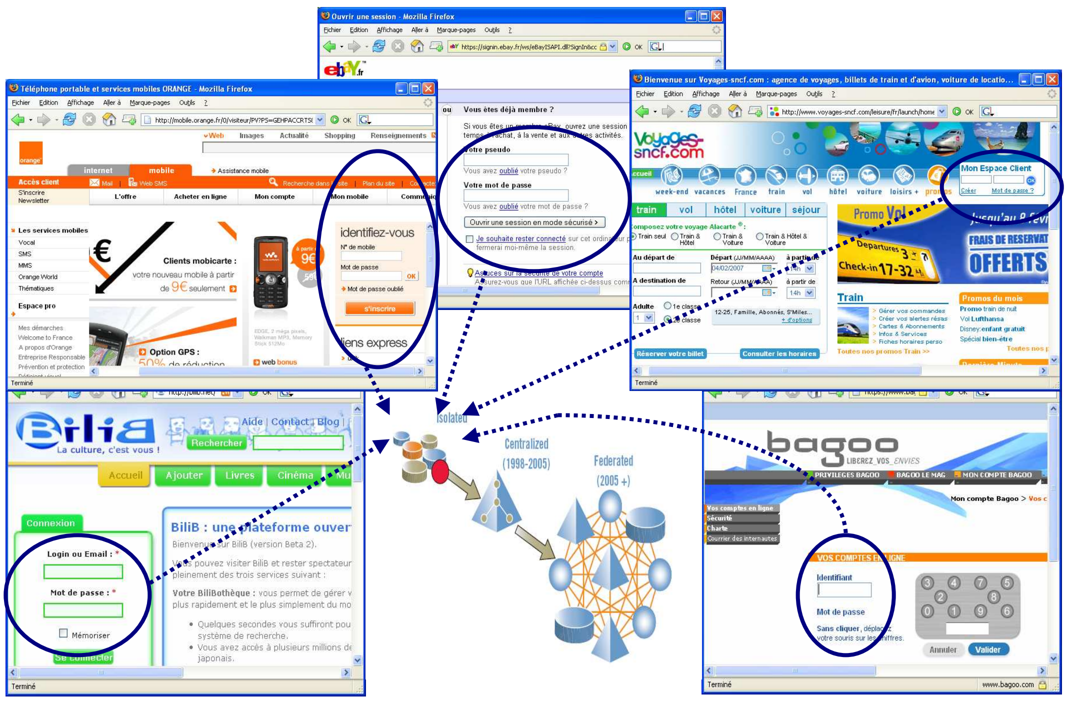
MONDESIR Eunice
WEILL-TESSIER Pierre

FEDERATED IDENTITY



ASR 2006/2007 Final Project

Supervisers: Maryline Maknavicius-Laurent, Guy Bernard

Federated Identity

Project topic

Superviser: Maryline Maknavicius

In order to allow users to access easier and faster to a set of services with always the same authentication parameters (often login/password), it is required to employ Federation Identity architectures solutions. From now on, several solutions are available, such as Shibboleth, Liberty Alliance and WS-Federation.

In a first step, the students will study the Federated Identity concepts and solutions, and later, will set up a platform based on Liberty Alliance (such as the open source software Ping Federate). Finally, the tests will be done according to the framework with a Mexican partner who has already installed a Liberty Alliance platform.

This project will be subject to a report. It will particularly include an installation/configuration user guide about Liberty Alliance. Since the project is in collaboration with Mexico, the report will preferably written in English.

References :

U. Fragoso-Rodriguez, M. Laurent-Maknavicius, J. Incera-Dieguez,
"Federated Identity Architectures ", 1st Mexican Conference on
Informatics Security 2006 MCIS’2006, International Conference approved
by IEEE Computer Society, Oaxaca, Mexico, November 2006.

This document is composed by two parts:

- a report about the project framework
- installation/configuration user guide

MONDESIR Eunice
WEILL-TESSIER Pierre

PROJECT FRAMEWORK

ASR 2006/2007 Final Project

Supervisors: Maryline Maknavicius-Laurent, Guy Bernard

SOMMAIRE

I. FEDERATED IDENTITY

Identity is one of the most crucial issues when speaking about security with applications. Being a company or a standard user, the security concerning credential exchanges and use need to be reliable and efficient when private information is requested by another entity.

Federation Identity is a process that allows a secure way to identify people for applications, based on strong specifications, but that fulfils the requirements of today's activities and technologies.

Before getting deeper in the definition of Federated Identity, it is very important to understand the background that led stakeholders of the informatics and business markets to initiate the creation of this process.

The following figure is a good summary of the evolutions of authentication processes with applications.

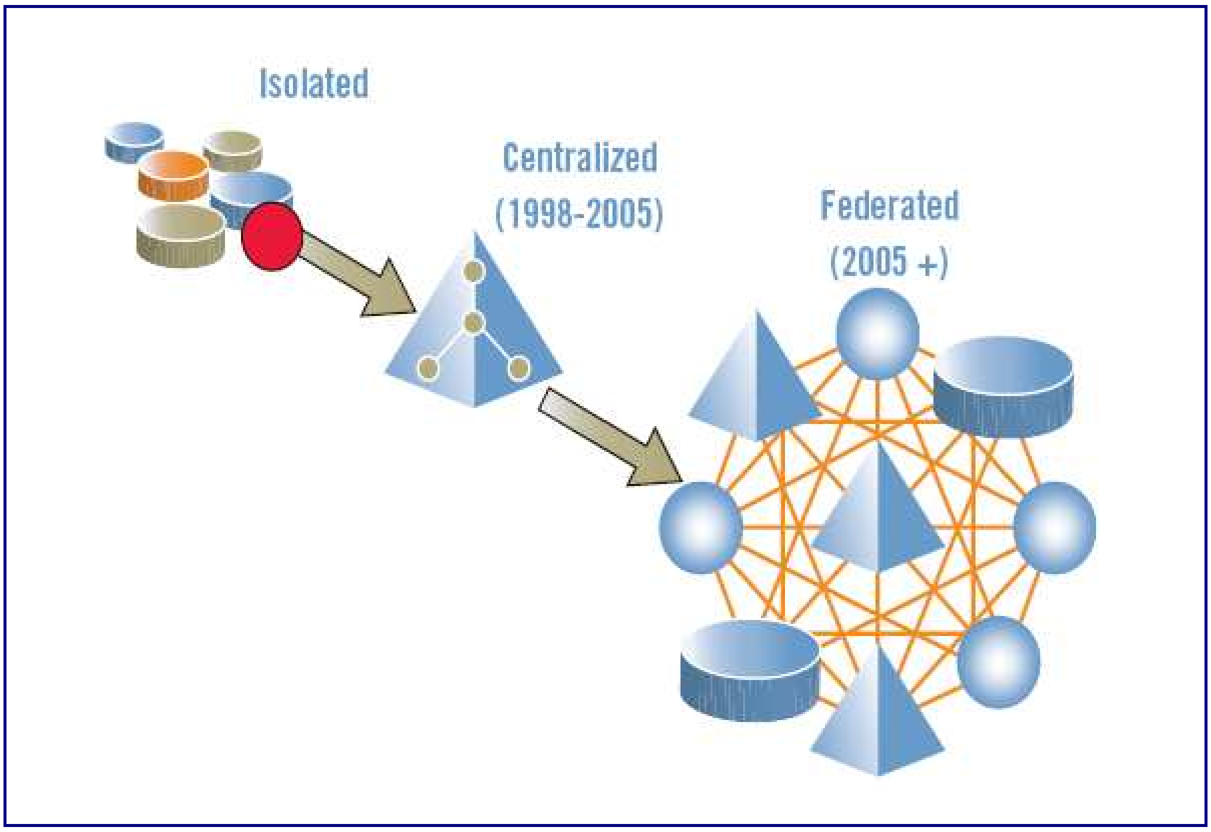


Figure 1: evolution of identification processes (©Ping Identity)

At first, authentication process could be thought in a way that each application manages its own identities; this is called 'isolated process'. In other words, the user should authenticate every time he (she) wants to use different applications. This process is a serious constraint for the user since he (she) need to deal with potentially several different authentication parameters, and required to be identified on each application anyway.
As it is unconceivable for applications to share identities together, the isolated process is barely not used. It is common to have a centralized process that allows the applications to share identities.

The **SSO** (Single Sign-On) is here the key concept to make this process efficient enough: when the user authenticates through an **IdM** application (Identity Management such as CAS), his identity parameters are saved by the IdM. When the user needs to log in another application, the IdM uses SSO and automatically authenticates this user on the application. The process of logging-out – **SLO** (Single Log-Out)- runs the other way wrong: the user is automatically logged out from each application he (she) were authenticated, on a single request from the user.

Nevertheless, the centralized process runs in a 'private' domain. The need nowadays is to allow SSO between partners which does not belong to the same domain, but which have a trust, an agreement between each other. This is the role of Federated Identity.
The domain is here called 'circle of trust'. The SSOs/SLOs are concerned only by the applications belonging to a same circle of trust.

We can highlight here what are the main issues of Federated Identities:

- the user should choose its security policy independently of its partners,
- the user should be able to control the identifications parameters which are sent to applications, and be known by different identities with different applications,
- the data crossing domains must be completely protected and reliable,
- the partners must be accurately authenticated for security reasons,
- the agreements between partners must be established in a way they can cooperate,
- the process must be adaptable according to each partner (which is partly covered by the use of Internet applications).

This is the reason why some specifications about Federated Identities do exist.
We are going to present them in the following part of this report.

Interactions between IdM, Federated Identity control entity and the applications are resumed by the following figure:

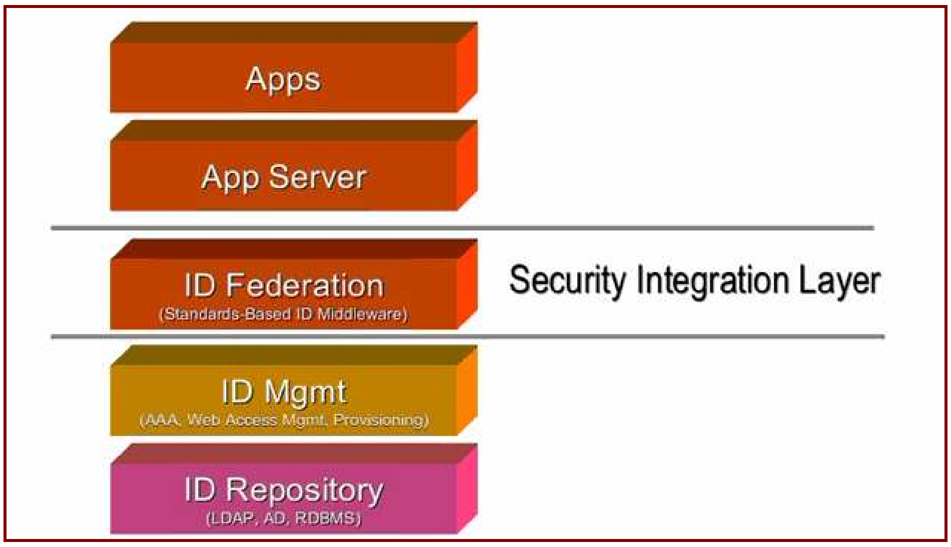

Apps
App Server
ID Federation
(Standards-Based ID Middleware)
Security Integration Layer
ID Mgmt
(AAA, Web Access Mgmt, Provisioning)
ID Repository
(LDAP, AD, RDBMS)

Figure 2: the layers of the Federated Identity process (©Ping Identity)

In Federated Identities, two key roles are set up: the **Service Provider**, and the **Identity Provider (SP and IdP)**. They are the entities that communicate together for Federation Identity. Therefore, they represent a link between the outside and the inside environment of a circle of Trust. Moreover, they need to be known from each partner of the federation.

The IdP send the identification attributes to other applications, ensuring that a good security is deployed when sending the data outside the circle of trust. It also manages the partnerships to allow SSO with accredited applications, and it keeps records of where the user is authenticated to allow SLO.
The SP 'consumes' identities attributes, and manages the connection of the users with all the applications concerned when a SSO request has been received from an IdP. Similarly, it manages the logging-out of the users from the same applications when a SLO request has been received from an IdP. The SP also needs to keep records of the partnerships to accept or not the SSO request, and in case of success, the SP should create a temporary local account, under the local settings managed by the user, with the applications.

The following figure describes a Federated Identity process of SSO, between two circles of trust (relevant to company A and B).

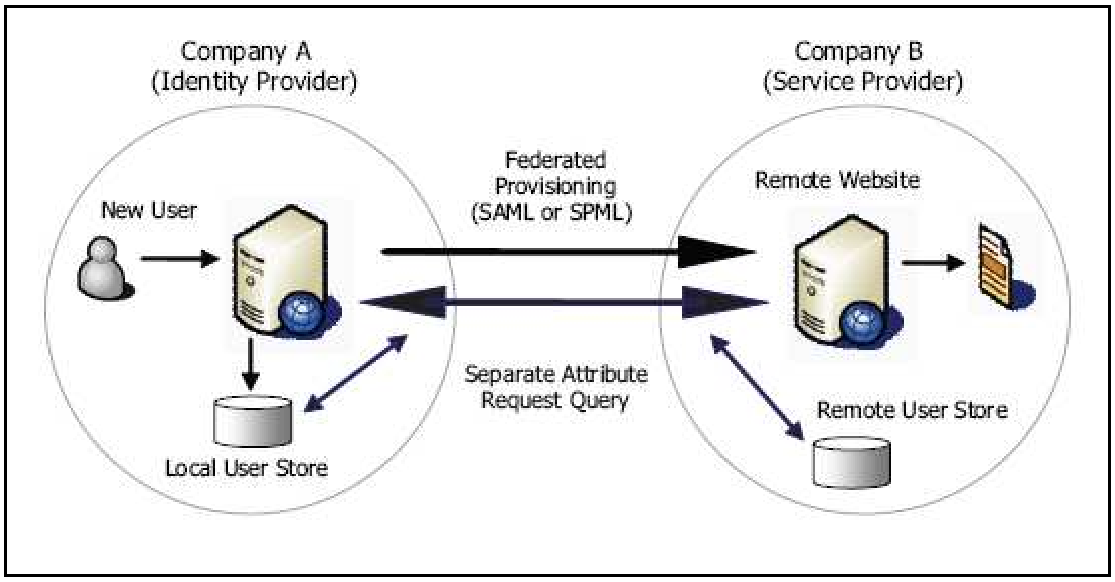


Figure 3: Federated Identity process between partners (©Ping Identity)

II. SAML

Security Assertion Markup Language (SAML) is an XML standard for exchanging authentication and authorization data between security domains, that is between an identity provider and a service provider. SAML is a product of the OASIS Security Services Technical Committee.

The single most important problem that SAML is trying to solve is the web single sign-on (SSO) problem. SSO solutions at the intranet level abound (using cookies, e.g.) but extending these solutions beyond the intranet has been problematic and has led to the proliferation of proprietary technologies that do not interoperate. SAML has become the definitive standard underlying many web SSO solutions in the identity management problem space.

SAML assumes the '**principal**' (often a user) has enrolled with at least one identity provider. This identity provider is expected to provide local authentication services to the principal. However, SAML does not specify the implementation of these local services; indeed, SAML does not care how local authentication services are implemented (although individual service providers most certainly will).

The most current version of the SAML standard is V2.0, which was approved on 15 March 2005. It is important to concentrate on SAML 1.1 since it is the definitive standard on which most other standards and implementations depend. Some characteristics of **SAML 1.1** follow:

A. SAML Assertions

SAML assertions are usually transferred from identity providers to service providers. Assertions contain statements that service providers use to make access control decisions. Three types of statements are provided by SAML:

1. **Authentication statements**
2. **Attribute statements**
3. **Authorization decision statements**

Authentication statements assert to the service provider that the principal did indeed authenticate with the identity provider at a particular time using a particular method of authentication. The three statement types are not mutually exclusive. For example, both authentication statements and attribute statements may be included in a single assertion (as shown above). This precludes the need to make subsequent round trips between the service provider and identity provider.

Example of SAML assertions:

```
<saml:Assertion
 xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
 MajorVersion="1" MinorVersion="1"
 AssertionID="..."
 Issuer="https://idp.org/saml/"
 IssueInstant="2002-06-19T17:05:37.795Z">
 <saml:Conditions
```

```
  NotBefore="2002-06-19T17:00:37.795Z"
  NotOnOrAfter="2002-06-19T17:10:37.795Z"/>
 <saml:AuthenticationStatement
  AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
  AuthenticationInstant="2002-06-19T17:05:17.706Z">
  <saml:Subject>
   <saml:NameIdentifier
    Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
    user@mail.idp.org
   </saml:NameIdentifier>
   <saml:SubjectConfirmation>
    <saml:ConfirmationMethod>
     urn:oasis:names:tc:SAML:1.0:cm:artifact
    </saml:ConfirmationMethod>
   </saml:SubjectConfirmation>
  </saml:Subject>
 </saml:AuthenticationStatement>
</saml:Assertion>
```

Figure 4: examples of SAML assertions

B. SAML Protocol

The SAML *protocol* is a simple request-response protocol. A SAML requester sends a SAML `Request` element to a responder:

```
<samlp:Request
 xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
 MajorVersion="1" MinorVersion="1"
 RequestID="..."
 IssueInstant="...">
 <!-- insert other SAML elements here -->
</samlp:Request>
```

Similarly, a SAML responder returns a SAML `Response` element to the requester:

```
<samlp:Response
 xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
 MajorVersion="1" MinorVersion="1"
 ResponseID="..."
 InResponseTo="..."
 IssueInstant="...">
 <!-- insert other SAML elements here, including assertions -->
</samlp:Response>
```

C. SAML Bindings

SAML 1.1 defines just one binding, the SAML SOAP binding. A compatible SAML 1.1 implementation must implement SAML over SOAP over HTTP (synchronous protocol). A SAML requester wraps a SAML Request element within the body of a SOAP message. Similarly, a SAML responder returns a SAML Response element within the body of a returned SOAP message. If there is an error, the responder returns a SOAP fault code instead.
Any SAML markup must be included in the SOAP body.
In SOAP 1.1, a SOAPAction HTTP header must be included with each HTTP request (although its value may be empty). A SAML requester may give the following value to the SOAPAction header:

```
SOAPAction: http://www.oasis-open.org/committees/security
```

D. SAML Profiles

In general, *profiles* describe the HTTP exchanges required to ultimately transfer assertions from an identity provider to a service provider. SAML 1.1 specifies two browser-based, single sign-on profiles:

1. **Browser/Artifact Profile** (subject confirmation method identifier of "urn:oasis:names:tc:SAML:1.0:cm:artifact")
2. **Browser/POST Profile** (subject confirmation method identifier of "urn:oasis:names:tc:SAML:1.0:cm:bearer")

These profiles support cross-domain single sign-on (SSO). Both SAML 1.1 profiles begin at the inter-site transfer service, which is managed by the identity provider. In practice, a client accessing a secured resource at a service provider will be redirected to the inter-site transfer service at the identity provider. After visiting the inter-site transfer service, the principal is transferred to the assertion consumer service at the service provider.
In the case of the Browser/Artifact Profile, a redirect is used; in the case of the Browser/POST Profile, the client issues a POST request (with or without user intervention).

To expedite processing by the assertion consumer service, two separate URLs are specified:

1. Artifact Receiver URL (Browser/Artifact Profile)
2. Assertion Consumer URL (Browser/POST Profile)

These and other URLs may be recorded in a SAML metadata archive.

III. Our project

Even if the Federated Identity process is still emerging in the real-context business activities, we can find many solutions to set the process between entities.

In our case, INT wanted to perform tests on security and privacy issues on SAML messages or whatever protocol used to communicate authentication data (such as attributes, account information…) between the partners.

INT is working on this topic with ITAM, in Mexico. They agreed on performing tests thanks to a platform called Ping Federate, which is edited by Ping Identity Corp, one of the leaders on the market of the open-source solutions in Federated Identity platforms.

Therefore, both INT and ITAM would have had their own Ping Federate server, set up to communicate between each other.

Our project consisted in installing the Ping Federate server in INT, setting it up to run, and realising a test with ITAM, where the same work was made. In our project, we have also planned to develop our own service applications, in order to be able to set the server up with personal applications that would not be described in the manual.

In the following parts, we are going to detail how Ping Federate works, and what was our case study for the project.

A. How does Ping Federate work?

As seen above, dealing with Federated Identity process asks for considering two types of services: IdP and SP. As a consequence, Ping Federate server can be used as an IdP, a SP or both of them.

The IdM would be the centralized identification tool, called here Identity Management. We must clearly understand that this IdM is an *inside* tool: therefore, it allows local identification, whether Ping Federate has been installed or not.

When Ping Federate has been installed, we can say that the applications can be either accessed from local identification for local users – through IdM as we said, or they can be accessed by local and outside used – through Ping Federate and the Federated Identity process.

We are now detailing the steps that lead to a communication between the Ping Federate server (as an IdP and as a SP) and the applications/IdMs of an entity.

B. Federated identity communication scheme in Ping Federate

As the following figure shows, Ping Federate could communicate with any kind of applications/IdMs. To permit adaptability, the Ping Federate server has to pass through Integration Toolkit (that adapts with the application at the other end).

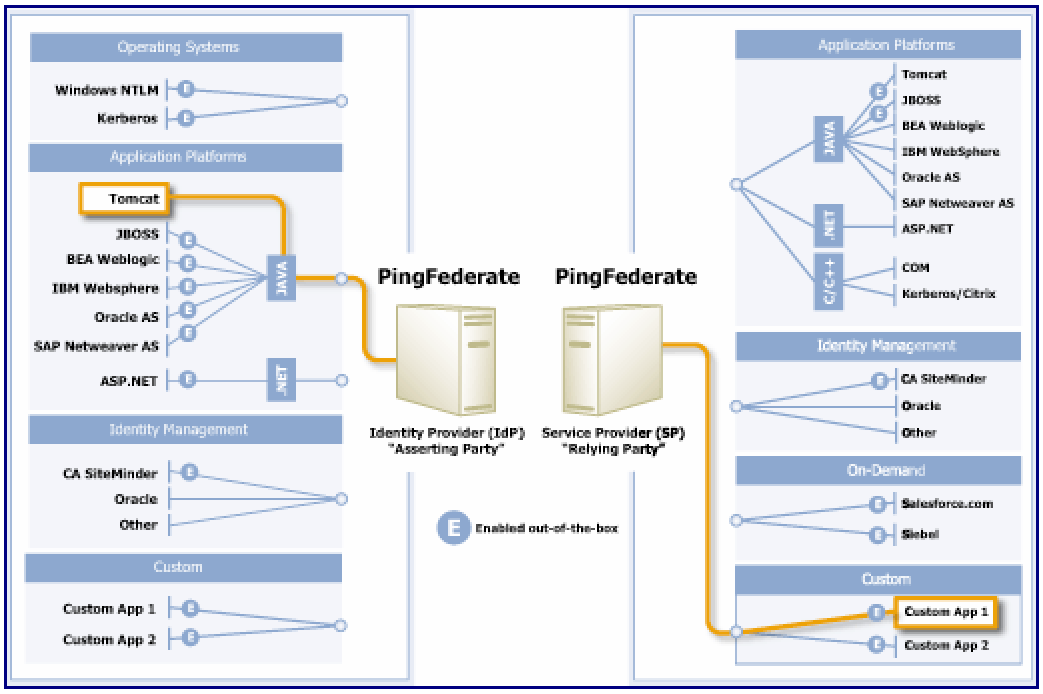


Figure 5: Integration Toolkit in Ping Federate (©Ping Identity)

Ping Federate distinguishes two parts in the Integration Toolkit: the **adapter**, which interacts with the server itself, and the **agent toolkit**, which interacts with the application or the IdM.

The following figure illustrates the roles of the agent toolkit and the adapter.

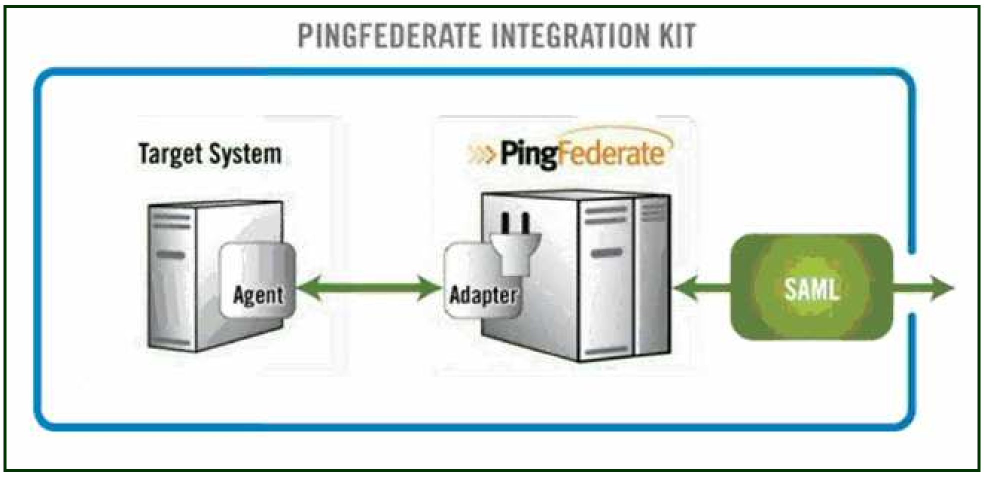


Figure 6: Agent and Adapter roles in Ping Federate

In the case of an IdP, the Integration Toolkit retrieves the attributes in a local data store to allow authentication on the (remote) application.
In the case of a SP, the Integration Toolkit passes the information to the application, validating the authentication that could have probably been required if the application was used locally. When the process succeeds, the user is authenticated on the application's partner, even if he or she was not a local account linked to this application before.

The protocol of communication between the agent toolkit and the adapter is a proprietary secure format called PFTOKEN. The PFTOKEN contains all the information that is requested for the Ping Federate server or the application/IdM to enable the Federated Identity. The PFTOKEN is encrypted for security purpose, but the server has a tool to visualise the PFTOKEN .

Therefore, when the server plays the role of an IdP, the agent must retrieve the attribute and place them into the PFTOKEN, accordingly with the PFTOKEN format. When the server plays the role of a SP, the agent parses the PFTOKEN to pass the attributes accordingly with the application's specifications.

To allow the encryption of the PFToken, Ping Ferderate uses a PFToken agent that specifies how the PFTOKEN is encrypted, where it comes from…

An example auto-generated by the server (upon the settings we made) is given here:

```
#PFToken agent properties
#Sat Feb 03 15:14:19 CET 2007
encryption.algorithm=AES
encryption.padding=PKCS5Padding
encryption.mode=CBC
domain=
cookie.path=/
encryption.password=H7Ujx5qnKVgieuuzSIBLig\=\=
cookie.maxage=300
encryption.iteration=1000
encryption.keysize=128
container.name=IdPJava
delete.cookie=true
transfer.method=Query parameter
```

Figure 7: PFToken properties

C. The cases allowed by Ping federate

Mainly two cases are significant in the Federation Identity process used with Ping Federate: the IdP-initiated SSO/SLO and the SP-initiated SSO/SLO.

The IdP-initiated SSO/SLO allows the user to perform a SSO/SLO request from a local IdM, hence from the IdP server, and be authenticated/unauthenticated to all the partner's applications that are potentially reachable for this user.

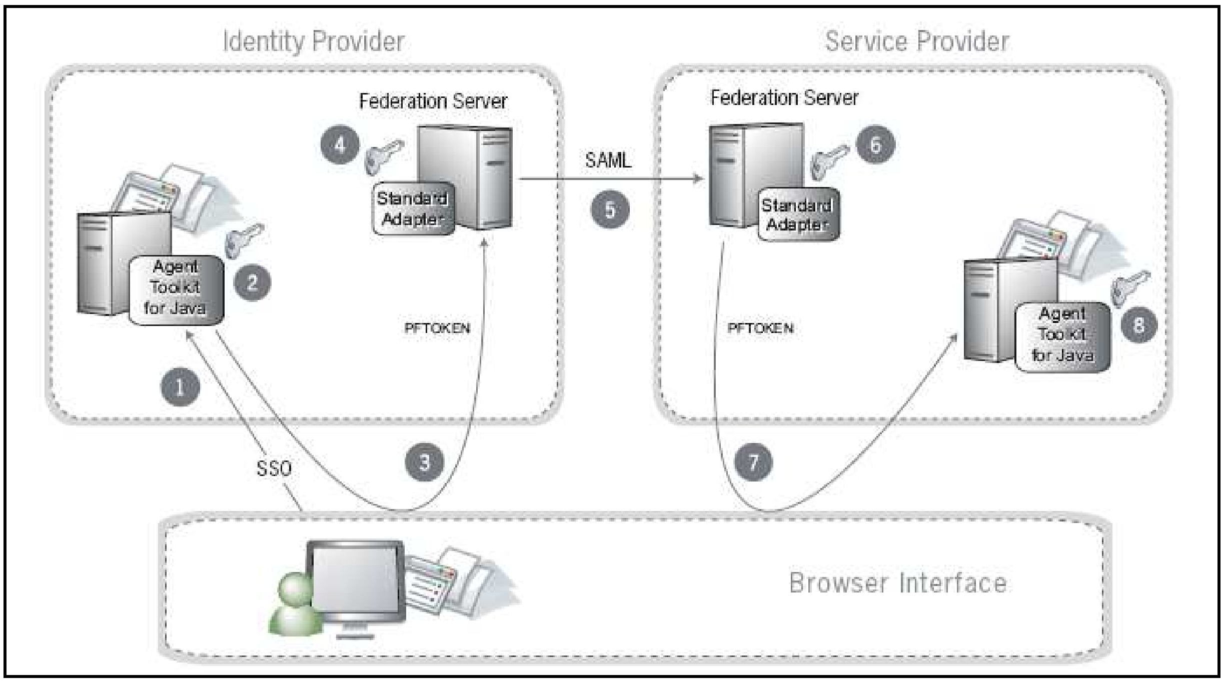


Figure 8: IdP-initiated SSO with the standard adapter

The SP-initiated SSO/SLO allows the user to retrieve the attributes from the IdP he or she belongs to, directly from the application situated the partner's circle of trust. This is only possible when the partners agreed on that scenario since to retrieve the data, the user must be 'linked' to his or her IdP.

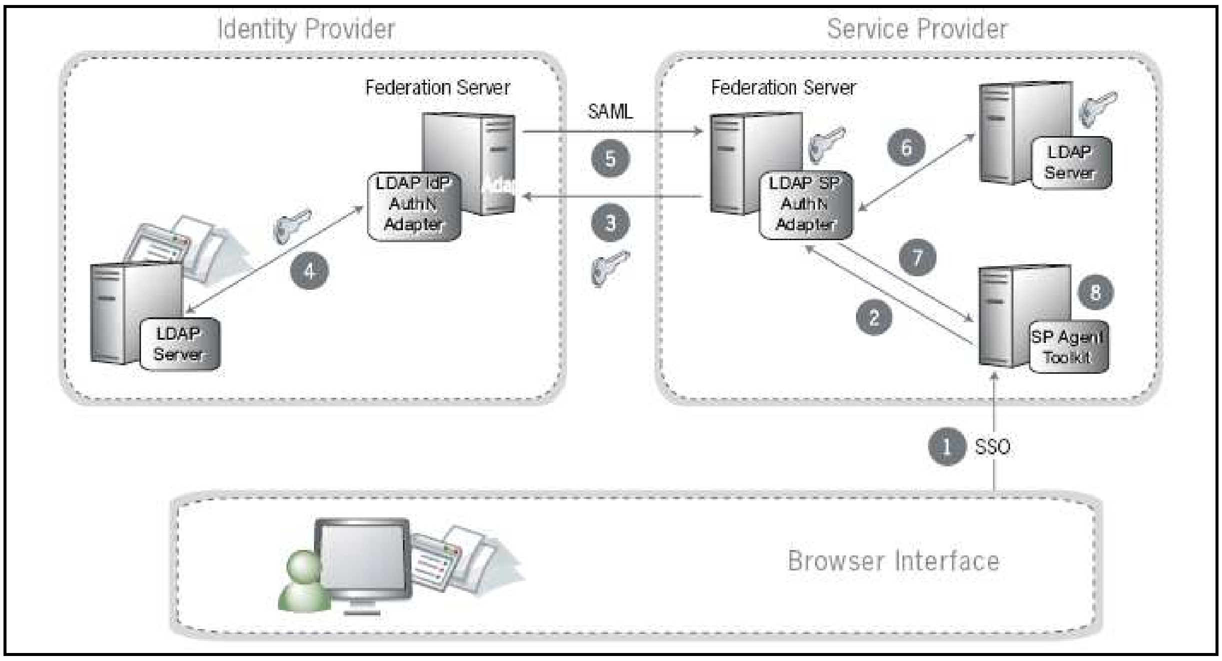


Figure 9: SP-initiated SSO with LDAP adapter

IV. WHAT REMAINS TO DO

The installation details are explained in the “installation/configuration” user guide.

Since we have installed an IdP LDAP adapter, we also wanted to install the SP LDAP adapter and run the authentication on our own test application (INTest), instead of the application samples provided by Ping Federate.

As explained in this report, we have installed an LDAP-based authentication on our test service, which is perfectly running locally.

Nevertheless, if this application is being used with Ping Federate, the authentication process is a bit different, in the sense that if someone logs on through SSO Federated Identity process, the login popup **must not appear again**, the person must be authenticated and be on the main page of the service.

We have started preparing the connection between Ping Federate and this test application downloading a Ping Identity Integration Toolkit for Java (http://www.pingidentity.com/products/java/download).

This toolkit provides the necessary jar library (**pf4-pftoken-agent-1.1.jar**) to place in the **/lib** repertory of the application (for instance, **~/apache-tomcat-5.5.20/webapps/INTest/lib** in our case). This library can be found in the **/lib** folder of the Integration Toolkit.

The role of the library is to give the user the necessary classes to work with in order to parse or create the PTOKEN that will be sent by the agent.

Since you need to encrypt and “tailor” the PFTOKEN according to your Ping Federate server, one file need to be create in the **/config** repertory of the application.
This file, **pfagent-sp.properties** contains the auto generated properties that are generated on the server, under the adapter configuration page (which would have been the **SP adapter** page in our case).

We planned to analyse how we could use both Tomcat (for the authentication process) and Ping Federation (for the creating or retrieving of the PFTOKEN process) classes and create the jsp pages that would have made the service working.

In a first step, we would have allowed IdP-initiated SSO/SLO only, and later we would have changed the test application to allow the SP-initiated SSO/SLO (by adding components in the application, such as a link or a button…).

This has not been made for lack of time, but we have understood the main concepts of the classes’ functionality, that would have made use certainly capable of finalizing the INTest test application.

For more information to reach this objective, here some very useful links that could help:

- http://tomcat.apache.org/tomcat-5.5-doc/catalina/docs/api/index.html (Java doc for Tomcat)
- **Index.html** in the **docs/api/** repertory of the Integration Toolkit for Java folder (Java doc for the PFTOKEN Agent)
- **Java_Integration_Kit_User_Guide.pdf** in the /docs repertory of the Integration Toolkit for Java folder (for more information about the installation of the Integration Toolkit)

V. BIBLIOGRAPHY

- **FederatedIdentityPrimer**, White Paper September 2006, PingIdentity
- **Animated SAML 2.0 Tutorial**, Quicktime movie, PingIdentity
- **PingFederate_with_WS-Federation_Webinar**, movie, PingIdentity
- **PingFederate_Admin_Manual**, PingIdentity
- **Quick_Start_Guide**, PingIdentity
- **Tutorialv2**, Liberty Specs Tutorial, ProjectLiberty
- " **Federated Identity Architectures** ", U. Fragoso-Rodriguez, M. Laurent-Maknavicius, J. Incera-Dieguez, 1st Mexican Conference on Informatics Security 2006 MCIS’2006, International Conference approved by IEEE Computer Society, Oaxaca, Mexico, November 2006.
- http://fr.wikipedia.org/wiki/SAML
- http://fr.wikipedia.org/wiki/Liberty_Alliance
- http://fr.wikipedia.org/wiki/Authentification_unique