessing all information systems. Although the enterprises adopt *LDAP* to manage single employee's account, multi-signons are still needed when users access different web-based systems. Consequently, the user will feel inconvenience and tiredness, especially for numerous web-based systems in an office.

The *WSSO* is not only for achieving single account, actually that is an approach whereby the single behavior of user authentication and authorization can grant a legal user to access all information systems where he can access, without the need to enter the username and password repeatedly. Hence the advantages of *WSSO* are convenience to the user, improved security, administrative convenience, and reduced expenditure. For the user, with only a single username and password to remember for access to different applications, it would eliminate the sticky notes ubiquitous on the user's desks, which in turn, prevents passwords being stolen. For the IT administration, lesser overhead is taken in maintaining the applications. However, the *WSSO* scheme is not easy to be implemented. There are numerous *WSSO* products announced in the marketplace such as *Oracle 9iAS*[11] , *iPlanet* portal server[13], *IBM WebSphere* application server[12] and so on. The *WSSO* solutions provided among commercial products are quite different to each other. The features for the above approaches almost must change the existing *LDAP* server's configurations or attach to the extra relational databases. Clearly, such methods seem to have some limitation in implementation. In addition, some *WSSO* solutions depending on *PKI* or password-store[11] also have been proposed. Such kind of approaches may have the requirements of the client-side applications to be involved. A *WSSO* approach based on *HTTP* domain cookies[6] was also proposed in the literature[3], however, it seems hard to be implemented because the domain cookies is not functioned completely within browser for the security consideration.

During 1998, Internet Portals became very popular. These provide consumers with personalized points of entry (or gateways) to a wide variety of information on the Internet. Examples include *MyYahoo* (*Yahoo*), *NetCenter* (*Netscape*), *MSN* (*Microsoft*) and *AOL*. A Merrill Lynch report (published on November 16, 1998) was the first time that "*portal*" was also used for enterprises[17]. *EIP*s are applications that enable companies to unlock internally and externally stored information, and provide users a single gateway to personalized information needed to make informed business decisions. Generally, the *EIP* system is integrated with existing systems by listing the *HTTP* hyperlinks of those existing web-based systems on the web pages.

For the environment of a non-*WSSO EIP* system, the scenery for a user to operate should be as follows:

- The user opens a *WWW* browser and enters a suitable *URL*(Uniform Resource Locators) on his desktop *PC* to access the *EIP* system.
- The *EIP* system prompts a login shell that includes username and password form.
- After the username and password are entered, the user can acquire information from the *EIP* system.
- When the user wants to access other information systems through *HTTP* hyperlinks, a login page of the clicked system will be prompted to the user. The same username and password need to be entered by the user again in order to login.

From the above descriptions, sign-on actions are always needed when the user wants to logon other systems. Obviously, to achieve *WSSO* is one of the most important functionality of an *EIP* system. Therefore, to overcome such inconvenient procedure, we propose a *WSSO* solution. Our architecture is based on the *HTTP* cookies and the existing *LDAP* server. Compare with the existing *WSSO* methods, the proposed approach has the following advantages:1) simple to be implemented, 2)The *LDAP* server configuration does not be modified substantially, except one extra attribute named *signOnKey* is added, 3) No extra relational database is needed. The proposed *WSSO* procedure will be discussed in next section.

III. The Proposed WSSO procedure

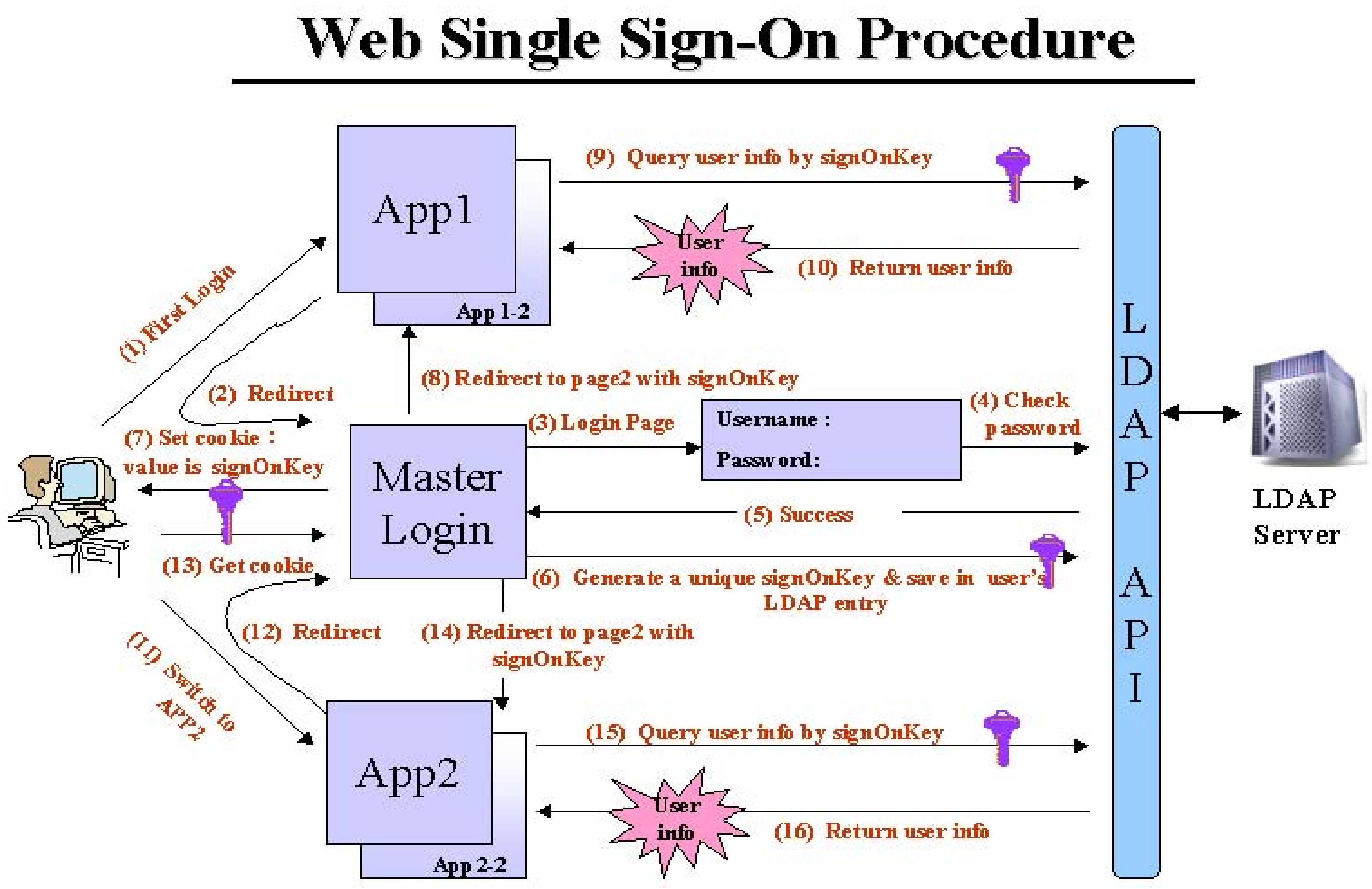
Web Single Sign-On Procedure
App1
App 1-2
(9) Query user info by signOnKey
User info
(10) Return user info
(1) First Login
(2) Redirect
(8) Redirect to page2 with signOnKey
(7) Set cookie : value is signOnKey
(3) Login Page
Username :
Password:
(4) Check password
Master Login
(5) Success
(6) Generate a unique signOnKey & save in user's LDAP entry
(13) Get cookie
(12) Redirect
(14) Redirect to page2 with signOnKey
(11) Switch to APP2
App2
App 2-2
(15) Query user info by signOnKey
User info
(16) Return user info
LDAP API
LDAP Server

Fig. 2: The proposed *WSSO* procedure.

In the proposed *WSSO* approach, user information(for example, user name, password, and so on) is stored in the *LDAP* server. Sign-on action is processed mainly by the Master Login page. The main idea of the proposed *WSSO* scheme is described briefly as follows. The unauthenticated access to some web-based system(called *App1)* is redirected to Master Login page with *App1*'s own *URL*. Note that Master Login page is a simple web page. If the entered username and password is authenticated by the *LDAP* server successfully, a random encryption key(called *signOnKey*) is inserted into *LDAP* server and user's browser cookie.

Then, Master Login page redirects this *HTTP* request to the *App1*'s *URL* with the *signOnKey*. *App1* uses this *signOnKey* as key to retrieve user information from the *LDAP* server. If the read from the *LDAP* server is successful, *App1* permits the user to access the system information, that is, this user is a legal user. When the user accesses another web-based system(called *App2*) by clicking *App*2's *URL*, *App2* redirects the *HTTP* request to Master Login page. Master Logon page retrieves the *signOnKey* from the browser's cookie, and redirects the *HTT*P request to *App2* with the *signOnKey*. Thus, *App2* also uses this *signOnKey* to retrieve user's information from the *LDAP* server. From the above description, it is easy to see that *WSSO* can be achieved by the proposed architecture.

The details of the proposed *WSSO* algorithm are shown in Fig. 2. Sixteen steps are given to demonstrate the proposed *WSSO* mechanism. All steps are described as follows:

Step 1: First, we suppose that an user opens his browser and connects to some web-based system *App1*.

Step 2: Since it is the first login for this user, *App1* does not contain any information about this user in this session. Then, user's *HTTP* request is redirected to *MLWP*(Master Login Web Page) with the parameter of the second page *URL* of system *App1*.

Step 3: *MLWP* prompts a login screen including the input form of username and password.

Step 4: User enters his appropriate username and password, then *MLWP* authenticates the user via the authentication function of the *LDAP* server. Note that all user informations are unified into the *LDAP* server.

Step 5: If the authentication is successful, a "success" message is returned from the *LDAP* server; otherwise; an "error" message is returned from the *LDAP* server.

Step 6: If an "error" message is received by *MLWP*, an illegal message is sent to the user via the *HTTP* response. If a "success" message is received by *MLWP*, a unique random encryption key *X* is inserted into the *signOnKey* attribute of this user's entry in the LDAP server. Note that the *signOnKey* attribute is defined in advance for each user's entry in the *LDAP* server.

Step 7: After inserting *X* into the *LDAP* server, *MLWP* also sets a cookie named *signOnKey* with the value *X* into the user's browser.

Step 8: *MLWP* redirects the user's *HTTP* request to the second page *URL* of *App1* with the parameter *X*.

Step 9: The second page of *App1* receives the redirected *HTTP* request and the attached parameter *X*. Then, *App1* calls a non-transparent *API* with value *X* to query the user information. Note that the non-transparent *API* is developed by the *LDAP*'s administrator for the security consideration.

Step 10: Next, the *LDAP* server verifies the parameter *X* with all existed values of attribute *SignOnKey* in the server. Note that the *LDAP* server can offer fast search for indexed data. If *X* is found, some user informations are returned to *App1,* for example, *cn*(common name), employee number and distinguished name and so on. In addition, *App1* also allows the user to browse and access the system. While an illegal *X* is given, the *LDAP* server will return an "error" message to *App1*. So that the user access can not be granted by *App1*.

Step 11: Suppose that the user accesses another web-based system *App2* by clicking the hyperlink or opening a new *URL*.

Step 12: Since it is the first login for this user in *App2*, *App2* does not contain any information about this user in this session. So, the user's *HTTP* request is redirected to *MLWP* with the parameter of the second page *URL* of *App2*.

Step 13: *MLWP* tries to obtain the value of cookie named *signOnKey* from the user. Note that this cookie is inserted to the user's browser by *MLWP* in ***Step* 7**. So *MLWP* can retrieve the value *X* of the cookie *signOnKey*. If this cookie can not be found, *MLWP* should go to ***Step* 3** due to the access of unauthenticated user. Note that the cookie *signOnKey* here is not the domain cookie and is viewed only by *MLWP*.

Step 14: The *MLWP* redirects the user's *HTTP* request to the second page *URL* of *App2* with the parameter *X*.

Step 15: The second page of *App2* receives the redirected *HTTP* request and the attached parameter *X*. Next, *App2* calls a non-transparent *API* with value *X* to acquire user information. This step is similar to the ***Step* 9**.

Step 16: Then, the *LDAP* server verifies the parameter *X* with all existed values of attribute *SignOnKey* in the server. If *X* is found, user information is returned to *App2*, for example, *cn*, employee number and distinguished name and so on. *App2* can grant the user to browse and access the system. While an illegal *X* is given, the *LDAP* server will return an "error" message to *App2*. Consequently, the access to *App2* should be denied. This step is also similar to the ***Step* 10**. ■

In the above *WSSO* procedure, ***Steps* 11~16** illustrate that how the user switches to another web-based system through the *WSSO* mechanism. *MLWP* tries to retrieve the value of cookie named *SignOnKey* for this user, then *MLWP* redirects the *HTTP* request with *signOnKey*'s value to the second page of *App2* for achieving *WSSO*. Note that when the user clicks logout icon or the browser is closed, the cookie *signOnKey* stored in browser and the value of *signOnKey* stored in the *LDAP* entry should be removed. This is for the security consideration. From the above discussion, it is clear that the proposed approach can work based on little overhead:

- One *signOnKey* attribute is added in each user entry.
- One cookie is stored in the user's browser.
- All web-based systems operate in coordination, that is, the first and second pages need to be re-coded a little.
- *MLWP* can be modularized easily to serve different *WSSO* groups.

IV. Implementation

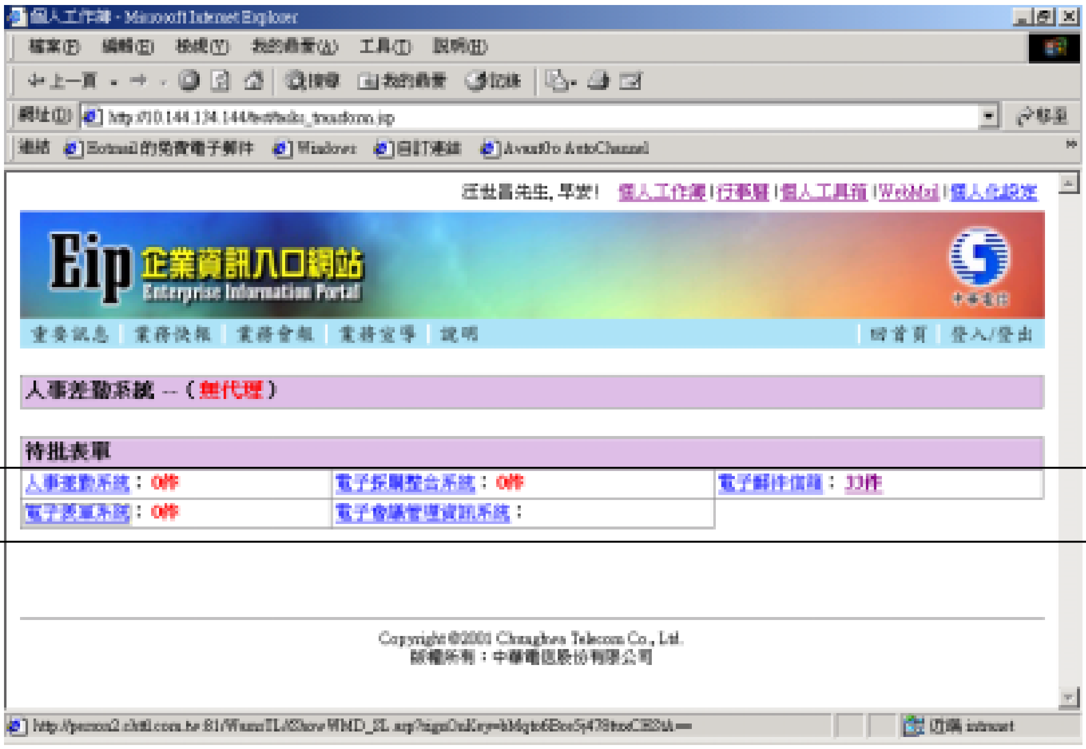

Fig 3: The *EIP* system developed in *CHT*.

At present, we have implemented the proposed *WSSO* in *CHT*'s *EIP* system. Some important *CHT*'s web-based systems(for example, Employee Management System, Web Mail System, e-Procurement Integrated system, e-Form System and e-Meeting System) have been integrated into *EIP* system to provide the *WSSO* function. An illustration for our implementation is shown in Fig. 3. According to the mechanism proposed in section III, five CHT's information systems are integrated successfully to support *WSSO* function. In this section, we describe the details about our implementation. First, an attribute named *signOnKey* must be added in the user's objectclass of the *LDAP* server. In addition, an entry named "*cn=wsso*" is created to manipulate the attribute *signOnKey*. The access right for attribute *signOnKey* is given in Table 1. Under the security consideration, only this special account "*cn=wsso*" has the search right of attribute *signOnKey*.

entries / access right	*self*	*cn=wsso*	*others*
compare	×	✓	×
read	×	✓	×
search	×	✓	×
write	✓	✓	×

Table 1: The assignment of access rights for attribute *signOnKey* in the *LDAP* server.

Second, the non-transparent search *API* used for the second page of each web-based system needs to be developed. In *CHT*'s implementation, we developed a method called "*getUserInfoBySignOnKey(signOnKey)*" to carry out the search. In this method, we use the *DN*(Distinguish Name) "*cn=wsso*" as the binding identification, and some user information are retrieved if the specified *signOnKey* can be found. The page *MLWP* is also needed to be coded according to the procedure described in Sec. III. Recall that the information about the inserted cookie and attribute *signOnKey* should be removed while the user makes a logout or the browser is closed.

Finally, a sample code is also needed to guide the co-operated systems to code the second page. Such a sample code is used to facilitate the *WSSO* integration. A *JSP*(Java Server Pages) example in *CHT*'s implementation is shown in Fig.4. It is easy to see that the co-operation code to support *WSSO* is very simple. The Java class *OpenLDAPAPI* is developed by the *CHT*'s *LDAP* group. Class *OpenLDAPAPI* contains many high-level generic *API*s to facilitate the *LDAP* access for the client's application. Furthermore, the *API* components and sample code for *ASP*(Active Server Pages) applications are also provided in *CHT*'s implementation to facilitate the integration of Windows platforms.

```
<%@page import="chttl.OpenLDAPAPI%>
String signOnKey =request.getParameter("signOnKey"); //get parameter
//.….
OpenLDAPAPI api = null;
api = new OpenLDAPAPI();
api.connect("10.144.21.5",389);
String[] cnUidDN = api.getUserInfoBySignOnKey(signOnKey);
If(cnUidDN==null){ out.print("illegal access");return;}
// process user's information
//…..
```

Fig. 4: An example for each co-operated system to code the second page.

V. Conclusions

EIP system is the application that enable company to unlock internally and externally stored information, and provide users a single gateway to personalized information needed to make informed business decisions. Recently, the *WSSO* issue is getting more critical in developing the *EIP* system. Some *WSSO* solutions were proposed in the literature or commercial products. The previous approaches mostly depend upon *Kerberos*, *PKI*, domain cookie or password-store, but they require client side infrastructure and new administrative steps. In this paper, we propose the *WSSO* solution used in the *CHT*'s *EIP* system. The proposed approach is only based on the unified *LDAP* server and the *HTTP* cookie. In our approach, user authentication is proceeded mainly by web page *MLWP*, and *signOnKey* is the bridge to communicate *MLWP*, information systems and user's browser to complete *WSSO*. Compared with the existing schemes, the proposed scheme is simple and consumes with little overhead. The practical implementation is also given to demonstrate that the proposed scheme can be easily applied to the integration of enterprise web-based systems. Because *MLWP* is a simple web page, multiple *MLWP*s can be established to implement individual *WSSO* among different application groups.

References

[1] C. Severance, "Could LDAP Be the Next Killer DAP?", *IEEE Computer*, pp. 88-89, Volume 30, Number 1, January 1997.

[2] W. Yeong, T. Howes, and S. Kille, "X.500 Lightweight Directory Access Protocol", *RFC 1487, IETF*, July 1993.

[3] V. Samar and S. Velloor, "Single Sign On Using Cookies For Web Application", http://www.coe.uncc.edu/~gahn/courses/ITSC8077/students/Sree.pdf

[4] M. Upadhyay and R. Marti, "Single Sign-on Using Kerberos in Java", http://java.sun.com/j2se/1.4/docs/guide/security/jass/single-signon.html

[5] K. Botzum, "Single Sign On – A Contrarian View", http://www7b.software.ibm.com/wsdd/library/techarticles/0108_botzum/botzum.html

[6] "The HTTP Cookie specification", *RFC #2965*, http://www.ietf.org/

[7] "Iplanet LDAP Directory", http://www.iplanet.com/products/iplanet_directory

[8] "IBM Policy Director", http://www.tivoli.com/products/index/secureway_policy_dir/index.html

[9] S. W. Sun, et al.,"LDAP APIs for C/C++/COM", *Coptyright #527, Telecommunication Laboratories Chunghwa Telecom Co., Ltd*, 2001

[10] S. W. Sun, et al.,"LDAP APIs for Java", *Coptyright #543, Telecommunication Laboratories Chunghwa Telecom Co., Ltd*, 2001

[11] "Oracle HTTP Server: Single Sign On Integration" ,http://technet.oracle.com/products/ias/daily/jan22.html

[12] "Domino and WebSphere Together", Second Edition,
http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg245955.html?Open

[13] "Single Signon Overview" ,
http://docs.iplanet.com/docs/manuals/portal/30/progref/signon.htm

[14] RFC 1777: "Lightweight Directory Access Protocol".

[15] RFC 2251: "Lightweight Directory Access Protocol(v3)".

[16] "Netscape Directory Server", http://developer.netscape.com/tech/directory/.

[17] C. Finkelstein, "The Emergence and Potential of Enterprise Information Portals (EIPs)",
http://www.tdan.com/i010fe02.htm.