國立交通大學

資訊科學與工程研究所

碩 士 論 文

基於 SAML 架構達到多層式跨 IDP 之

單一登入系統

Multi-Layered Cross-IDP SSO in SAML-based Architecture

研 究 生：張長蓉

指導教授：葉義雄　教授

中 華 民 國 九 十 六 年 六 月

基於 SAML 架構達到多層式跨 IDP 之單一登入系統

研究生：張長蓉　　　　指導教授：葉義雄 博士

國立交通大學資訊科學與工程研究所碩士班

摘要：

Web service 的出現是為了讓使用者能夠迅速及時使用網路資源，它使用 XML 來傳輸資訊以適應各種開發環境。隨著電子商務的興起，為了解決資訊安全的問題，OASIS 在 2002 年發展了一種以 XML 為基礎的語言 SAML，可安全產生和交換使用者認證和授權資訊。SAML 明確地定義了許多安全認證方式並以 XML 架構來加強之，這樣的優勢令許多網路廠商廣泛使用它來達到 Web SSO 的功能。

以目前 SAML 提供的 SSO（Single Sign-On）機制，是透過一個認證中心來管理使用者資訊，這個認證中心整合各種服務，使用者必須先到認證中心確認身分後，才能使用這些服務，只要在認證中心登入過一次，底下所屬的服務都無須再做登入動作即可使用。但若想使用不同 IDP（identity provider）底下的服務時，仍然要做多次登入動作認證身分。

為了提供使用者一個跨企業間的整合性服務，我們必須讓使用者在不同的認證中心底下仍可達到單一登入的功能，因此本論文以 SAML1.1 為基礎，提供了一種可跨 IDP 做聯合身分認證以達到 SSO 的系統。

關鍵字：Web 服務、安全宣示標記語言、單一登入系統 SSO、聯合身分認證

Multi-Layered Cross-IDP SSO Based on SAML

Student：Chang-Jung Chang　　　　Advisor：Dr. Yi-Shiung Yeh

Institute of Computer Science and Information Engineering

National Chiao Tung University

Abstract：

The development of Web Service enables users rapidly to access network resources in time. As a result of the electronic commerce starting, Web service uses xml to transmit the information to be able to adapt each kind of development environment. In order to solve the information secure problem, the Security Assertion Markup Language (SAML) which is an XML-based framework has been developed by the OASIS (the Organization for the Advancement of Structured Information Standards) to describe and exchange authorization and authentication information between on-line business partners in 2002. SAML explicitly defines several safe confirmations ways and the security of xml architecture will be enhanced with these methods. The superiority causes SAML widely to be used to achieve Web SSO by the on-line commercial systems.

At present SAML SSO mechanism is that there is an identity provider (IDP) which integrates several services managing users information. After logging in at IDP, the user can access these services. So long as a user has logged in at the authentication center, he does not need to authenticate again and then he directly can access these services at the same time. But a user has to login many times to provide valid credentials to use the services which are subordinate under different IDPs.

In order to provide the users a enterprise-crossed and integrated service, we must enable the users also to achieve SSO under many identity providers, the thesis designs a SSO architecture to achieve identity federation cross-IDP using SAML 1.1.

Key words：Web Services, SAML, SSO, Federated Identity, Identity federation

致　　謝

這篇論文能夠順利的完成，首先必須感謝我的指導教授，葉義雄教授，在研究所兩年中，不論是在學問上或是待人處事方面都給我莫大的鼓勵與指導，令我獲益良多。另外也很感謝鎮宇、定宇學長和靜紋學姊，在我遇到瓶頸時，給了我很多的寶貴意見以及協助；還有，也謝謝實驗室裡的各位學長姊、同學與學弟妹，雅婷、伯昕、小強、家明與佳君，大家平日的相互勉勵與學習，讓我成長許多；此外，我要特別感謝好友信州，在研究期間幫我一起跑程式，讓我因此節省許多寶貴的時間；擁有這群知心好友，讓我這兩年的生活更加的豐富，在此希望他們未來都能一路順風。

最後，要謝謝我的家人多年來的栽培與支持，以成就今日的我，家人的溫暖關懷讓我不輕易被挫折打倒，也讓我毫無牽掛的專心在學業之上，在學習的這條路上有了家人的支持和鼓勵，讓我感到相當的溫馨與幸福，在此真的非常感謝家人為我做的一切！

謹將我這篇論文獻給所有關心我與支持我的人，謝謝你們！

張長蓉
中華民國九十六年六月

Contents

Figure List

Table List

Chapter 1 Introduction

1.1 Background

The appearance of web service enables users to access network resources immediately and rapidly. As a result of the electronic commerce starting, web service uses xml to transmit the information to be able to adapt each kind of development environment. Although it is convenient and fast, the information secure anxiety is still existed.

Therefore the Security Assertion Markup Language (SAML) which is an XML-based framework has been developed by the Security Services Technical Committee (SSTC) of the standards organization OASIS (the Organization for the Advancement of Structured Information Standards) to describe and exchange authorization and authentication information between on-line business partners in 2002. SAML explicitly defines several safe confirmations ways depending on the different goals as follows: Authentication Assertion, Attribute Assertion, Decision Assertion, and Authorization Assertion. The security of xml architecture will be enhanced with these methods. The superiority causes SAML widely to be used to achieve Web SSO by the on-line commercial systems.

1.2 Motivation

At present SAML SSO mechanism is that there is an Identity Provider (IDP) which integrates several services managing users information. After logging in at IDP, the user can access these services. So long as a user has logged in at the authentication center, he does not need to authenticate again and then can access these services at the same time. It can be imagined that after logging in at Yahoo home page, the user could use its mailbox, auction, photo album services, and so on without providing credentials again. But a user has to login many times to provide valid credentials to use the services which are subordinate under different IDPs. The user still must remember a pile of accounts/passwords. Such inconvenient operation may reduce the willingness of using web services.

In order to support more commercial situations in real world, only considering the security aspect is not enough. We also hope to achieve extra mechanisms and provide the users a enterprise-crossed and integrated service.

1.3 Objective

In order to provide the users a enterprise-crossed and integrated service, we must enable the users also to achieve SSO under many Certificate Authorities. In order to provide a more conveniently sign-on mechanism, the thesis designs a SSO architecture to achieve identity federation cross-IDP.

Pointed out in the SAML V1.x specification, it is PKI-based, the information transmitted is protected by SSL. Therefore we premise that messages are exchanged safely with SSL.

Chapter 2 Related Work

2.1 Web Service

2.1.1 Concepts

Today, companies rely on thousands of different software applications each with their own role to play in running a business. To name just a few, database applications store information about customers and inventories, web applications allow customers to browse and purchase products online, and sales tracking applications help business identify trends and make decisions for the future. These different software applications run on a wide range of different platforms and operating systems, and they are implemented in different programming languages. As a result, it is very difficult for different applications to communicate with one another and share their resources in a coordinated way. To solve the problem of application-to-application communication, businesses need a standardized way for applications to communicate with one another over networks, no matter how those applications were originally implemented.

Web Services provide exactly this solution by providing a standardized method of communication between software applications. Specifically, web services are a stack of emerging standards that describe a service-oriented, component-based application architecture. With a standardized method of communication in place, different applications can be integrated together in ways not possible before. Different applications can be made to call on each other's resources easily and reliably, and the different resources that applications already provide can be linked together to provide new sorts of resources and functionality. Moreover, application integration becomes much more flexible because Web services provide a form of communication that is not tied to any particular platform or programming language. The interior implementation of one application can change without changing the communication channels between it and the other applications with which it is coordinated. In short, a Web service makes its resources available in such a way that any client application, regardless of its internal implementation, can operate and draw on the resources provided by the Web service.[1]

2.1.2 Components of Web Services

Web services provide a standard way to expose an application's resources to the outside world so that any user can draw on the resources of the application. Web services are built on standard technologies such as HTTP and XML. All Web service messages are exchanged using a standard XML messaging protocol known as SOAP (Simple Object Access Protocol), and Web service interfaces are described using documents in the WSDL (Web Services Description Language) standard. These standards are all completely agnostic of the platform on which the Web services were built.

Web services are able to expose their resources in this generally accessible way because they adhere to the following communication standards [1]:

1. A Web service publicly describes its own functionality through a WSDL file.
2. A Web service communicates with other applications via XML messages.
3. A Web service uses a standard network protocol such as HTTP.

WSDL

A WSDL file provides a description (written in Web Service Description Language) of how the Web service is operated and how other software applications can interface with the Web service. Think of a WSDL file as the instruction manual for a Web service explaining how a user can draw on the resources provided by the Web service. WSDLs are generally publicly accessible and provide enough detail so that potential clients can figure out how to operate the service solely from reading the WSDL file. If a Web service translates English sentences into French, the WSDL file will explain how the English sentences should be sent to the Web service, and how the French translation will be returned to the requesting client [1].

XML and SOAP

XML messages provide the common language by which different applications can talk to one another over a network. To operate a Web service a user sends an XML message containing a request for the Web service to perform some operation; in response the Web service sends back another XML message containing the results of the operation. Typically these XML messages are formatted according to SOAP syntax. SOAP, an acronym for Simple Object Access Protocol, specifies a standard format for applications to call each other's methods and pass data to one another. Note that other non-SOAP forms of XML messages are possible, depending on the specific requirements of the Web service. But, in any case, the sort of XML message and the specific syntax required can be found in the WSDL file, making the Web service generally available to any client application capable of sending and receiving the appropriate XML messages [1].

SOAP (Simple Object Access Protocol) is the method by which you can send messages across different modules. This is similar to how you communicate with the search engine that contains an index with the Web sites registered in the index associated with the keywords [2].

HTTP

To make it accessible to other applications across networks, such as the Internet and in-house intranets, Web services receive requests and send responses using widely used protocols such as HTTP (HyperText Transfer Protocol) and JMS (Java Message Service) [1].

2.2 Web Single Sign-On

2.2.1 Introduction

Single sign-on (SSO) is mechanism whereby a single action of user authentication and authorization can permit a user to access all computers and systems where he has access permission, without the need to enter multiple passwords. Single sign-on reduces human error, a major component of systems failure and is therefore highly desirable but difficult to implement [3].

Single sign on is generally a process that allows the user to access multiple applications requiring authentication by passing his credentials only once. The user first authenticates to some trusted authentication authority and then is granted access to all the applications trusting that authority. The SSO systems usually preserve the state of the user for some period of time, so the user may repeatedly access these applications without the need to authenticate each time.

One of the main advantages of SSO systems is the convenience for the user. Another major advantage is security. There is only one place of authentication, which receives user's credentials. The applications only receive information about whether they may let the user in or not. Also, the user authenticates only once, so there is minimum transfer of sensitive information over the network, not to mention that SSO systems usually force the users to use secure communication channels.

Web single sign on provides SSO infrastructure for web applications. On community networks, there is often a number of web applications and services designed to aid community members and thus requiring authentication. In these cases it is convenient and secure to use a centralized SSO infrastructure bound to the central authentication authority. The most common example of such community networks are university campus networks and some of the most common web SSO systems were developed there [4].

2.2.2 Overview of Web SSO Functionality

There is always a central authority, which handles user authentication. It may support various backend authentication mechanisms like Kerberos, LDAP, relational database, etc. The central authentication server may provide also a user interface needed to retrieve user's credentials - usually a login form [4].

The applications in the SSO infrastructure are protected by special application layers called filters or clients. These filters, usually implemented as modules for the web server running the application, check if the user is authenticated before letting him to access the protected application. To perform these checks they need to communicate with the authentication server either directly or through redirects of the user's web browser.

There are two common scenarios for a SSO session: [4]

- login first - the user first performs login to the SSO infrastructure and then chooses a service to access
- application first - the user first tries to access a service, but because he has not been authenticated yet, the service redirects him to the login service and after a successful logon he is redirected back to the service

The use case [6], shown in Figure 2.2.1, described here demonstrates what the concept of web SSO is. In this use case, a user has a login session (that is, a *security context)* on a web site (*Airplane.com*) which maintains local identities for users and is accessing resources on that site. At some point he is directed over to a partner's web site (*CarRent.com*). We assume that a trust relationship has been previously established between *Airplane.com* and *CarRent.com* based on a business agreement between them. The site (*Airplane.com*) asserts to the service provider site (*CarRent.com*) that the user is known, has authenticated to it. Since *CarRent.com* trusts *Airplane.com*, it trusts that the user is valid and properly authenticated and thus creates a local session for the user. The user is not required to re-authenticate when directed over to the *CarRent.com* site.

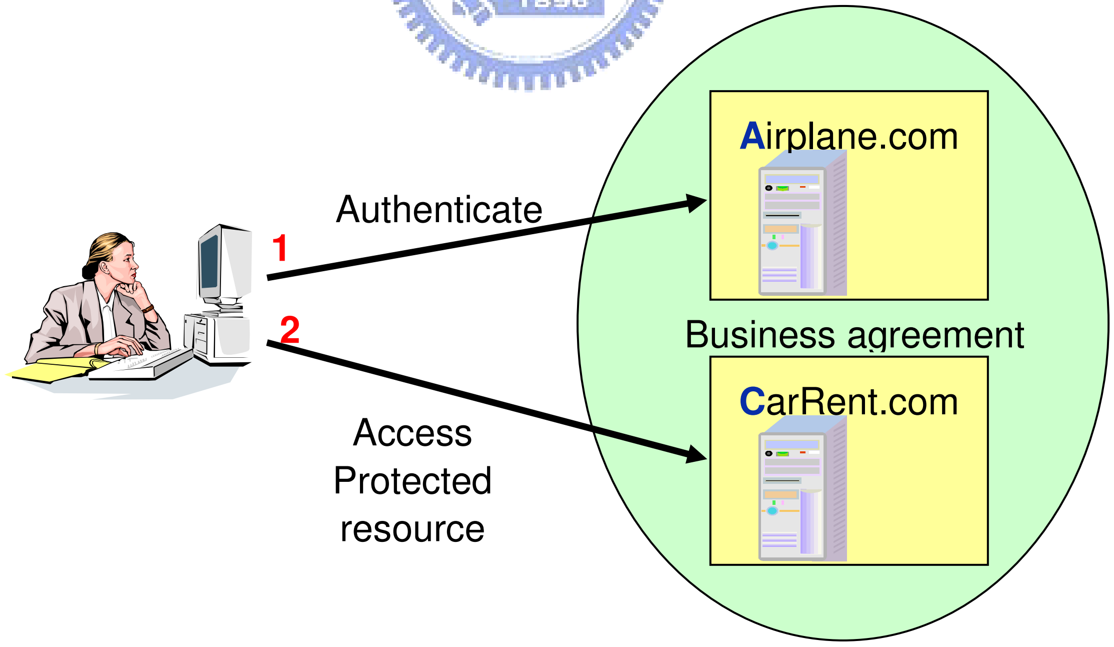


Figure 2.2.1 General Single Sign-On Use Case

Chapter 3 Technical Overview of SAML 1.1

3.1 Introduction

The OASIS Security Assertion Markup Language (SAML) standard defines an XML-based framework for describing and exchanging security information between on-line business partners. This security information is expressed in the form of portable SAML assertions that applications working across security domain boundaries can trust. The OASIS SAML standard defines precise syntax and rules for requesting, creating, communicating, and using these SAML assertions.

SAML defines a XML-based solution to the problem of Web Single Sign-On (SSO). Web SSO allows users to gain access to website resources in multiple domains without having to re-authenticate after initially logging in to the first domain. To achieve SSO, the domains need to form a trust relationship before they can share an understanding of the user`s identity that allows the necessary access.

3.2 SAML Architecture

This section describes the core SAML concepts and components briefly.

3.2.1 SAML Participants and Scenarios

SAML which exchanges take place between system entities referred to as a SAML *asserting party* and a SAML *relying party* is different from other security systems due to its approach of expressing assertions about a subject that other applications within a network can trust [5].

- **Asserting party**

  The system that makes SAML assertions asserts information about a subject. It is also called a *SAML authority* which role is defined as *identity provider (IDP).*

- **Relying party**

  The system that relies on information supplied to it by the asserting party uses assertions it has received. It is up to the relaying party as to whether it trusts the assertions provided to it. SAML defines the role called *service provider (SP).*

- **Subject**

  At the heart of SAML assertions, a subject, could be a human but also could be some other kind of entity, such as a company or a computer. The terms subject and user tend to be used interchangeably in this thesis.

When a SAML asserting or relying party makes a direct request to another SAML entity, the party making the request is called a *SAML requester*, and the other party is referred to as a *SAML responder*. A relying party`s willingness to rely on information from an assertion party depends on the existence of a trust relationship with the asserting party.

A typical assertion about a subject who has been authenticated and has given associated attributes from an identity provider might convey information such as "This user is ***Bob Li***, he has an email address of ***bob.li@edu.com***, and he was authenticated into this system using a ***password*** mechanism," A service provider could choose to use this information, depending on its access policies, to grant ***Bob Li*** web SSO access to local resources. Figure 3.2.1 illustrates the concept of Web SSO, and how to achieve this purpose will be demonstrated later in the thesis.

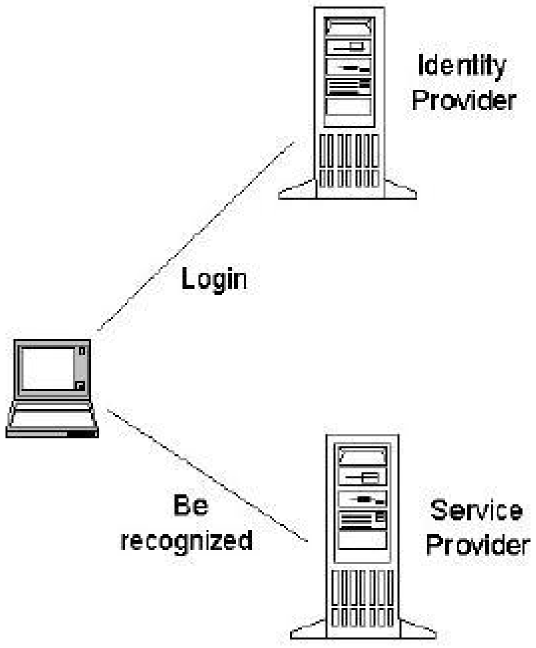


Figure 3.2.1 SAML Participants

In this case, a user has a login session on a web site, *Company.com*, and is accessing resources on that site. At sometime, he is directed over to a partner`s web site, *Travel.com*. Assuming a trust relationship has been previous established between *Company.com* and

Travel.com based on a business agreement between them. The identity provider site, *Company.com*, asserts to the service provider site, *Travel.com*, that the user has authenticated to it and has certain identity attributes (e.g. has a "Gold" status). Since *Travel.com* trusts *Company.com*, it trusts that the user is valid and properly authenticated and thus creates a local session for the user. The user is not required to re-authenticate when directed over to the *Travel.com* site. Once logged in, the IdP can produce an assertion that can be used by the SP to validate the user`s access rights to the protected resource [5].

3.2.2 SAML Components

SAML consists of building-block components that allow many use cases to be supported, when put the components together. It has the following key concepts: [6]

- **Assertion:** A SAML assertion is a package of information that carries statements about a subject. Assertions are created by a SAML authority. SAML defines three kinds of statements that can be carried within an assertion.
  - **Authentication statements:** They describe the means used to authenticate the user and the specific time at which the authentication took place.
  - **Attribute statements:** These contain specific details or identifying attributes about the subject, for example, the user holds "Gold" card status.
  - **Authorization decision statements:** These define what the subject is entitled to do, for example, whether a user is permitted to buy a specific production.

- **Protocol:** SAML defines a number of generalized request/response protocols for obtaining assertions. SAML protocol messages are used to make the SAML requests which can either ask for a specific known assertion or make authentication, attribute, and authorization decision queries and return appropriate responses.

- **Bindings:** A binding details exactly how the various SAML protocol messages map onto underlying transport and messaging protocols. For example, SAML provides a binding of how request/response protocols are carried within SOAP exchanges over HTTP.

- **Profiles:** SAML profiles typically define how the SAML assertions, protocols, and bindings are combined and constrained to solve particular business use cases in an interoperable fashion, for example the Web Browser SSO profile.

Figure 3.2.2 illustrates the relationship between these basic SAML concepts. [6]

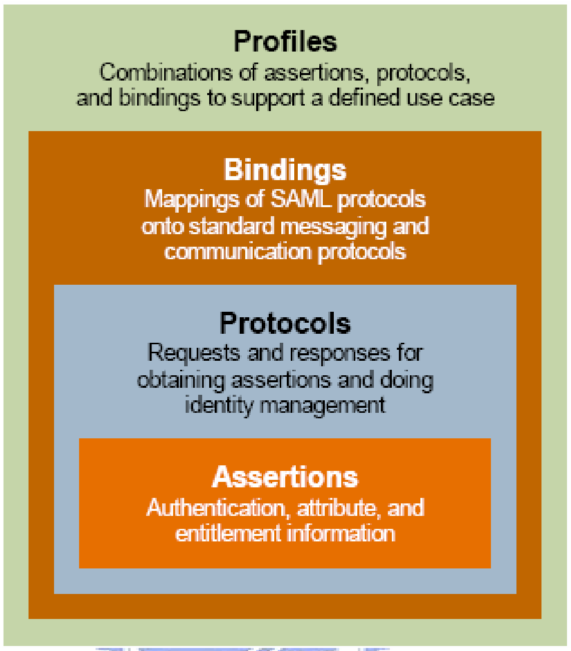

Profiles
Combinations of assertions, protocols, and bindings to support a defined use case
Bindings
Mappings of SAML protocols onto standard messaging and communication protocols
Protocols
Requests and responses for obtaining assertions and doing identity management
Assertions
Authentication, attribute, and entitlement information

Figure 3.2.2 Basic SAML Component

3.2.3 Security in SAML

SAML defines many security mechanisms to detect and protect against the "man-in-the-middle" attacks. The primary mechanism is for the relying party and asserting party to have a pre-existing trust relationship which typically relies on a Public Key Infrastructure (PKI). Using PKI is recommended by SAML, however, what is recommended is provided below:

- HTTP over SSL 3.0 or TLS 1.0 is recommended to ensure ***message integrity*** and ***message confidentiality***.
- When a relying party requests an assertion from an asserting party then **bi-lateral authentication** is required and the use of SSL 3.0 or TLS 1.0 using mutual authentication or authentication via digital signatures is recommended.
- When a response message containing an assertion is delivered to a relying party via a

user`s web browser, it is mandated that the response message be ***digitally signed*** using the XML signature standard to ensure message integrity [5][7].

3.3 High-Level SAML Use Cases

Early in its business requirements analysis, the SSTC (Security Services Technical Committee) defined many use cases for SAML. This section will describe in detail the Web SSO and identity federation use cases.

3.3.1 Web Single Sign-On Use Case

To date, only the Web SSO use case has been profile in SAML V1.1. The following Figure shows the processing and message flow in the Source-Site-First scenario, that is the IDP-initiated Web SSO scenario. It indicated that the user had first authenticated at the IDP before accessing a protected resource at the SP. When users whom had not been authenticated subsequently attempts to access a protected resource at the SP first, the SP will send the user to the IDP with an authentication request in order to have the user log in.

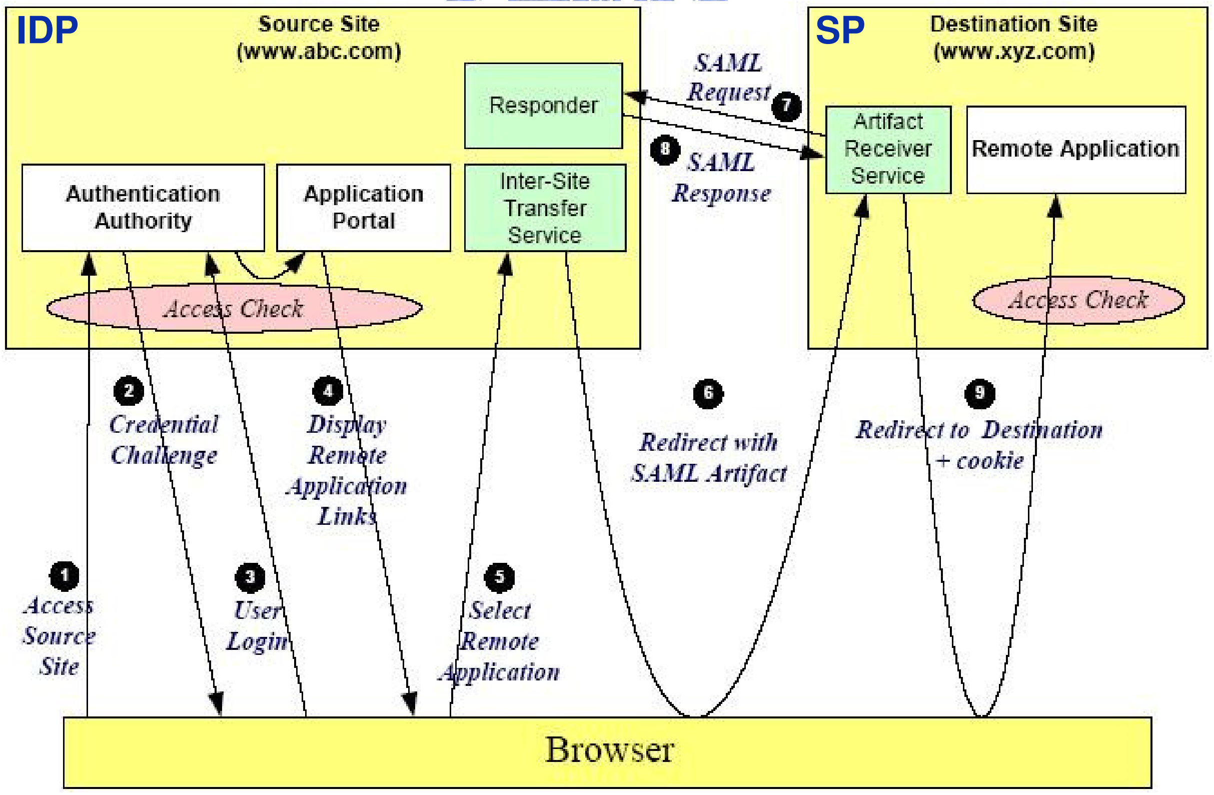


Figure 3.3.1 Detailed Processing for the SSO

The processing is as follows: [5]

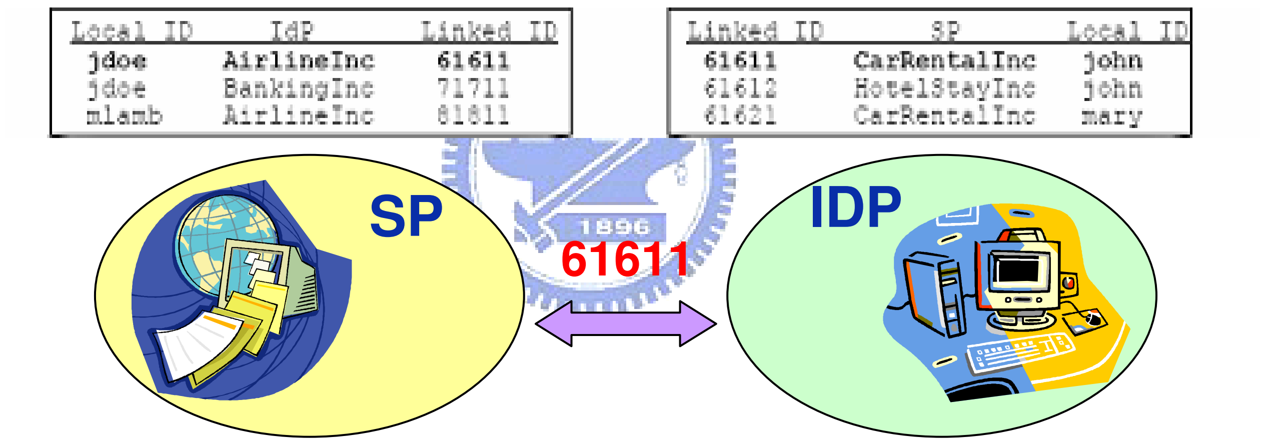
1. The user accesses the IDP site (www.abc.com).
2. The IDP performs an access check and determines that the user does not have a current session. As a result, the user is challenged to authenticate.
3. The user logs in and supplies back credentials, for instance username and password.
4. If the authentication is successful, then a session is created for the user who can access resources.
5. At some point, the user wants to access resources on a SP site www.xyz.com. This causes a HTTP request which contains the URL of the resource on the SP site (TARGET URL) to be send to the IDP site
6. The IDP generates an assertion which contains the source ID of the www.abc.com SAML responder for the user and sends back an HTTP redirection response to the browser, with the HTTP location header containing the TARGET URL.
7. The SP site receives the HTTP message and extracts the source ID. A mapping between source IDs and remote Responders will already have been established. Therefore the SP will know that it has to contact the www.abc.com SAML responder, then it will send a SAML request to the www.abc.com SAML responder.
8. The www.abc.com SAML responder supplies back a SAML response message containing the assertion about the user. If a valid assertion is received back, then a session on www.xyz.com is established for the user at this point.
9. The SP site sends a redirection message containing a cookie back to the browser. The message indicates that the user has the correct authorization to access the SP site.

3.3.2 Identity Federation Concepts and Use case

Multi-domain web single sign-on is the most important use case for which SAML is used. This section describes mechanisms supported by SAML for establishing and managing federated identities.

SAML V2.0 introduced two features to enhance its federated identity capabilities. First, new constructs and messages were added to support the dynamic establishment and management of federated name identifiers. Second, two new types of name identifiers were introduced with privacy-preserving characteristics. SAML uses the new features to dynamically establish a federated identity for a user during a web SSO exchange. Most identity management systems maintain *local identities* which might be represented by the user's local login account for users. These local identities must be linked to the federated identity that will be used to represent the user when the provider interacts with a partner. The process of associating a federated identifier with the local identity at a partner where the federated identity will be used is often called *account linking*.

Figure 3.3.2 illustrates dynamic identity federation using persistent pseudonym identifiers. The user ***jdoe*** on *CarRentalInc.com* wishes to federate this account with his ***john*** account on the IDP, *AirlineInc.com* [6].



Local ID	IdP	Linked ID
jdoe	AirlineInc	61611
jdoe	BankingInc	71711
mlamb	AirlineInc	81811

Linked ID	SP	Local ID
61611	CarRentalInc	john
61612	HotelStayInc	john
61621	CarRentalInc	mary

Figure 3.3.2 Identity Federation with Persistent Pseudonym

In summary, the processing is as follows:

1. The user attempts to access a resource on *CarRentalInc.com*. The user does not have any current logon session (i.e. security context) on this site, and is unknown to it.
2. The service provider sends the user to the Single Sign-On Service at the identity provider (*AirlineInc.com*) and requests the IDP to provide an assertion using a persistent name identifier for the user.

3. The user will be challenged to provide valid credentials.
4. The user logs in as ***john*** user account and the IDP creates a local security context for the user.
5. The Single Sign-On Service looks up user ***john*** in its identity store and creates a persistent name identifier (61611) to be used for the session at the service provider. It then builds a SAML assertion where the subject uses a transient name identifier format rather then the name ***john***.
6. The browser receives the assertion and issues a request to send the form to the service provider.
7. The SP validates the digital signature on the SAML Response and validates the SAML assertion. The supplied name identifier which maps to a local account is then used to determine whether a previous federation has been established. If a previous federation has been established then go to step 9. If no federation exists for the persistent identifier in the assertion, then the SP needs to determine the local identity to which it should be assigned. The user will be challenged to provide local credentials at the SP.
   Optionally the user might first be asked whether he would like to federate the two accounts.
8. The user provides valid credentials and identifies his account at the SP as ***jdoe***. The persistent name identifier is then stored and registered with the ***jdoe*** account along with the name of the identity provider that created the name identifier.
9. A local logon session is created for user ***jdoe*** and an access check is then made to establish whether the user ***jdoe*** has the correct authorization to access resources at the SP site.
10. If the access check passes, the desired resource is returned to the browser.

Chapter 4 Our System Architecture

In this thesis, we provide a system architecture to achieve the cross-IDP identity federation. In this chapter, we discuss the mechanism concepts before the design of our system, and at last we will demonstrate the implementation result.

4.1 System Overview

4.1.1 Scenario

The relationship of level exists in the commercial system. Lots of homogeneous organizations are subordinate to a kind of commercial system, an organization manages a number of same kind companies, each company is composed of many departments, and the department provides several services. It is graded layer upon layer in real commercial situation, as Figure 4.1.1 demonstrates. Assume a user is registered on some companies which are cooperative enterprises. As long as he has logged in at one of the web sites, he does not need to authenticate again and then can access these services which are subordinate under different webs.

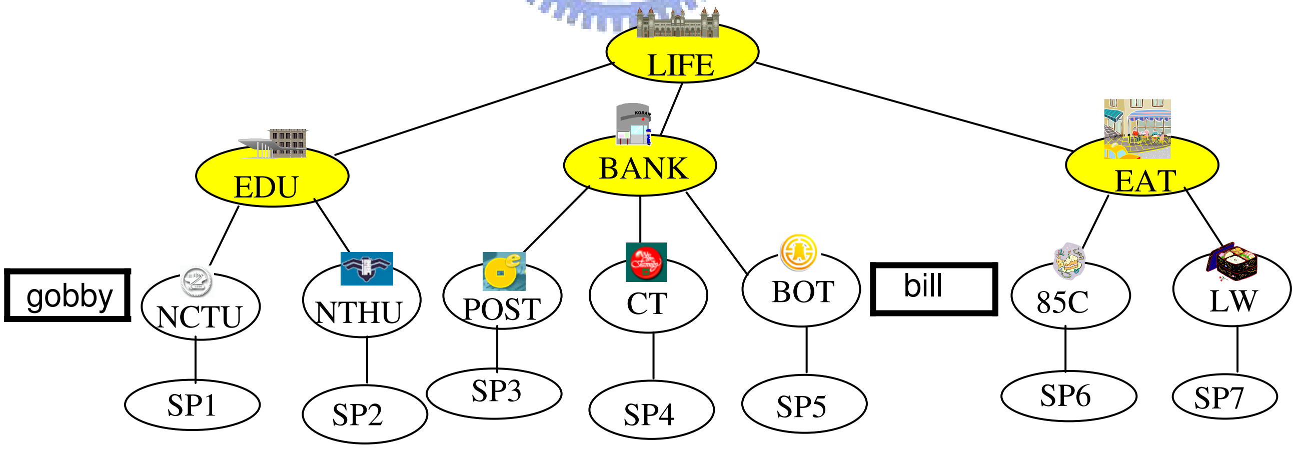


Figure 4.1.1 Layered Concepts in Actual Commercial System

It may be suitable for a certain commercial scenario as below. There are lots of companies or alliance shops being opened or closed every day. An alliance shop may join in or leave from the business system willfully. When users visit another IDP site, they must know whether it continues to operate or not. The participants and users shall know the cooperation relationship between the mutable commercial system immediately. In accordance with such scenario, the topology of business system must be maintained and managed dynamically.

4.1.2 System Components

In contrast with the realistic commercial system, Figure 4.1.2 illustrates the components which are provided in our system architecture. There are three main rules, Service Provider (SP), Identity Provider (IDP), Manager, in the architecture. All IDPs are portal sites and provides many kinds of services. All levels above IDPs are Managers, which exchange users` identities. First we outline these three components and introduce them in detail in the next section.

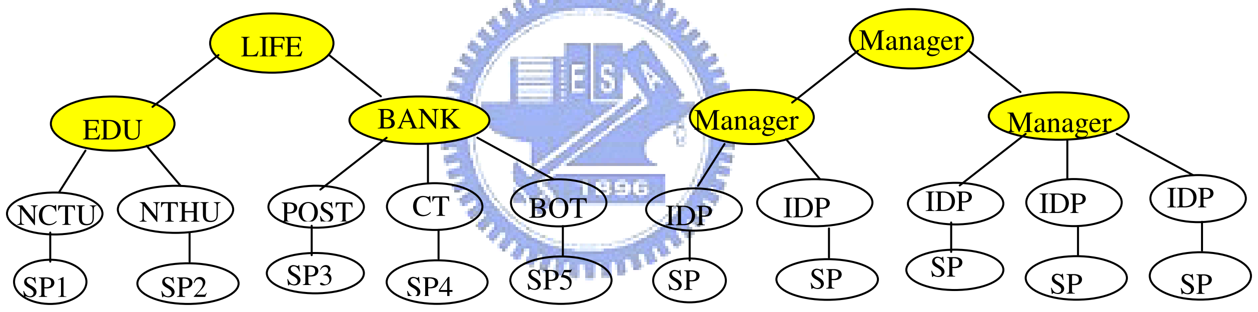


Figure 4.1.2 System Components

- **Service Provider (SP)**

  The system provides resources and services

- **Identity Provider (IDP)**

  The system integrates several services and maintains users information. Users must log in at IDPs before accessing SPs.

- **Manager**

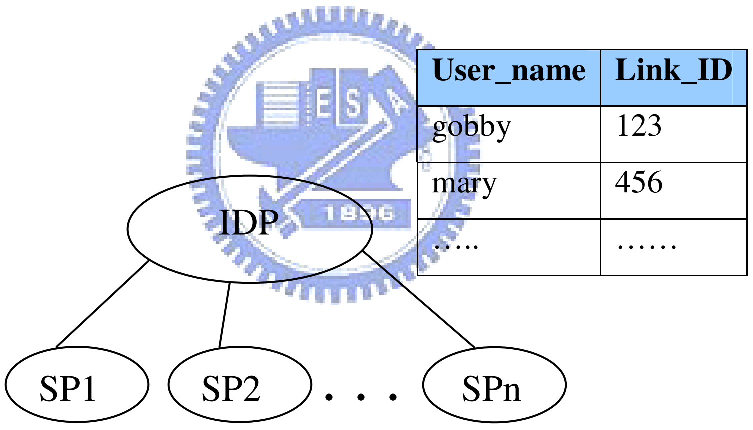
  The system exchanges users` identities and records the corresponding identities between different IDPs for a user.

4.2 Concepts and Mechanisms

4.2.1 User Link_ID

Essentially, the concept of user Link_ID is similar to the pseudonym. The IDPs maintain local identities which might be represented by the user local login account for users. These local identities must be linked to the federated identity, that is Link_ID, which is created by the IDP that will be used to represent the user when the provider interacts outward. Link_ID is unique for every user and generated randomly. It provides privacy-preserving characteristic for users. Figure 4.2.1 illustrates the database maintained by the IDP using Link_ID. For example, Link_ID (123) is represented the local user account ***gobby*** externally.

The process of associating a federated identifier with the local identity at a partner where the federated identity will be used is called ***account linking*** afterward.



User_name	Link_ID
gobby	123
mary	456
…..	……

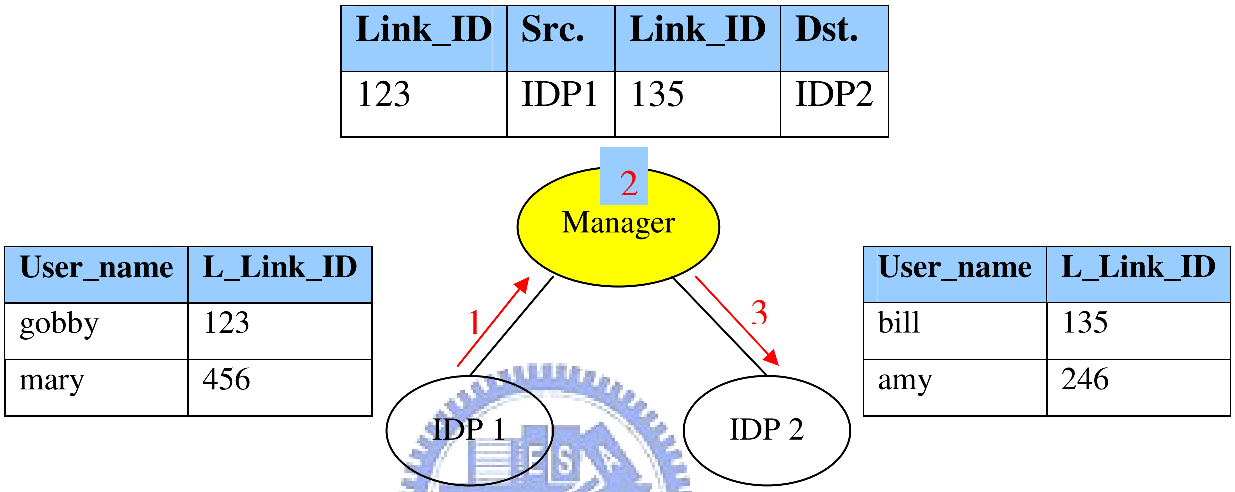
Figure 4.2.1 IDP Associates a Federated Identifier with Local Identity

4.2.2 Manager

In order to achieve identity federation, we define a third-party architecture, manager, to exchange users` identities. The manager records the corresponding identities at different IDPs for a user, however, the information recorded is the Link_ID rather than the local account. It provides privacy-preserving characteristic for a user during a web SSO exchange .The process of associating a Link_ID with another Link_ID at a partner is called ***account linking*** afterward.

Besides account linking, the manager can be used to classify the IDPs according to their properties. For instance, the IDPs which are the educational categories petition the "*EDU*" manager for account linking; the financial categories petition the "*BANK*" manager, etc. Thus, commercial system can be integrated more efficiently.

Figure 4.2.2 shows the operation of a manager, the example assumes a user is registered on two provider sites, but the local accounts have different account identifiers, Link_ID. Assume that the user has logged in at *IDP1* using his **gobby** account.



Link_ID	Src.	Link_ID	Dst.
123	IDP1	135	IDP2

User_name	L_Link_ID
gobby	123
mary	456

User_name	L_Link_ID
bill	135
amy	246

Figure 4.2.2 Account Linking Table in the Manager

In summary, the processing is as follows:

1. The user access resources at the IDP1 and then visits *IDP2*. The IDP1issues a request to manager: "(123) at *IDP1* wants to visit *IDP2*".
2. The manager receives the request and then looks up its account linking table to determine whether a previous federation has been established. If a previous federation has been established then go to step3, otherwise the IDP2 needs to determine the local identity to which it should be assigned. The user will be challenged to provide local credentials at the IDP2. Assume the federation exists, thus the manager finds (123) at IDP1 is (135) at *IDP2*.
3. Then the manager issue an assertion "(135) is authenticated" to the IDP2, and then the user bill associated with the Link_ID (135) has the correct authorization.

4.2.3 Sorting

If there are many users joining the federation organization, the account linking table at a manager will be very large. We must provide an efficient way to determine whether a previous federation has been established for a certain user. The index numbers of entries involved the same IDP are collected and stored orderly in another table. As Table 4.2.1 shows, the index numbers of entries involved the IDP1 are (1, 3, 5), and the index numbers of entries involved the IDP2 are (1, 4). We can get (1) by intersectint (1,3,5) and (1.4), so only the first entry must be traced when a certain user at IDP1 links to IDP2 or reversed.

Src./Dst.	Index
IDP1	1, 3, 5
IDP2	1, 4
IDP3	2, 3, 4
IDP4	2, 5

Index	Link_ID	Src.	Link_ID	Dst.
1	ABC_IDP1	IDP1	ABC_IDP2	IDP2
2	ABC_IDP4	IDP4	ABC_IDP1	IDP3
3	ABC_IDP3	IDP3	ABC_IDP1	IDP1
4	ABC_IDP3	IDP3	ABC_IDP2	IDP2
5	ABC_IDP2	IDP4	ABC_IDP4	IDP1

Table 4.2.1 Sorting Mechanism

4.2.4 Maintain the net topology

The net topology always changes as a result of the affiliation of sunrise industry or member IDPs. For the purpose of expansibility, we define that each manager must record the nodes of its upper layer and lower layers. The parent node and children nodes of a manager will be marked 0 and 1. Assume the topology is as Figure 4.2.3 shows, the manager EAT marks 1 to represent its children nodes as Table 4.2.2 shows.

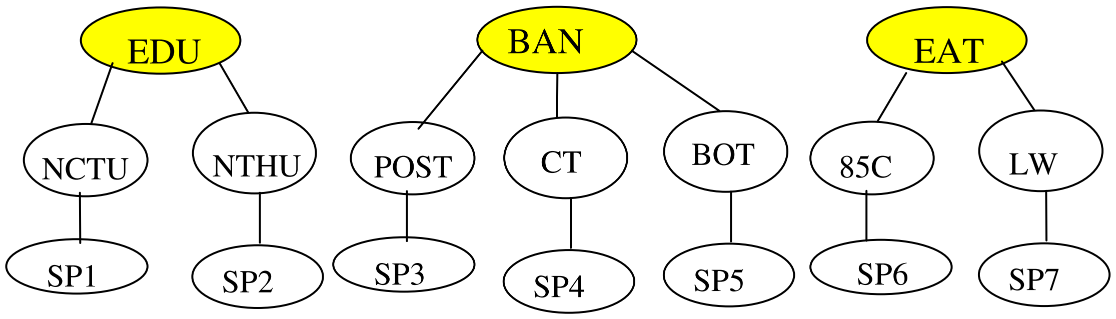


Figure 4.2.3 Topology of Initial Commercial System

EAT	
1	85C
1	LW

Table 4.2.2 Local Relationship of EAT manager

As Figure 4.2.4 shows, if someday an upper layer manager, LIFE, is increased EAT must mark 0 to represent it. LIFE also needs to record its children nodes, as disputed in Figure. Tables must be updated every time when the net topology changed.

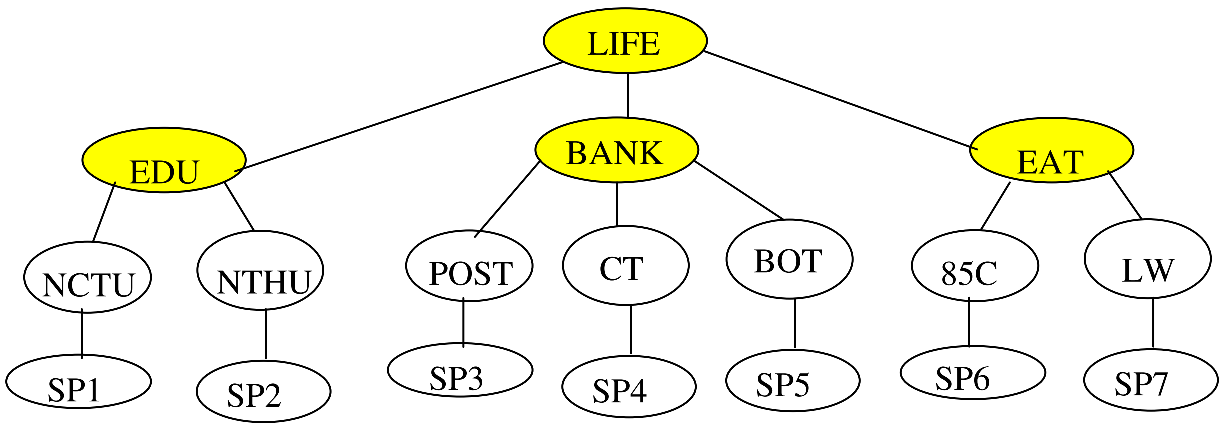


Figure 4.2.4 Topology of Commercial System after Adding LIFE Manager

EAT	
0	LIFE
1	85C
1	LW

Table 4.2.3 Local Relationship of EAT Manager after Modification

4.2.5 Global Path Finding

The architecture defines that a user at a IDP wants to visit another IDP by following the tree path rather than peer to peer. Deliberate that the source IDP and destination IDP might not be below the same manager, so how to know the position of the destination IDP must be solved.

The last section describes that each manager must record the nodes of its upper layer and lower layers. We can take advantage of it to find the global path of a certain IDP over whole topology.

Briefly, in a path, a manager stores the Link_ID of the left neighbor along with the Link_ID of the right neighbor for every user. We can see the following case to understand the operation of global path finding.

The global path of an IDP is represent to a sequence string which can indicate the position and relationship of the IDP. For the purpose of expansibility, the global path string is coded dynamically. In this case, the global path of IDP E should be coded "A_C_E", it indicates that E`s manager is C and C`s manager is A, and so on.

Figure 4.2.5 illustrates the processing of petition which is as follows:

1. E sends a request to its manager to get its global path.
2. C receives the request and then appends C. The path code becomes " C_E ". C looks up its table to know that it is not on the top layer, then C keeps on sending the request to upper layer.
3. A receives the request and then appends A. The path code becomes " A_C_E ". A refers to its table to know there is no more upper layer, then sends back " A_C_E " to E.

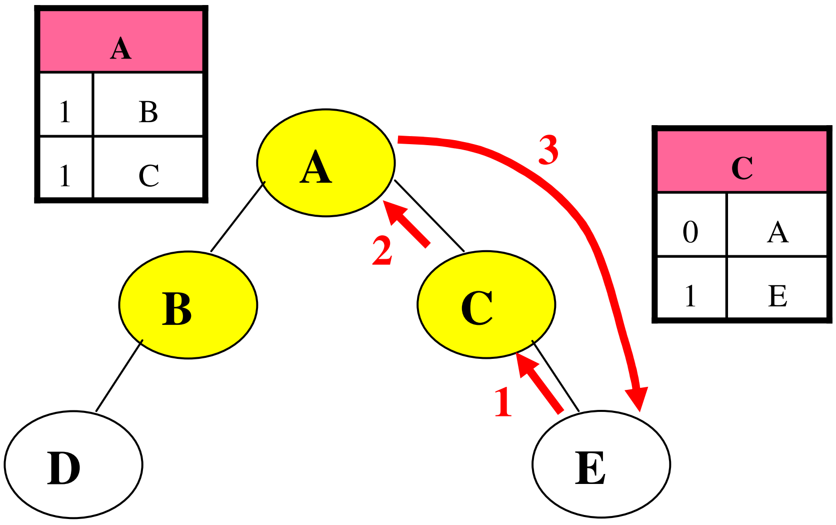


Figure 4.2.5 A Simple Global Path Finding Example

An IDP site can ask its global path upward gradually as above and stores the path code in local site. Each IDP asks upward periodically and updates its path code when the topology changed.

4.2.6 Core algorithm of manager

When getting the global path of the destination IDP, the source IDP will send the string and the request of identity federation to its manager. After receiving, the manager would compare the string with its name to determine whether its name appears or not. If it appears, the fact expresses that the target IDP is under its subtree surely, and the manager should send the request downward. Otherwise, the manager sends the request upward.

Each manager which receives the request of identity federation will look up its account linking table to determine whether a previous federation has been established for the user. If a previous federation has been established then continue to transmit forward according to the Link_ID, otherwise the manager needs to generate a Link_ID and record it to establish a new account linking entry.

However, two situations would be considered in transmission process downward:

1. The manager sends to its lower manager, the operation is the same as describing above.
2. The manager sends to the target IDP, the operation is similar to as describing above. If a previous federation has not been established, the manager needs to generate a Link_ID

and record it to establish a new account linking entry as usual. But the Link_ID generated by the manager and the Link_ID represented another user at the target IDP might be exactly the same. If the manager sends the Link_ID to the IDP, the wrong user account with the Link_ID will be logged in automatically which results in inaccurate identity federation. To solve such problem, we define that the manager generates a temporary Link_ID which might be expressed by "managerName_Tmp_Link_ID" form when sending to the destination IDP. After receiving the temporary Link_ID, the IDP sends back the real Link_ID to accomplish correct identity federation.

The algorithm of managers in details as Figure 4.2.6 shows.

```
if   path code match its name
    if   mapping record exists
        find the link_id from database
        send to the lower node
    else
        if   self is the first layer manager
            generate a temporary link_id, create a new mapping record
            send to the lower idp, and request the correct link_id
        else
            generate a link_id, create a new mapping record
            send to the lower node
else
    if   mapping record exists
        find the link_id from database
        send to the upper manager
    else
        generate a link_id, create a new mapping record
        send to the upper manager
```

Figure 4.2.6 Core Algorithm of Manager

4.3 Use Cases

Up to now, the concepts of mechanisms used in our system architecture are explained. The detailed processing will be described in this section. It deserves to be mentioned that the architecture is PKI-based and the information transmitted is protected by SSL.

4.3.1 Detailed Processing

Above all, let`s consider the simplest situation. That is, there are just two IDPs in the net topology. If the IDPs become partners, the users at two IDPs are federated identity by a manager. The example assumes a user is registered on two IDPs, but the local accounts both have different Link_IDs. As Figure 4.3.1 shows, at the IDP1, the user is registered as gobby, on the IDP2 his account is bill. User ID gobby at IDP1 recommend user ID bill at IDP2 to use the federated identity service for the user who has not previously federated his identity between these sites. Initially, there is not any account linking record about the user in the manager.

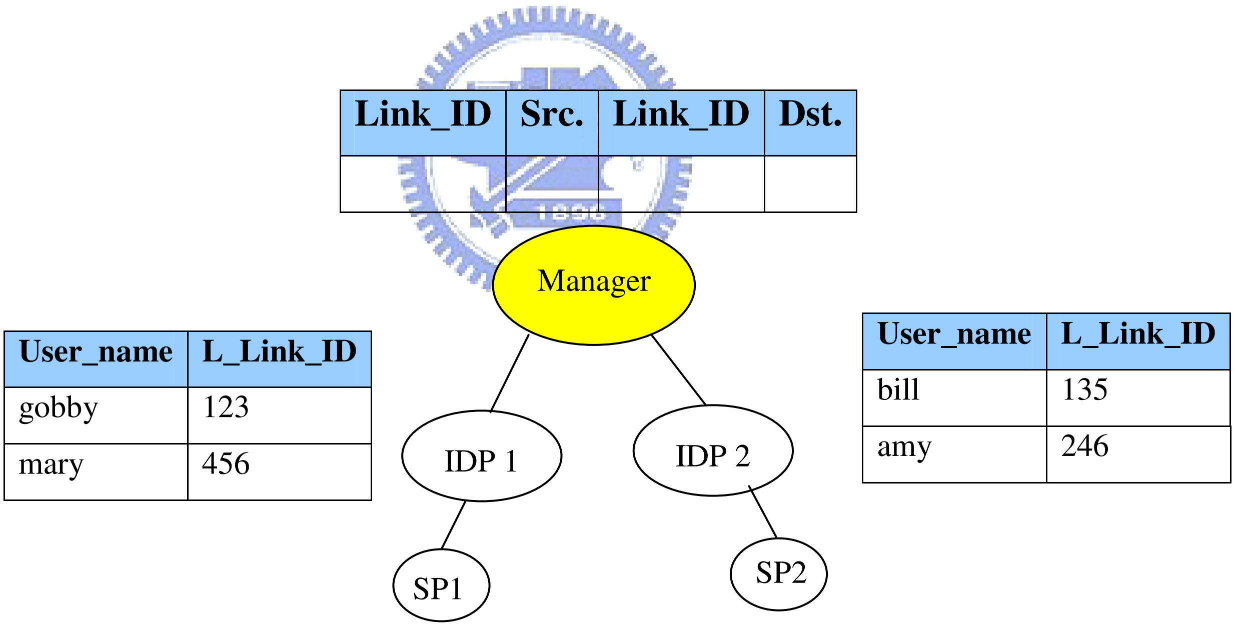


Figure 4.3.1 Initial State before Federating

Figure 4.3.2 illustrates the processing of petition which is as follows:

1. The user attempts to access a resource at the SP1 site. The user does not have any current logon session at this site, and is unknown to it.

2. The SP1 sends the user to the IDP1 and request the IDP1 to provide an assertion for the user. Thus the user logs in the IDP1 and gets the services on SP1 using SAML authentication.
3. The user petitions for federated identity to the IDP1.
4. The IDP1 issue a request for the user to manager "(123) at *IDP1* wants to visit *IDP2* ".
5. The manager receives the request and looks up that there is no account linking information between the IDP1 and the IDP2 for the user. Thus it creates an new entry and fills in the blanks with "(123)、IDP1、IDP2" as Table 4.3.1 demonstrates. Then a redirection is issued to the target web, IDP2, and which is requested the user`s Link_ID at the IDP2.
6. The user provides valid credentials at the IDP2 as bill. The IDP2 sends back the Link_ID, (135), of the bill account to the manager. The persistent name identifier is then stored along with the entry as Table 4.3.2 shows.
7. The IDP2 issues the assertion, and then the user bill has the correct authorization to access the SP2.

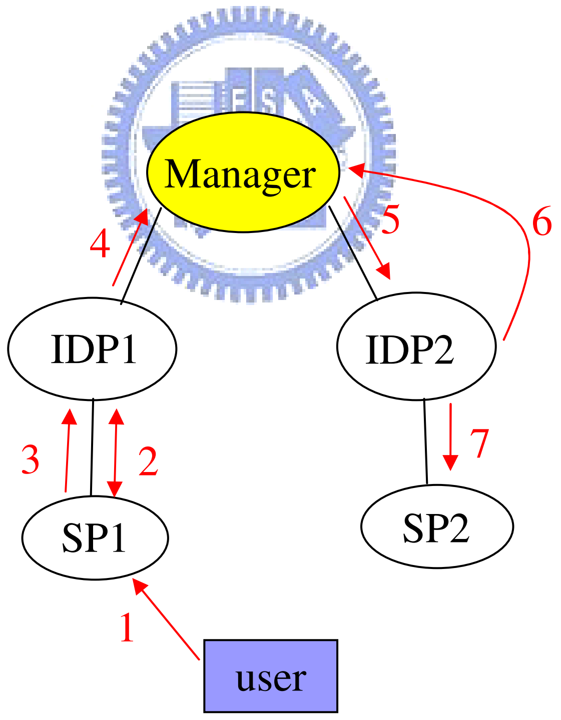


Figure 4.3.2 Processing Sequence of federation petition

Link_ID	Src.	Link_ID	Dst.
123	IDP1		IDP2

Table 4.3.1 The Mapping Table in Manager in Step 5

Link_ID	Src.	Link_ID	Dst.
123	IDP1	135	IDP2

Table 4.3.2 The Mapping Table in Manager in Step 6

The identity linking record which has been established exists perpetually as long as there is a business agreement between both IDPs. After petition the user at the IDP1 wants to access a resource at the IDP2, the manager looks up the mapping record about the user in its identity store. Then the manager issue an assertion "(135) is authenticated" to the IDP2, and then the user bill associated with the Link_ID (135) has the correct authorization.

We will describe a more complicated situation. The questions how to achieve identity federation involving multi-layered managers must be considered. Assume the net topology as Figure 4.3.3 shows and the user ID gobby at NCTU recommend user ID bill at BOT to use the federated identity service.

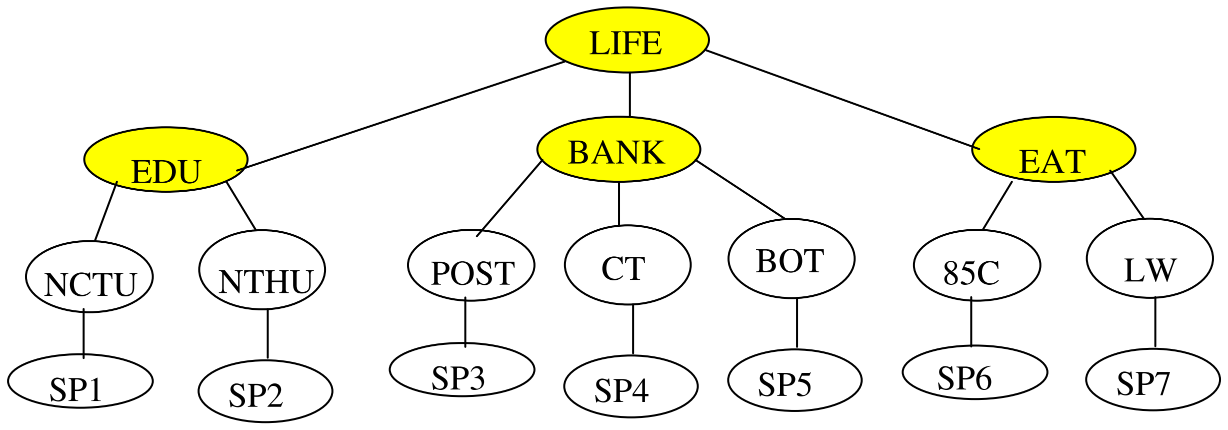


Figure 4.3.3 Topology of Commercial System

Briefly, the processing of petition is as follows:

1. The user logs in at NCTU as gobby, and requests the service provided by SP5 using SSO.
2. NCTU asks the global path of the target web, BOT, and BOT response its global path. (4.2.5)
3. Account linking

We will illustrate the step3 in detail immediately. After getting the global path of BOT, NCTU proceeds to BOT and operates account linking.

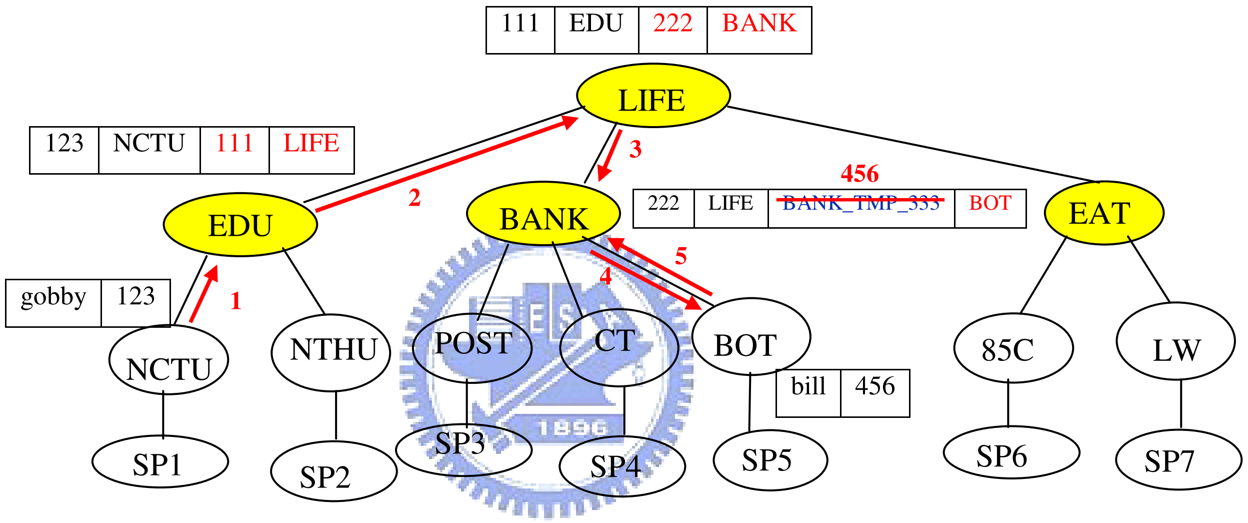


Figure 4.3.4 Processing Sequence of Account Linking

As Figure 4.3.4 demonstrates, the processing of account linking is as follows:

1. NCTU issues a request to its manager, EDU: "**(123)** at *NCTU* wants to visit *LIFE_BANK_BOT*".
2. EDU receives the request and compares the path string. The fact that its name, "EDU" does not appear indicates the target IDP (BOT) is not under its subtree. Thus EDU stores (123), NCTU, and a Link_ID (111) which is generated randomly by EDU and keeps on sending the request "**(111)** at *EDU* wants to visit *LIFE_BANK_BOT*" to upper layer.
3. Similarly, LIFE compares the path string after receiving the request. The name "LIFE" appears in the path means that the target IDP is in lower layer. And the next receiver is BANK according to the path string, "*LIFE_BANK_BOT* ". Thus LIFE looks up its

mapping table to determine whether a previous federation has been established or not. If LIFE does not find any record about this request, it stores (111), EDU, and a Link_ID (222) which is generated randomly by LIFE, and then keeping on sending the request "(**222)** at LIFE wants to visit *LIFE_BANK_BOT*" to BANK.

4. BANK receives the request and operates similarly as step 3. Because the previous federation has not been established, BANK stores (222), LIFE, and a **temporary** link_ID (BANK_TMP_333) generated randomly by BANK It keeps on sending the request "(BANK_TMP_333**)** at BANK wants to visit *LIFE_BANK_BOT*" to lower layer.
5. After the petitioner logs in as bill at BOT, BOT sends back the real Link_ID, (456), which represent the user account bill. The temporary Link_ID will be covered with (456) and the correct federation has been established.

4.3.2 Implementation Result

In the section, in order to enable readers to understand the actual operation circumstance of our architecture, we will illustrate the implementation result with execution pictures gradually.

In this case, a user has registered the account amy at BOT site and mary at NCTU site. BOT and NCTU sites are partners. The user ID amy at BOT wants to recommend user ID mary at NCTU to use the federated identity service. Figure 4.3.5 shows the topology.

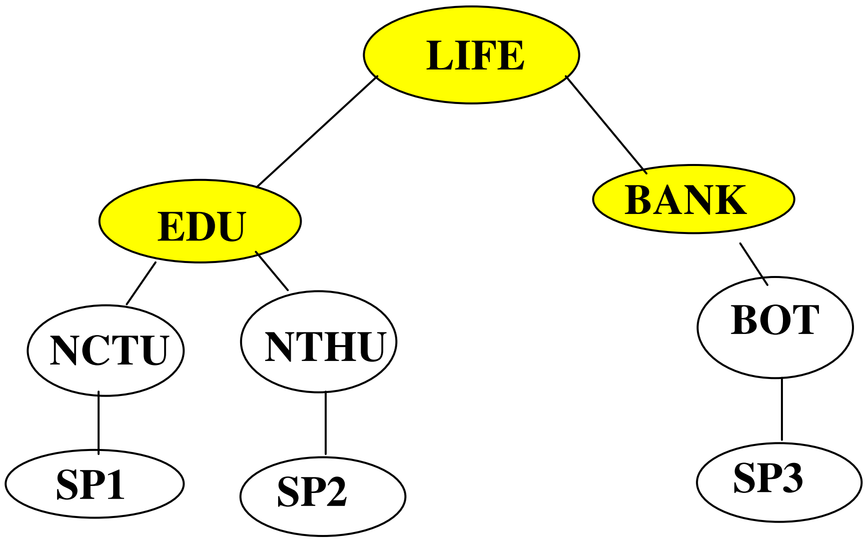
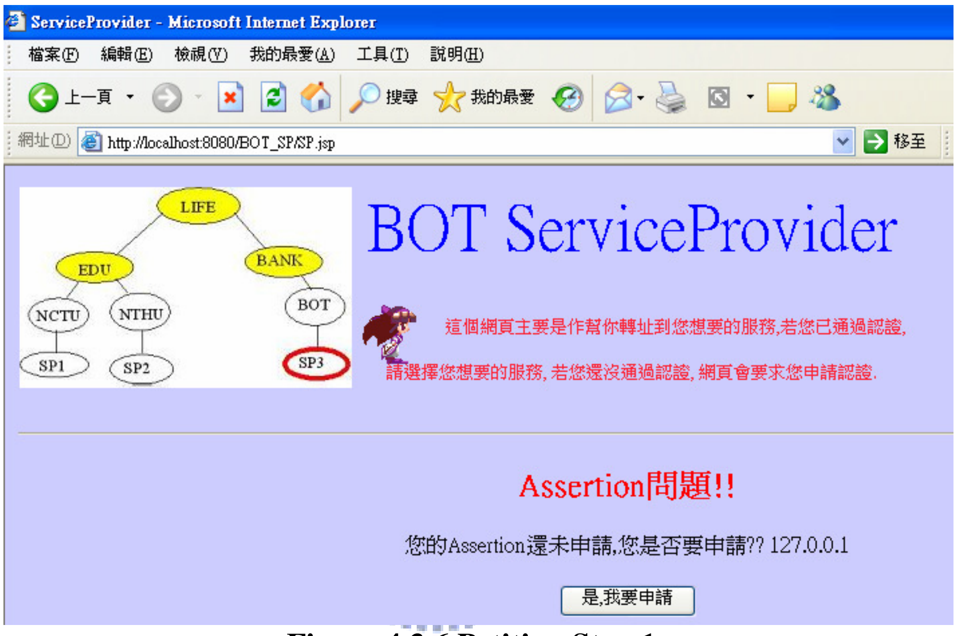


Figure 4.3.5 The Topology of Implemented Architecture

Following Figures will demonstrates the processing of petition and the identity federation result.

Step 1 : The user attempts to access resources on SP3 site (http://localhost:8080/BOT_SP/SP.jsp). Because he does not have any current logon session, SP3 sends the user to the identity provider (http://localhost:8080/BOT/IDAuth.jsp) and requests the IDP to provide an assertion for the user.


ServiceProvider - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 搜尋 我的最愛
網址(D) http://localhost:8080/BOT_SP/SP.jsp
移至
LIFE
EDU
BANK
NCTU
NTHU
BOT
SP1
SP2
SP3

BOT ServiceProvider

這個網頁主要是作幫你轉址到您想要的服務,若您已通過認證,
請選擇您想要的服務, 若您還沒通過認證, 網頁會要求您申請認證.

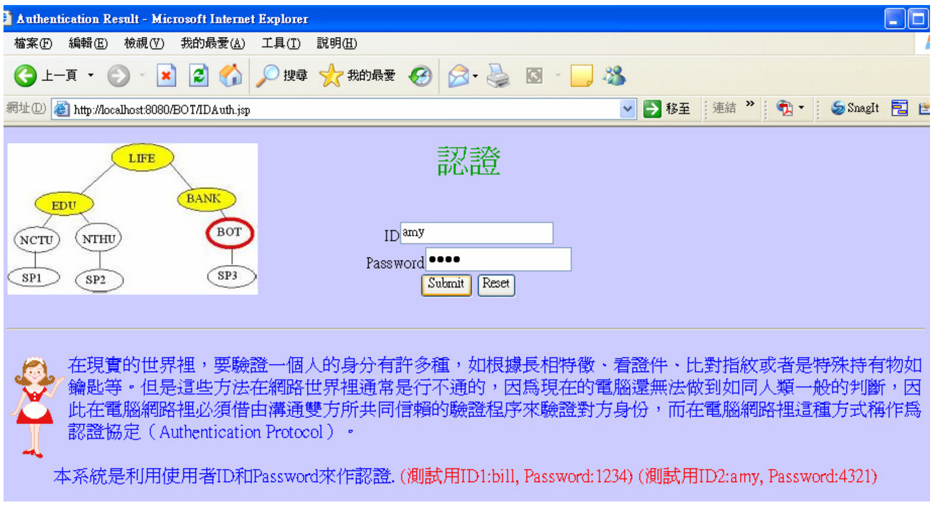
Assertion問題!!

您的Assertion還未申請,您是否要申請?? 127.0.0.1

是,我要申請

Figure 4.3.6 Petition Step 1

Step 2 : The user logs in as ***amy*** user account and provides valid credentials.


Authentication Result - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 搜尋 我的最愛
網址(D) http://localhost:8080/BOT/IDAuth.jsp
移至 連結 SnagIt
LIFE
EDU
BANK
NCTU
NTHU
BOT
SP1
SP2
SP3

認證

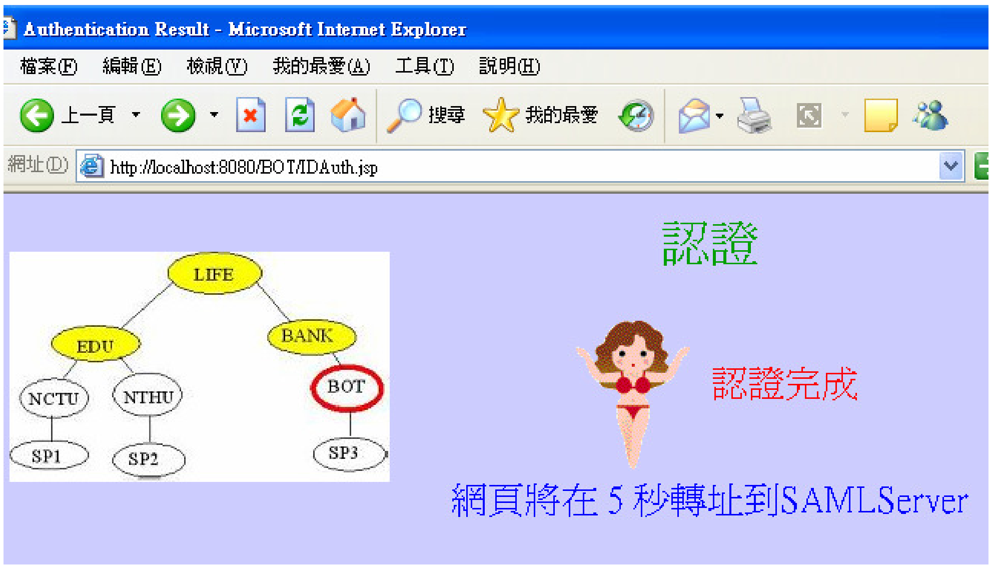
ID amy
Password ••••
Submit Reset

在現實的世界裡，要驗證一個人的身分有許多種，如根據長相特徵、看證件、比對指紋或者是特殊持有物如鑰匙等。但是這些方法在網路世界裡通常是行不通的，因爲現在的電腦還無法做到如同人類一般的判斷，因此在電腦網路裡必須借由溝通雙方所共同信賴的驗證程序來驗證對方身份，而在電腦網路裡這種方式稱作爲認證協定（Authentication Protocol）。

本系統是利用使用者ID和Password來作認證. (測試用ID1:bill, Password:1234) (測試用ID2:amy, Password:4321)

Figure 4.3.7 Petition Step 2

Step3 : The user logs in successfully and the BOT redirects him to the SAML server.


Authentication Result - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁
搜尋
我的最愛
網址(D)
LIFE
EDU
BANK
NCTU
NTHU
BOT
SP1
SP2
SP3

http://localhost:8080/BOT/IDAuth.jsp

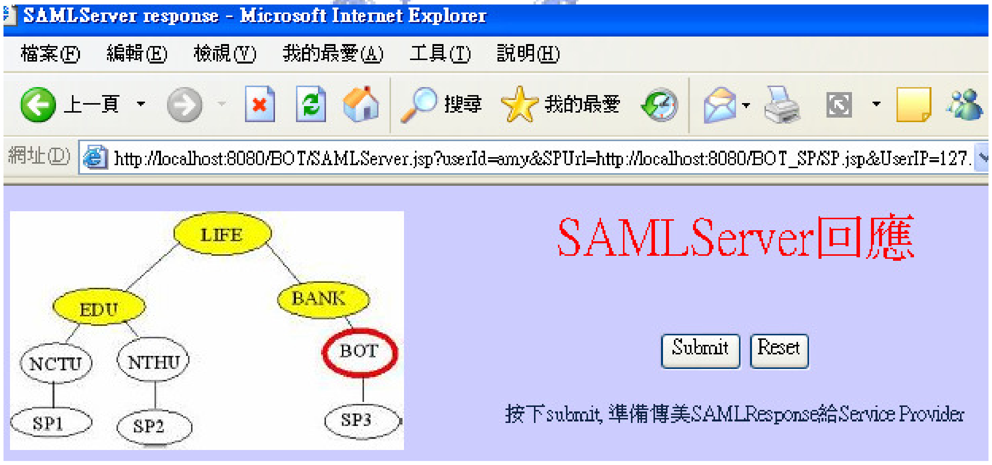
認證

認證完成

網頁將在 5 秒轉址到SAMLServer

Figure 4.3.8 Petition Step 3

Step 4 : The BOT builds a SAML assertion for the user.


SAMLServer response - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁
搜尋
我的最愛
網址(D)
LIFE
EDU
BANK
NCTU
NTHU
BOT
SP1
SP2
SP3
Submit
Reset

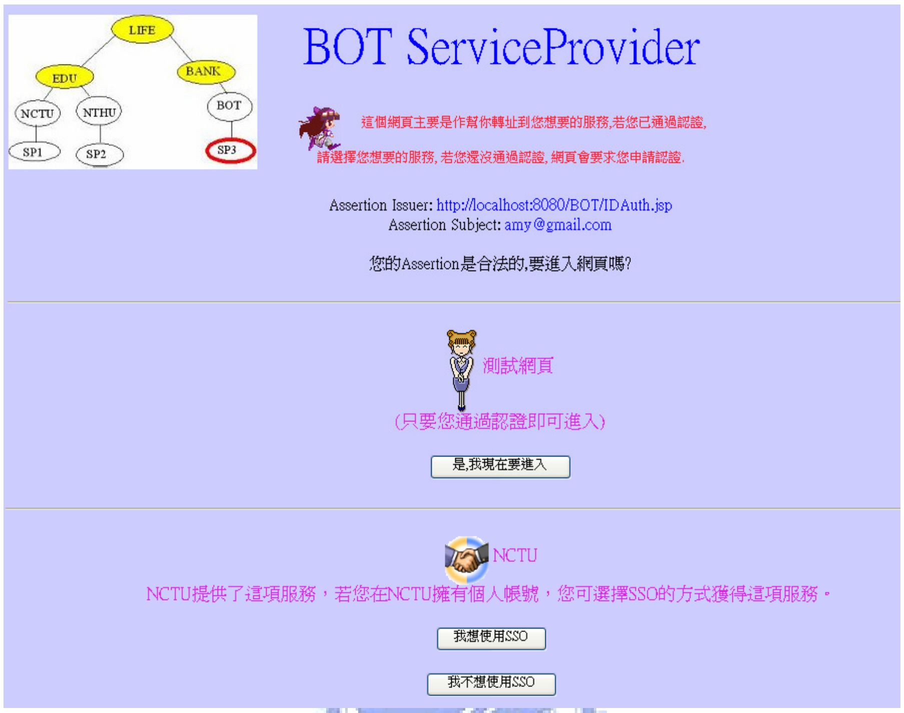
http://localhost:8080/BOT/SAMLServer.jsp?userId=amy&SPUrl=http://localhost:8080/BOT_SP/SP.jsp&UserIP=127.

SAMLServer回應

按下submit, 準備傳美SAMLResponse給Service Provider

Figure 4.3.9 Petition Step 4

Step 5 : The browser receives the assertion and issues a request to send to SP3. The user can access the resource at SP3. BOT and NCTU sites are partners. The user can recommend user ID ***mary*** at NCTU to use the federated identity service.



BOT ServiceProvider

這個網頁主要是作幫你轉址到您想要的服務,若您已通過認證,
請選擇您想要的服務, 若您還沒通過認證, 網頁會要求您申請認證.

Assertion Issuer: http://localhost:8080/BOT/IDAuth.jsp
Assertion Subject: amy@gmail.com

您的Assertion是合法的,要進入網頁嗎?

測試網頁

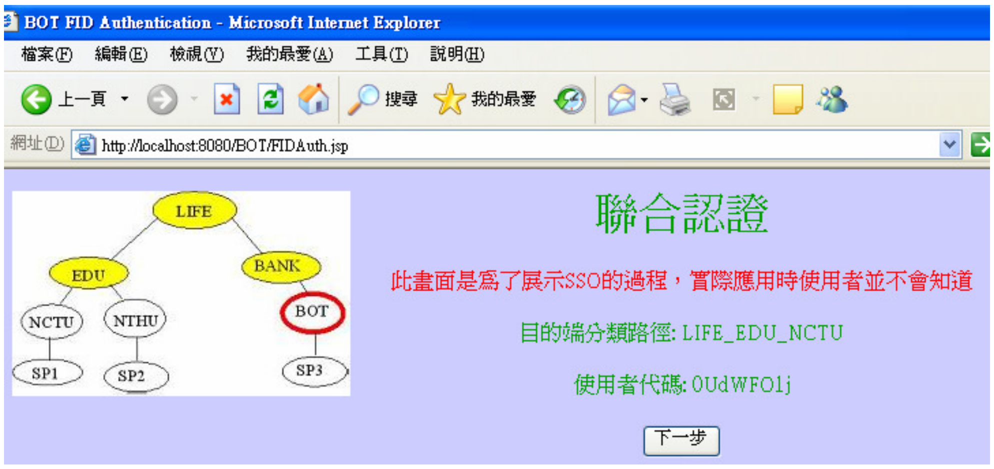
(只要您通過認證即可進入)

NCTU

NCTU提供了這項服務，若您在NCTU擁有個人帳號，您可選擇SSO的方式獲得這項服務。

Figure 4.3.10 Petition Step 5

Step 6 : BOT sends a request to NCTU to get NCTU`s global path, “LIFE_EDU_NCTU”. After getting the path string, BOT issues a request to its manager, BANK : “(**0UdWFOlj)** which is represented the user ***amy*** at *NCTU* wants to visit *LIFE_EDU_NCTU*”.


BOT FID Authentication - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁
搜尋
我的最愛
網址(D)
LIFE
EDU
BANK
NCTU
NTHU
BOT
SP1
SP2
SP3

http://localhost:8080/BOT/FIDAuth.jsp

聯合認證

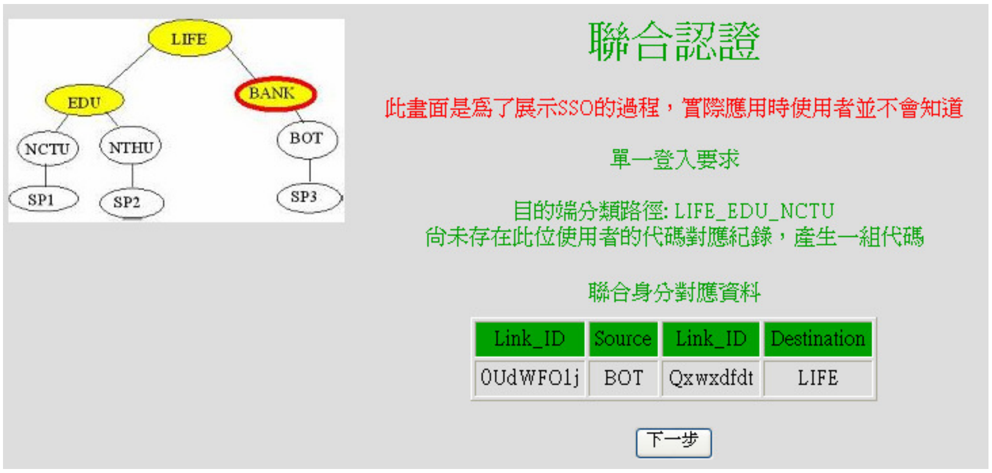
此畫面是爲了展示SSO的過程，實際應用時使用者並不會知道

目的端分類路徑: LIFE_EDU_NCTU

使用者代碼: 0UdWFOlj

Figure 4.3.11 Petition Step 6

Step 7 : BANK receives the request and compares the path string. Because the name “BANK” does not appear in the path sting, thus BANK stores (**0UdWFOlj**), NCTU, and a Link_ID (**Qxwxdfdt**) which is generated randomly by BANK and keeps on sending the request “(**Qxwxdfdt)** at *BANK* wants to visit *LIFE_EDU_NCTU*” to upper layer.



聯合認證

此畫面是爲了展示SSO的過程，實際應用時使用者並不會知道

單一登入要求

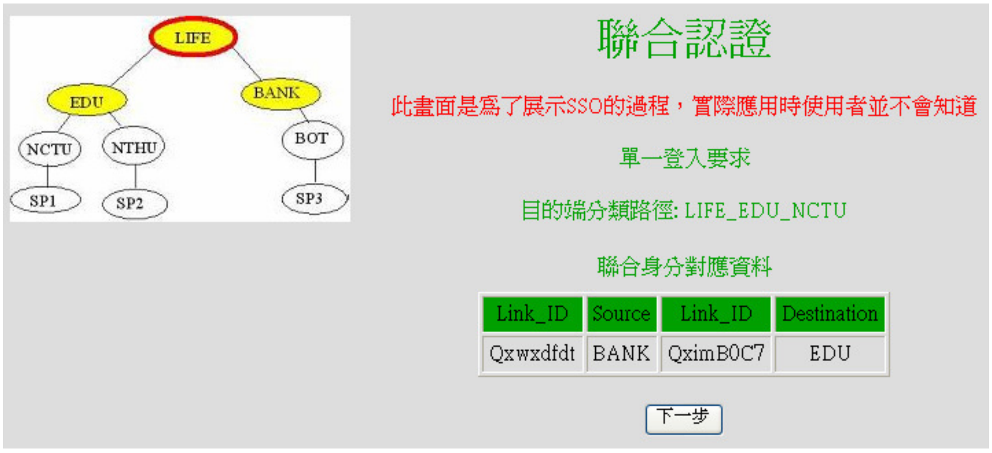
目的端分類路徑: LIFE_EDU_NCTU
尚未存在此位使用者的代碼對應紀錄，產生一組代碼

聯合身分對應資料

Link_ID	Source	Link_ID	Destination
0UdWFO1j	BOT	Qxwxdfdt	LIFE

Figure 4.3.12 Petition Step 7

Step 8 : LIFE compares the path string after receiving the request. Because the name “LIFE” appears in the path , thus LIFE looks up its mapping table to determine whether a previous federation has been established or not. If LIFE does not find any record about this request, it stores (**Qxwxdfdt**), EDU, and a Link_ID (**QximB0C7**) which is generated randomly by LIFE, and then keeping on sending the request “(**QximB0C7)** at LIFE wants to visit *LIFE_EDU_NCTU*” to EDU.



聯合認證

此畫面是爲了展示SSO的過程，實際應用時使用者並不會知道

單一登入要求

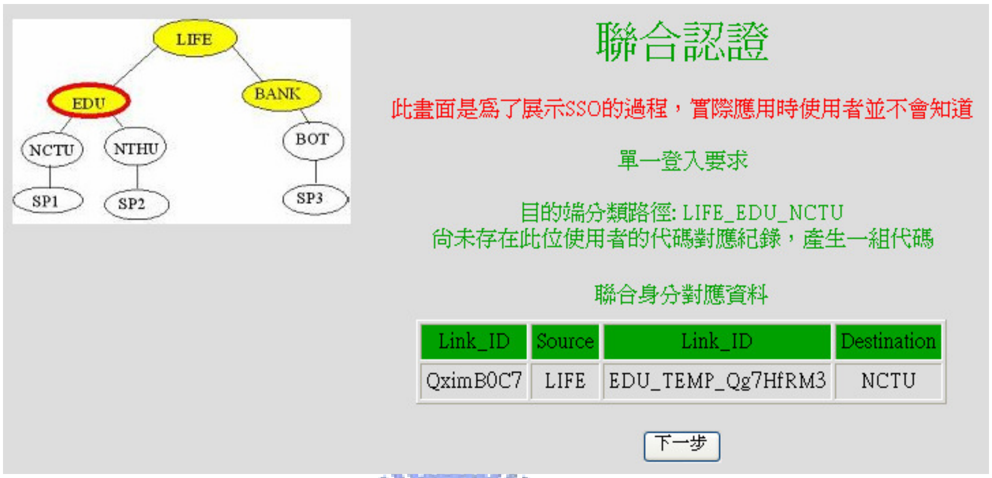
目的端分類路徑: LIFE_EDU_NCTU

聯合身分對應資料

Link_ID	Source	Link_ID	Destination
Qxwxdfdt	BANK	QximB0C7	EDU

Figure 4.3.13 Petition Step 8

Step 9 : EDU receives the request and operates similarly as step 8. Because the previous federation has not been established, EDU stores (**QximB0C7**), LIFE, and a **temporary** link_ID (**BANK_TMP_ Qg7HfRM3**) generated randomly by EDU It keeps on sending the request "(**BANK_TMP_ Qg7HfRM3)** at EDU wants to visit *LIFE_EDU_NCTU*" to lower layer.



聯合認證

此畫面是為了展示SSO的過程，實際應用時使用者並不會知道

單一登入要求

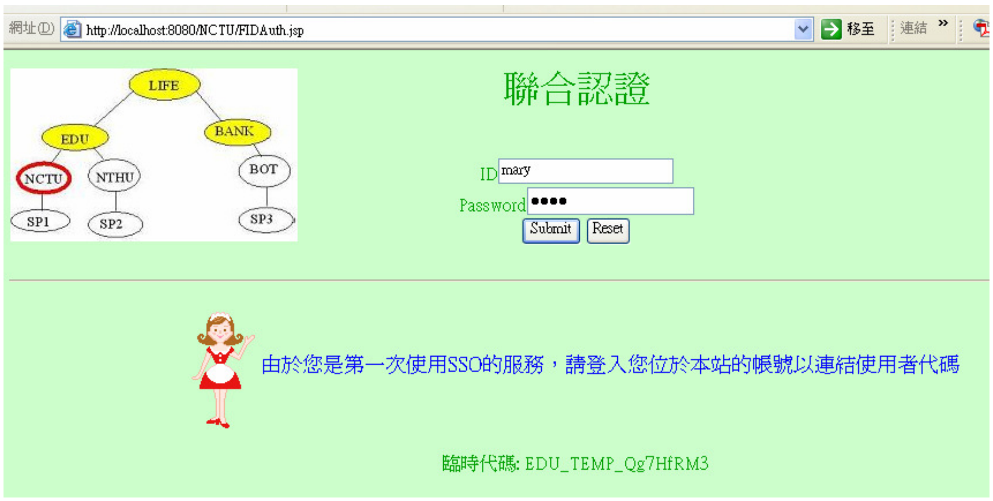
目的端分類路徑: LIFE_EDU_NCTU
尚未存在此位使用者的代碼對應紀錄，產生一組代碼

聯合身分對應資料

Link_ID	Source	Link_ID	Destination
QximB0C7	LIFE	EDU_TEMP_Qg7HfRM3	NCTU

Figure 4.3.14 Petition Step 9

Step 10 : The user logs in as ***mary*** at NCTU.



http://localhost:8080/NCTU/FIDAuth.jsp

聯合認證

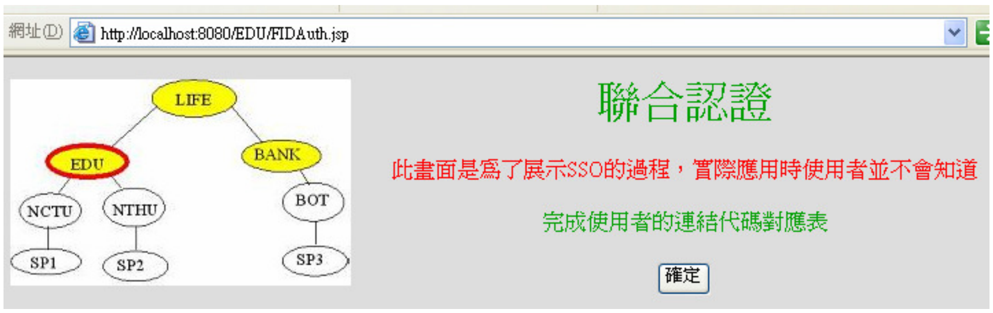
ID	mary
Password	●●●●

由於您是第一次使用SSO的服務，請登入您位於本站的帳號以連結使用者代碼

臨時代碼: EDU_TEMP_Qg7HfRM3

Figure 4.3.15 Petition Step 10

Step 11 : NCTU sends back the Link_ID, (**um4DO4Hk**) which represent the user account mary. The temporary Link_ID will be covered with (**um4DO4Hk**) and the correct federation has been established.



聯合認證

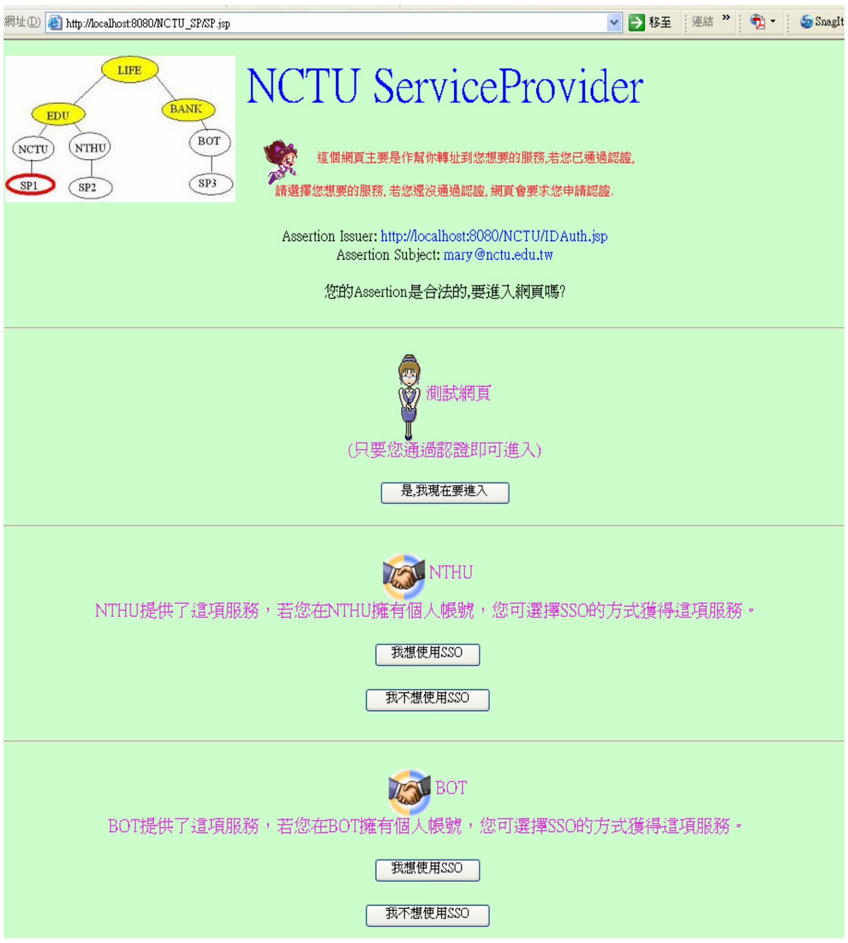
此畫面是爲了展示SSO的過程，實際應用時使用者並不會知道

完成使用者的連結代碼對應表

Figure 4.3.16 Petition Step 11

Now, let`s regard the operation of identity federation.

Step 12 : The use accesses resources at SP1 and logs in as ***mary*** at NCTU.



NCTU ServiceProvider

這個網頁主要是作幫你轉址到您想要的服務,若您已通過認證,

請選擇您想要的服務, 若您還沒通過認證, 網頁會要求您申請認證.

Assertion Issuer: http://localhost:8080/NCTU/IDAuth.jsp
Assertion Subject: mary@nctu.edu.tw

您的Assertion是合法的,要進入網頁嗎?

測試網頁

(只要您通過認證即可進入)

NTHU

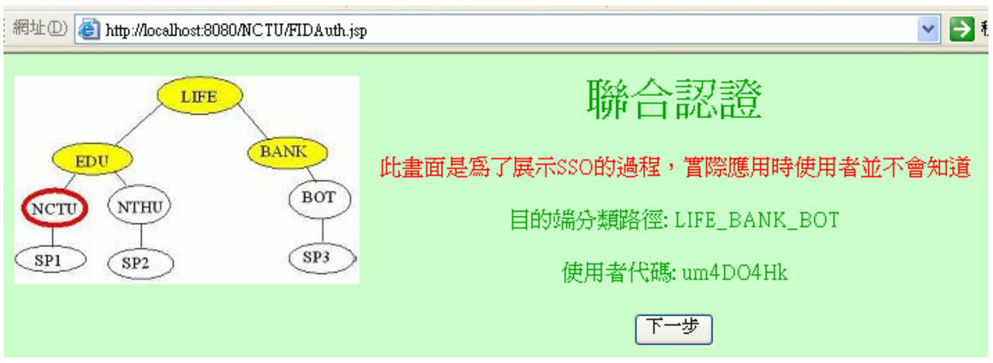
NTHU提供了這項服務，若您在NTHU擁有個人帳號，您可選擇SSO的方式獲得這項服務。

BOT

BOT提供了這項服務，若您在BOT擁有個人帳號，您可選擇SSO的方式獲得這項服務。

Figure 4.3.17 Identity Federation Step 1

Step 13 : Then the user wants to visit BOT. NCTU issues a request to BOT to get BOT`s global path. After getting the path string, "*LIFE_BANK_BOT*", NCTU issues a request to manager: "(**um4DO4Hk**) at *NCTU* wants to visit *LIFE_BANK_BOT*".



http://localhost:8080/NCTU/FIDAuth.jsp

聯合認證

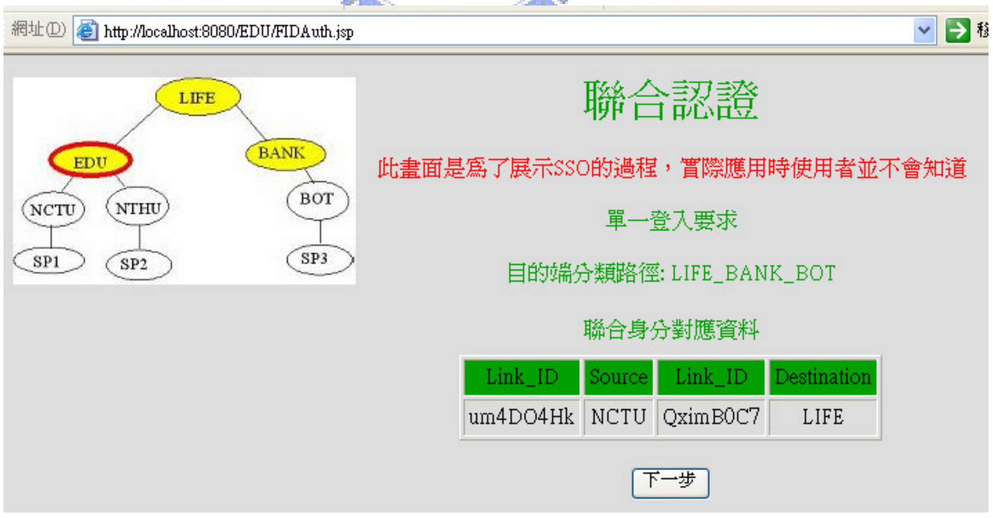
此畫面是為了展示SSO的過程，實際應用時使用者並不會知道

目的端分類路徑: LIFE_BANK_BOT

使用者代碼: um4DO4Hk

Figure 4.3.18 Identity Federation Step 2

Step 14 : EDU receives the request and then looks up its account linking table to determine whether a previous federation has been established. It finds (**um4DO4Hk**) at NCTU is (**Qxim4B0C7**) at *LIFE*. Then EUD issues a request to LIFE: "(**QximB0C7**) at *EDU* wants to visit *LIFE_BANK_BOT*".



http://localhost:8080/EDU/FIDAuth.jsp

聯合認證

此畫面是為了展示SSO的過程，實際應用時使用者並不會知道

單一登入要求

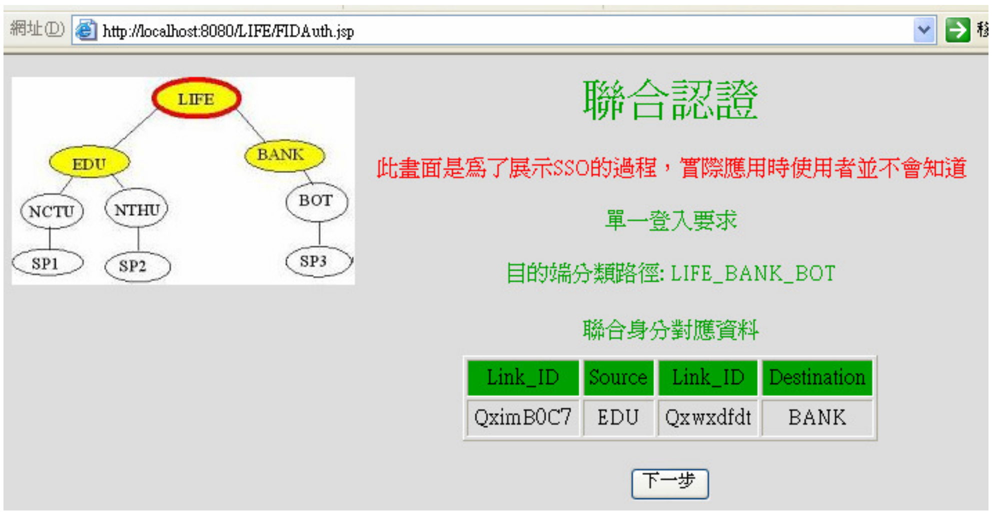
目的端分類路徑: LIFE_BANK_BOT

聯合身分對應資料

Link_ID	Source	Link_ID	Destination
um4DO4Hk	NCTU	QximB0C7	LIFE

Figure 4.3.19 Identity Federation Step 3

Step 15 : LIFE receives the request and finds that (**QximB0C7**) at EDU is (**Qxwxdfdt**) at BANK by looking up its account linking table. Then LIFE issues a request to BANK: "(**Qxwxdfdt**) at *EDU* wants to visit *LIFE_BANK_BOT*".



http://localhost:8080/LIFE/FIDAuth.jsp

聯合認證

此畫面是爲了展示SSO的過程，實際應用時使用者並不會知道

單一登入要求

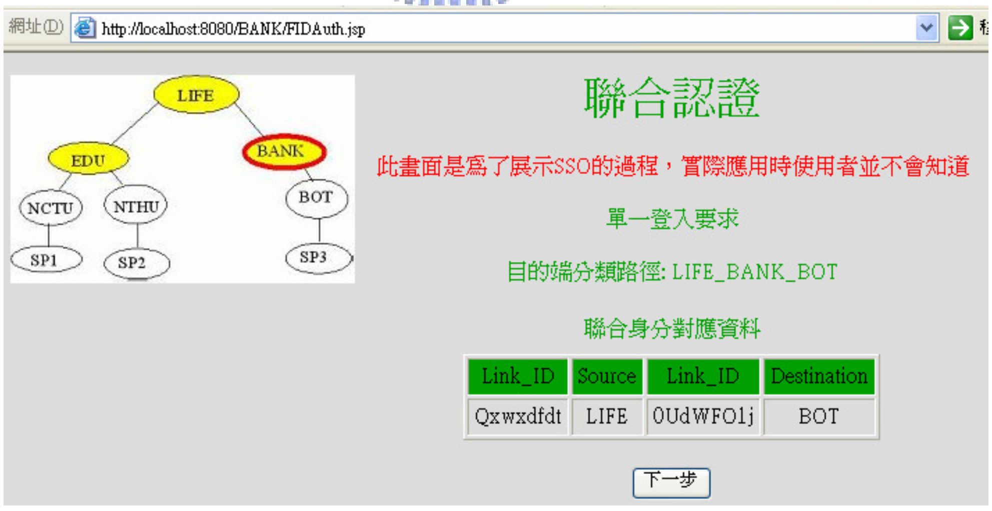
目的端分類路徑: LIFE_BANK_BOT

聯合身分對應資料

Link_ID	Source	Link_ID	Destination
QximB0C7	EDU	Qxwxdfdt	BANK

Figure 4.3.20 Identity Federation Step 4

Step 16 : BANK receives the request and finds that (**Qxwxdfdt**) at EDU is (**0UdWFOlj**) at BOT by looking up its account linking table. Then BANK issues a assertion to BOT: "(**0UdWFOlj**) is authenticated".



http://localhost:8080/BANK/FIDAuth.jsp

聯合認證

此畫面是爲了展示SSO的過程，實際應用時使用者並不會知道

單一登入要求

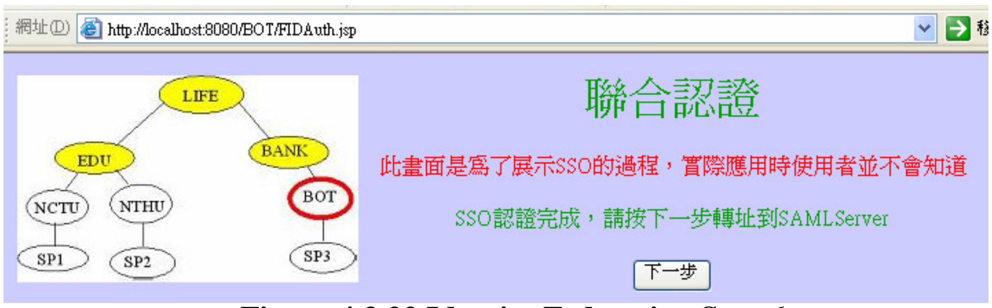
目的端分類路徑: LIFE_BANK_BOT

聯合身分對應資料

Link_ID	Source	Link_ID	Destination
Qxwxdfdt	LIFE	0UdWFOlj	BOT

Figure 4.3.21 Identity Federation Step 5

Step 17 : BOT receives the request and finds which user the Link_ID (**0UdWFOlj**) is represented.



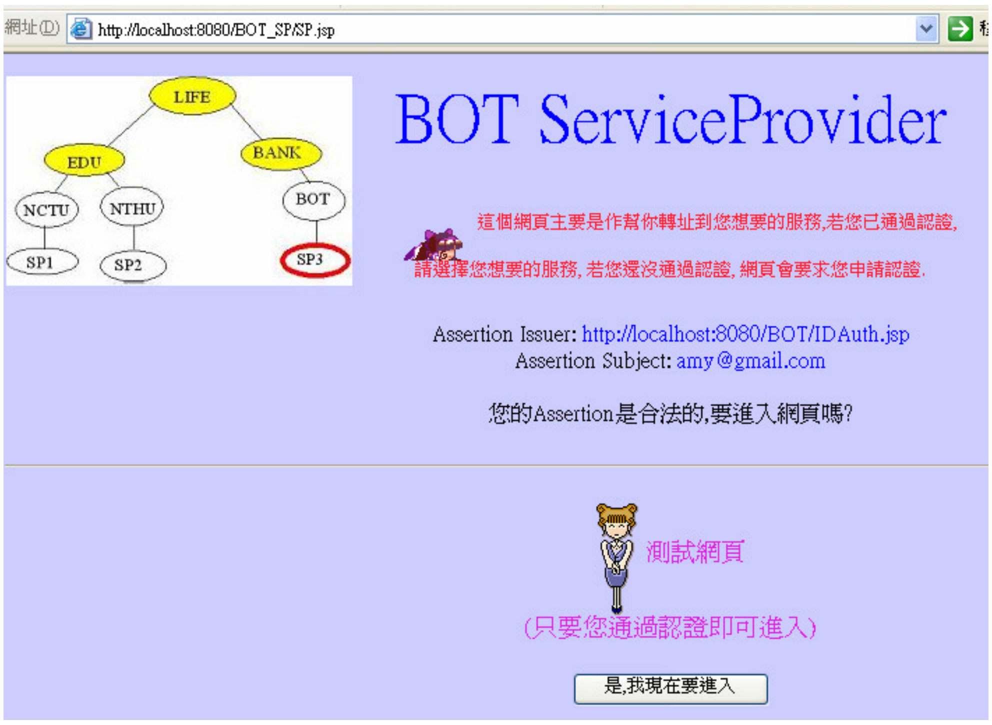
聯合認證

此畫面是爲了展示SSO的過程，實際應用時使用者並不會知道

SSO認證完成，請按下一步轉址到SAMLServer

Figure 4.3.22 Identity Federation Step 6

Step 18 : The user mary associated with the Link_ID (**0UdWFOlj**) has the correct authorization.



BOT ServiceProvider

這個網頁主要是作幫你轉址到您想要的服務,若您已通過認證,請選擇您想要的服務,若您還沒通過認證,網頁會要求您申請認證.

Assertion Issuer: http://localhost:8080/BOT/IDAuth.jsp
Assertion Subject: amy@gmail.com

您的Assertion是合法的,要進入網頁嗎?

測試網頁

(只要您通過認證即可進入)

Figure 4.3.23 Identity Federation Step 7

Chapter 5 System Analysis

5.1 Security

All information that delivered is Link_ID rather than real credentials. The source IDP and destination IDP neither know the ID used on the other side. It provides privacy-preserving characteristic for users.

5.2 Expansibility

The expansibility property can be discussed in vertical and horizontal dividedly. Each manager must record the nodes of its upper layer and lower layers. We can take advantage of it to insert or delete business systems. The following Figures can illustrate the concepts.

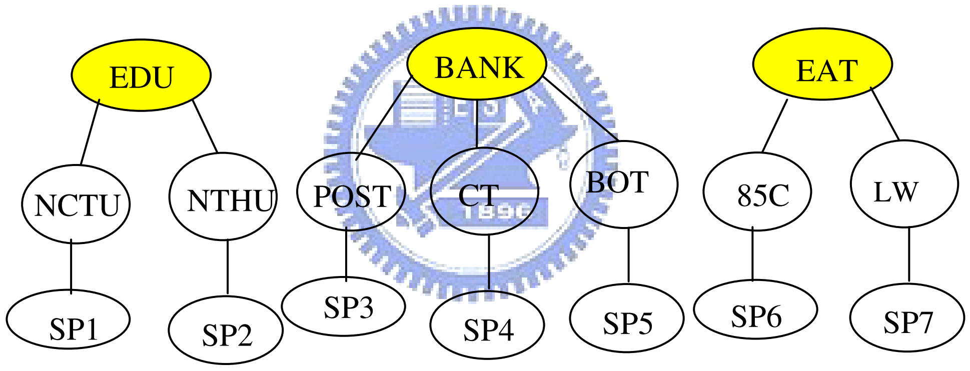


Figure 5.2.1 Initial Topology

Assume that a relationship exists as Figure 5.2.1. Someday all three groups want to cooperate in demand, then the manager LIFE will be constructed in need to integrate them. Figure 5.2.2 demonstrates the alteration of the net topology.

As Table 5.2.1 shows, the tables should be modified according to present topology, as mentioned in section 4.2.2. It can solve vertical expansibility problem.

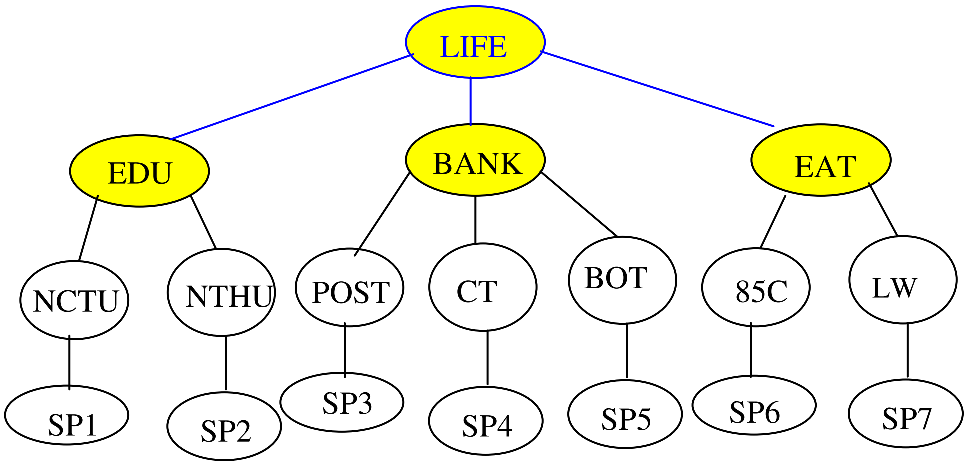


Figure 5.2.2 Topology State after Vertical Expansibility

LIFE	
1	EDU
1	BANK
1	EAT

EAT	
1	85C
1	LW

→

EAT	
0	LIFE
1	85C
1	LW

Table 5.2.1 Modification after Vertical Expansibility

As for the horizontal expansibility, the operation is similar to above. If another group "PLAY" wants to join in the commercial system, the modification is shown in Figure 5.2.3and Table 5.2.2.

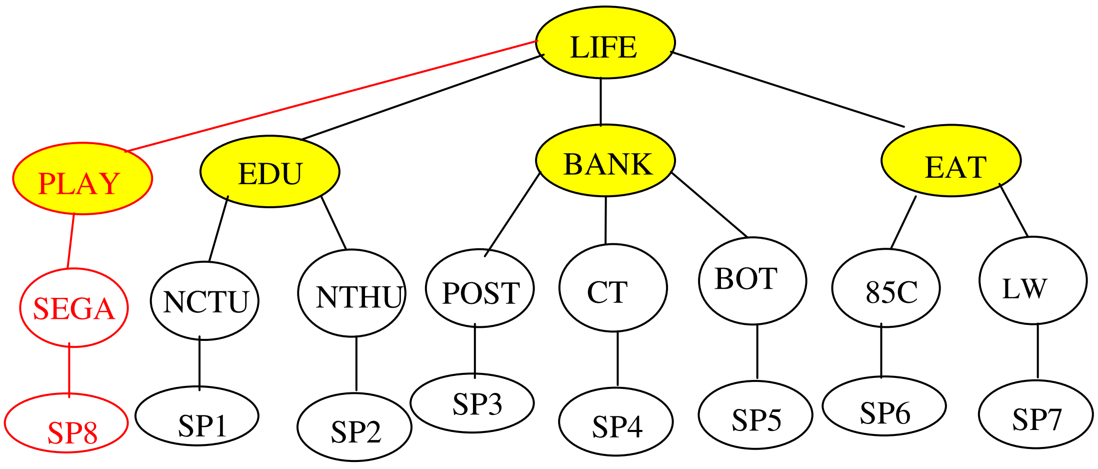


Figure 5.2.3 Topology State after Horizontal Expansibility

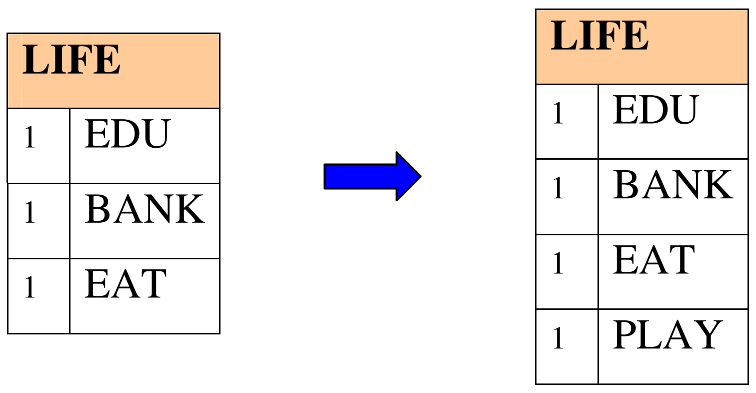


Figure 5.2.2 Modification after Horizontal Expansibility

5.3 Robustness

In the section, we would consider the influence when a manager is destroyed in the business system. Most probably account linking in the node would not be operated. But it can work without involving the crashed node, including account linking.

Besides, we consider another situation. That is, there is a petition to a certain manager and the connected was interrupted. The Link_ID which is generated previously would not disappear. After connecting successfully, the Link_ID can be used for federation.

Chapter 6 Conclusion and Future Work

6.1 Conclusion

Using the security features as well as the pseudonym concepts of SAML provide the users a enterprise-crossed and integrated service, moreover, it enables the users to achieve SSO under many identity providers. All identity federations are established by managers and dispersed evenly. The problem that a certain manager maintains all identity federation which results in exhaustion of the manager is not considered. Comparing traditional SSO, regardless of sorting or space storage is more efficient in our system architecture.

6.2 Future Work

When a manager is destroyed in the business system, it cannot operate involving the crashed node. We should develop a secure, efficient method to re-build the destroyed manager and identity federation records established previously.

References:

[1] INTRODUCTION TO WEB SERVICES, 2004
http://dev2dev.bea.com/pub/a/2004/02/introwebsvcs.html

[2] Introduction to Web Services By Lakshmi Ananthamurthy
http://www.developer.com/services/article.php/1485821

[3] Single Sign On
http://www.opengroup.org/security/sso/

[4] Web Single Sign On Systems, 2006
http://www.cesnet.cz/doc/techzpravy/2006/web-sso/

[5] OASIS, Technical Overview of the OASIS Security Assertion Markup Language (SAML) V1.1, 2004

[6] OASIS, Security Assertion Markup Language (SAML) 2.0 Technical Overview, 2005

[7] OASIS, Security and Privacy Consideration for the OASIS Security Assertion Markup Language V1.1, 2003

Appendix A

A.1 Manager EDU

```
<%
    String myName = "EDU";    // manager 代號
    String upManager = "LIFE";    //  上層 manager 代號

    String FID_req_src = new String();
    String tree_code = new String();
    String prev_node = new String();
    String prev_lid = new String();
    String next_node = new String();
    String next_lid = new String();

    String actSTR = request.getParameter( "Action" );
    int act = -1;
    if (actSTR != null) act = Integer.parseInt(actSTR);

    switch( act )
    {
        /*  找出分類路徑  */
        case 21:
        FID_req_src = request.getParameter( "FID_src" );
        tree_code = request.getParameter( "TreeCode" );

        if( FID_req_src == null )
            out.println("錯誤，沒有來源端位址");
        else {
            out.println("<p align = center>分類路徑要求，來源端:
                                    "+FID_req_src+"</p>");
            tree_code = myName + "_" + tree_code;
            out.println("<p align = center>目前分類路徑: "+tree_code+"</p>");
            //  沒有上層管理者，將分類路徑結果回傳給來源端
            if( upManager == null )
            {
```

```
%>
            <p align=center>沒有上層管理者，請按下一步回傳分類路徑
            <form method="post" action=<%=FID_req_src%> >
            <input type="hidden" name="Action" value=22 >
            <input type="hidden" name="TreeCode" value=<%=tree_code%> >
            <input type="submit" value = "下一步"/>
            </form></p>
<%
            }
            else   // 將分類路徑字串傳給上層管理者繼續編碼
            {
%>
            <p align = center>仍有上層管理者，請按下一步繼續編碼
            <form method="post" action="http://localhost:
                        8080/<%=upManager%>/FIDAuth.jsp") >
            <input type="hidden" name="Action" value=21 >
            <input type="hidden" name="FID_src" value=<%=FID_req_src%> >
            <input type="hidden" name="TreeCode" value=<%=tree_code%> >
            <input type="submit" value = "下一步"/>
            </form></p>
<%
            }
      }
      break;

      /* SSO 要求  */
      case 31:
      out.println("<p align = center>單一登入要求</p>");
      FID_req_src = request.getParameter( "FID_src" );
      prev_node = request.getParameter( "Previous_node" );
      prev_lid = request.getParameter( "Previous_LinkID" );
      tree_code = request.getParameter( "TreeCode" );

      out.println("<p align = center>");
      out.println("目的端分類路徑: " + tree_code + "<br>");

      /*  分析路徑代碼，找出自己的 name */
      int name_index = -1;
```

```
String[] tokens = tree_code.split("_");
for (int i = 0 ; i <tokens.length ; i++ ) {
        if( tokens[i].equals(myName) ) {
                name_index = i;
        }
}

/*  找出下次傳送的節點  */
if( name_index > -1 )    //  有找到自己的 name，往下層傳送
        next_node = tokens[name_index+1];
else    //  沒有找到自己的 name，往上層傳送
        next_node = upManager;

/*  連結資料庫，找出使用者的 link_id */
/*  尚未實現索引搜尋  */
Class.forName( "com.mysql.jdbc.Driver" )
                                    .newInstance();
String url = "jdbc:mysql://localhost:3306/"+myName;
String user = "root";
String password = "crypto123";
Connection con = DriverManager.getConnection( url , user , password );
Statement stmt = con.createStatement();
String query = "select * from mapping";
ResultSet rs = stmt.executeQuery( query );

//  是否存在使用者的對應紀錄
int mapping_exist = -1;
while( rs.next() )
{
        if( prev_node.equals( rs.getString("Src_IDP") ) )
        {
                if( prev_lid.equals(rs.getString(
                                "Link_ID_Src")) &&
                        next_node.equals( rs.getString(
                                "Dst_IDP"))    )
                {
                        next_lid = rs.getString("Link_ID_Dst");
                        mapping_exist = 1;
                break;
            }
        }
        else
        if( prev_node.equals( rs.getString("Dst_IDP") ) )
        {
            if( prev_lid.equals( rs.getString(
                        "Link_ID_Dst") ) &&
                next_node.equals( rs.getString(
                        "Src_IDP") )        )
            {
                next_lid = rs.getString("Link_ID_Src");
                mapping_exist = 1;
                break;
            }
        }
    }

    String nidp_addr = "http://localhost:8080/" + next_node + "/FIDAuth.jsp";
    act = 31;
    if( name_index > -1 )   // 有找到自己的 name，往下層傳送
    {
        // 代碼對應紀錄不存在，新增一筆資料
        if( mapping_exist != 1 )
        {
            out.println("尚未存在此位使用者的代碼對應紀錄，產生一組代碼
                            <br>");

            /*  產生一組代碼  */
            Random rand = new Random();
            String charset = "abcdefghijklmnopqrstuvwxyz
                    ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
            next_lid = "";
            for (int i = 0; i < 8; i++)
                next_lid += charset.charAt(rand.nextInt(
                                    charset.length()));

            /*  是否為第一層 manager? */
```

```
                if( name_index==(tokens.length-2) ) {
                    // 改用臨時代碼，並要求 IDP 回傳正確代碼
                    next_lid = myName + "_TEMP_" + next_lid;
                    act = 32;
                }
                /*  新增一筆資料到資料庫裡面  */
                query = "insert into mapping (Link_ID_Src,
                                Src_IDP,Link_ID_Dst,Dst_IDP) " +
                   "values ('" + prev_lid + "','" + prev_node +
                   "','" + next_lid + "','" + next_node + "')";
                stmt.executeUpdate( query );
            }
        }
        else   // 沒有找到自己的 name，往上層傳送
        {
            // 代碼對應紀錄不存在，新增一筆資料
            if( mapping_exist != 1 )
            {
                out.println("尚未存在此位使用者的代碼對應紀錄，產生一組代碼
                        <br>");

                /*  產生一組代碼  */
                Random rand = new Random();
                String charset = "abcdefghijklmnopqrstuvwxyz
                        ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
                next_lid = "";
                for (int i = 0; i < 8; i++)
                    next_lid += charset.charAt(rand.nextInt(
                                        charset.length()));

                /*  新增一筆資料到資料庫裡面  */
                query = "insert into mapping (Link_ID_Src,
                                Src_IDP,Link_ID_Dst,Dst_IDP) " +
                   "values ('" + prev_lid + "','" + prev_node +
                   "','" + next_lid + "','" + next_node + "')";
                stmt.executeUpdate( query );
            }
        }
```

```
            out.println("</p>");
%>
            <p align = center>
            <TABLE border=1 width=300 CELLPADDING=3>
            <CAPTION>
            <font color=glay>聯合身分對應資料</font>
            </CAPTION>
            <TR align=center bgcolor=glay>
            <TD>Link_ID</TD>
            <TD>Source</TD>
            <TD>Link_ID</TD>
            <TD>Destination</TD>
            </TR>
            <TR align=center>
            <TD><%=prev_lid%></TD>
            <TD><%=prev_node%></TD>
            <TD><%=next_lid%></TD>
            <TD><%=next_node%></TD>
            </TR>
            </TABLE></p>

            <p align = center>
            <form method="post" action=<%=nidp_addr%> >
            <input type="hidden" name="Action" value=<%=act%> >
            <input type="hidden" name="FID_src" value=<%=FID_req_src%> >
            <input type="hidden" name="Previous_node" value=<%=myName%> >
            <input type="hidden" name="Previous_LinkID" value=<%=next_lid%> >
            <input type="hidden" name="TreeCode" value=<%=tree_code%> >
            <input type="submit" value = "下一步"/>
            </form></p>
<%
            break;

            /* 取代 SSO 臨時代碼為正確的使用者連結代碼 */
            case 32:
            String temp_lid = request.getParameter( "temp_lid" );
            String user_lid = request.getParameter( "user_lid" );
            String dst_IDP = request.getParameter( "dst_IDP" );

            /* 將正確的使用者代碼取代資料庫的臨時代碼 */
            /* 尚未實現索引搜尋 */
            Class.forName( "com.mysql.jdbc.Driver" )
                                .newInstance();
            url = "jdbc:mysql://localhost:3306/"+myName;
            user = "root";
            password = "crypto123";
            con = DriverManager.getConnection( url , user , password );
            stmt = con.createStatement();
            query = "select * from mapping";
            rs = stmt.executeQuery( query );

            int index = -1;
            while( rs.next() ) {
                 if( dst_IDP.equals(rs.getString("Dst_IDP")) &&
                     temp_lid.equals(rs.getString("Link_ID_Dst"))
                 {
                         index = Integer.parseInt( rs.getString(
                                         "User_Index") );
                 }
            }

            if( index != -1 ) {
                 query = "UPDATE mapping SET Link_ID_Dst = '" + user_lid + "' WHERE
                        User_Index = " + index;
                 stmt.executeUpdate( query );
             }

            /* 重新導向使用者回去以完成 SSO 的步驟 */
            String addr = "http://localhost:8080/" + dst_IDP + "/FIDAuth.jsp";
%>
            <p align = center>完成使用者的連結代碼對應表
            <form method="post" action=<%=addr%> >
            <input type="hidden" name="Action" value=31 >
            <input type="hidden" name="Previous_LinkID" value=<%=user_lid%> >
            <input type="submit" value = "確定"/>
```

```
          </form></p>
<%
          break;

          default:
               out.println("錯誤，無法辨識的動作");
          break;
     }
%>
```

A.2 IDP

```
<%
     String myName = "NCTU";    // IDP 代號
     String myAddr = "http://localhost:8080/NCTU/IDAuth.jsp";
     String myManager = "http://localhost:8080/EDU/
                                   FIDAuth.jsp";    //  上層網址
     String SAMLServerAddr = "http://localhost:8080/NCTU/
                                   SAMLServer.jsp";
     String SPUrl = "http://localhost:8080/NCTU_SP/SP.jsp";
     String _userAddr = request.getRemoteAddr();

     String uid = new String();
     String sNameID = new String();
     String FID_req_src = new String();
     String tree_code = new String();
     String user_lid = new String();

     String actSTR = request.getParameter( "Action" );
     int act = -1;
     if (actSTR != null) act = Integer.parseInt(actSTR);

     switch( act )
     {
          /*  收到其它 IDP 的路徑要求，向上層送出分類路徑要求  */
          case 21:
          out.println("<p align = center><font color=\"red\">此畫面是爲了展示 SSO 的過
                    程，" +
                "實際應用時使用者並不會知道</font></p>");
        FID_req_src = request.getParameter( "FID_src" );

        if( FID_req_src == null )
            out.println("錯誤，沒有來源端位址");
        else {
            out.println("<p align = center>分類路徑要求，來源端:
                            "+FID_req_src+"</p>");
            out.println("<p align = center>目前分類路徑: "+myName+"</p>");
%>
            <p align = center>
            <form method="post" action=<%=myManager%> >
            <input type="hidden" name="Action" value=21 >
            <input type="hidden" name="FID_src" value=<%=FID_req_src%> >
            <input type="hidden" name="TreeCode" value=<%=myName%> >
            <input type="submit" value = "下一步"/>
            </form></p>
<%
        }
        break;

        /*  接收目的端分類路徑字串  */
        case 22:
        out.println("<p align = center><font color=\"red\">此畫面是爲了展示 SSO 的過
                    程，" +
                "實際應用時使用者並不會知道</font></p>");

        tree_code = request.getParameter( "TreeCode" );
        out.println("<p align = center>目的端分類路徑: "+tree_code+"</p>");
        sNameID = request.getParameter( "user" );

        /*  連結資料庫，找出使用者的 link_id */
        Class.forName( "com.mysql.jdbc.Driver" )
                            .newInstance();
        String url = "jdbc:mysql://localhost:3306/"+myName;
        String user = "root";
        String password = "crypto123";
        Connection con = DriverManager.getConnection( url , user , password );
        Statement stmt = con.createStatement();
        String query = "select * from authentication";
        ResultSet rs = stmt.executeQuery( query );

        while( rs.next() )
            if( sNameID.equals( rs.getString( "User_email" ) ) )
            {
                user_lid = rs.getString("User_Code");
                break;
            }

        out.println("<p align = center>使用者代碼: "+user_lid+"</p>");
%>
        <p align = center>
        <form method="post" action=<%=myManager%> >
        <input type="hidden" name="Action" value=31 >
        <input type="hidden" name="FID_src" value=<%=myAddr%> >
        <input type="hidden" name="Previous_node" value=<%=myName%> >
        <input type="hidden" name="Previous_LinkID" value=<%=user_lid%> >
        <input type="hidden" name="TreeCode" value=<%=tree_code%> >
        <input type="submit" value = "下一步"/>
        </form></p>
<%
        break;

        /* SSO 要求 */
        case 31:
        out.println("<p align = center><font color=\"red\">此畫面是爲了展示 SSO 的過
                        程，" +
                    "實際應用時使用者並不會知道</font></p>");
        user_lid = request.getParameter( "Previous_LinkID" );

        //connect to database
        Class.forName( "com.mysql.jdbc.Driver" )
                            .newInstance();
        url = "jdbc:mysql://localhost:3306/"+myName;
        user = "root";
        password = "crypto123";
        con = DriverManager.getConnection( url , user , password );
        stmt = con.createStatement();
        query = "select * from authentication";
        rs = stmt.executeQuery( query );

        while( rs.next() )
            if((user_lid.equals(rs.getString("User_Code"))))
            {
                uid = rs.getString( "User_Id" );
                break;
            }

        if( uid.length()>0 ) {    // 使用者代碼正確
%>
        <p align = center>SSO 認證完成，請按下一步轉址到 SAMLServer
        <form method="post" action=<%=SAMLServerAddr%> >
        <input type="hidden" name="userId" value=<%=uid%> >
        <input type="hidden" name="SPUrl" value=<%=SPUrl%> >
        <input type="hidden" name="UserIP" value=<%=_userAddr%> >
        <input type="submit" value = "下一步"/>
        </form></p>
<%
        }
        else {    // 使用者代碼不正確
%>
            <p align = center>SSO 認證失敗，找不到代碼爲<%=user_lid%>的使用者，
                          請重新登入
            <form method="post" action=<%=myAddr%> >
            <input type="hidden" name="UserIP" value=<%=_userAddr%>>
            <input type="hidden" name="SPUrl" value=<%=SPUrl%> >
            <input type="submit" value = "確定"/>
            </form></p>
<%
        }
        break;

        /* 第一次 SSO 引薦 */
```

```
            case 32:
            String temp_lid = request.getParameter(
                                            "Previous_LinkID" );
            String id = request.getParameter( "id" );
            String pass = request.getParameter( "password" );

            // 執行登入的動作，並回傳 link id 給管理者
            if( (id==null) || (pass==null) ) {
%>
                <p align=center>
                <form method="post" action="http://localhost:
                                8080/<%=myName%>/FIDAuth.jsp"><br>
                ID<input name="id" type="text" value=""><br>
                Password<input name="password" type="password" value=""><br>
                <input type="hidden" name="Action" value=32 >
                <input type="hidden" name="Previous_LinkID" value=<%=temp_lid%> >
                <input type="submit" name="Submit" value="Submit">
                <input type="reset" value="Reset">
                </form></p>

                <HR><p align = center><font size="4" color="blue">
                <img src="picture/comment1.gif" align=center>
                由於您是第一次使用 SSO 的服務，請登入您位於本站的帳號以連結使用
                者代碼
                <BR><BR></font></p>
<%
                out.println("<p align = center><font color=glay>臨時代碼:
                                "+temp_lid+"</font></p>");
            }
            else {
                try {
                    //connect to database
                    Class.forName( "com.mysql.jdbc.Driver" )
                                        .newInstance();
                    url = "jdbc:mysql://localhost:3306/"+myName;
                    user = "root";
                    password = "crypto123";
                    con = DriverManager.getConnection( url , user , password );
                    stmt = con.createStatement();
                    query = "select * from authentication";
                    rs = stmt.executeQuery( query );

                    user_lid = "";
                    while( rs.next() )
                         if( (id.equals(rs.getString("User_Id"))) && (pass.equals(rs.getString(
                                        "User_Password"))) )
                         {
                              user_lid = rs.getString("User_Code");
                         }

                    if( user_lid.length() > 0 ) {
%>
                         <p align = center>
                         <img src="picture/succ1.gif" align=center>
                         <br>
                         <font size="4" color="red">認證完成，請按下一步回傳使用者代
                                碼給上層管理者</font><BR>
                         <form method="post" action=<%=myManager%>>
                         <input type="hidden" name="Action" value=32 >
                         <input type="hidden" name="user_lid" value=<%=user_lid%> >
                         <input type="hidden" name="temp_lid" value=<%=temp_lid%> >
                         <input type="hidden" name="dst_IDP" value=<%=myName%> >
                         <input type="submit" value = "下一步"/>
                         </form></p>
<%
                    }
                    else {
%>
                         <p align = center>認證失敗，請重新登入
                         <img src="picture/fail1.gif" align=center>
                         <form method="post" action="http://
                                   localhost:8080/<%=myName%>/
                                   FIDAuth.jsp">
                         <input type="hidden" name="Action" value=32 >
                         <input type="hidden" name="Previous_LinkID"
                                    value=<%=temp_lid%> >
                    <input type="submit" value = "確定"/>
                    </form></p>
<%
                }
                out.println("</td><td width='27%'></td></tr>
                                    </table></p>");
            }
            catch( SQLException sqle ) {
                out.println( "sqle=" + sqle + "<p>" );
                out.println( "sqle produce some error");
                sqle.printStackTrace();
            }
        }
        break;

        default:
            out.println("錯誤，無法辨識的動作");
        break;
    }
%>
```