Federation Proxy for Cross Domain Identity Federation

Makoto Hatakeyama
NEC Corporation, Common Platform Software Res. Lab.
1753, Shimonumabe, Nakahara-Ku, Kawasaki,
Kanagawa 211-8666, Japan
+81-44-431-7663
m-hatake@ax.jp.nec.com

ABSTRACT
Identity federation is a key factor for a user to access multiple service providers seamlessly. Many protocols that federate between the providers have been proposed. Such protocols are operated under the same rules of identity management between the federated providers. Moreover, cross-protocol federation that uses different protocols for federation between the providers has also been proposed. However, the providers cannot federate all user information even though they do so using the cross-protocol federation because they do not confirm whether the federated providers obey the same rules or not. A federation bridge that converts and regulates the interaction messages between the providers is described herein. With it, the provider can federate based on the rules that a user and the providers determine.

Categories and Subject Descriptors
H.3.5 [Information Storage and Retrieval]: On-line Information Services – Data sharing. Web-based services.

General Terms
Management, Security.

Keywords
Identity Federation, Attribute Exchange, Privacy.

1. INTRODUCTION
Identity federation is a key factor for a user to access multiple service providers conveniently. The federation enables the user to use these providers seamlessly without multiple authentication, when the providers federate with each other and exchange his or her information on authentication and authorization. Such information exchange reduces unnecessary operations before he or she uses them. Providers can use multiple protocols such as SAML [17] and OpenID [21] to federate identity management systems. By using these protocols, providers can offer users seamless access between their services. Moreover, federation between providers that use different protocols has also been proposed [15] because the providers are operated independently and they may use different protocols for federation. With this cross-protocol federation, they can share user’s attributes, even though one service uses only one protocol and the others use another.

However, the providers cannot exchange all users’ attributes though they use the cross-protocol federation because they do not check whether they can share the attributes or not based on the rules to manage them. Such federation is related to user privacy, so it must obey the OECD privacy guidelines [18]. These guidelines require that the providers manage attributes based on the rules described by a user and the providers. The providers manage user’s attributes under some rules through an agreement based on the conditions of attribute management. In OpenID, for example, providers federate based on the user consent. Without this consent, they cannot exchange user information. In SAML federation, in addition to the user consent, providers are required to agree on the same contract beforehand. This relationship based on the agreement is called a “Circle of Trust (CoT).” Even if they use different protocols for federation, they obey the same rules between them. To federate between providers that use different protocols they have to use cross-protocol federation and to judge based on the agreement of how they can manage an attribute exchange.

Several studies focused on management of identity federation to achieve federation between different providers. Several systems for access control of attributes [1, 2, 3, 6, 11, 13 and 14] have been proposed. For example, Huang et al. proposed a trust relationship between providers [1]. When a provider interacts with an unknown one, it can judge whether the unknown provider is trustworthy or not by checking the relationship between the unknown one and the other trustworthy entity. Although the provider can verify the others for federation, the decision of attribute management between different systems is out of the scope of this study. Wang and Vassileva [3] also proposed the trust and reputation values for evaluating other providers. A provider can check whether a request from another provider is acceptable or not using the values. This method can check whether it can federate or not, but the providers do not understand the ways attributes are managed. Blaze et al. proposed trust management for SOA systems to federate with each other [2]. The trust management system controls access between different services in the SOA system using trust policies. However, they did not mention federation between multiple SOA systems that do not use the same rules for management. Gomi et al. [11] proposed a delegation framework to access services for attributes. It enables a provider to access the others instead of a user using a delegation assertion. In addition to needing this framework, identity federation based on the rule confirmation needs to be covered. Other systems use role-based access control with privacy policies [13 and 14]. This system
controls access using a user role and privacy policies, though it does not cover federation between multiple protocols. Trust Negotiation protocols that address the trust relationship between distanced providers have also been proposed [8, 9 and 10]. In these protocols, a provider determines whether it can interact with another provider by disclosing its credentials each other. They can check the conditions for initiation of attribute exchange, but they need a trust relationship between a user and them. Several attribute management models that handle attribute exchange in identity federation [4, 5, 7, 16, and 19] have been proposed. Myers et al. proposed an expressive model of information flow [7]. This model can express all attribute exchanges. It aims to control the information flow between a user and a provider, but the provider does not judge the exchange based on the rules described by all entities in the exchange. Policies that consist of purpose, recipient, retention, data, access, disputes, and remedies have been specified [16 and 19]. A user and a provider describe these policies, and the website can determine the rules that it must obey for attribute management using policy matching rules. However, the site does not address interaction between providers. Backes et al. also proposed a privacy policy comparison for attribute exchange [4]. This enables providers to check the policy to be complied. However, when exchanging attributes between them, they cannot decide how to federate with the others. Framework for privacy policy management between providers (Liberty Identity Governance Framework (IGF)) has been proposed [26]. By using this framework, providers can describe their privacy policies and check the rules for identity federation. However, IGF did not address the federation between different protocols. Rostad and Nytro [5] proposed access control decided by a user. The user describes policies for role-based access control; hence, he or she can control access to attributes. However, the provider does not exchange them when it interacts with the others under the agreed rules.

This paper introduces a federation bridge to proxy an interaction for attribute exchange between providers that use different protocols for federation under the same rules. The bridge regulates attributes to be exchanged so that the system obeys federation rules between the providers as well as converting protocols, though they use different ones. This paper specifies functions necessary for the federation bridge, including protocol conversion and message regulation, that enable the providers to federate different systems in an interoperable way. As described in OECD guidelines, when providers share the attributes with others, they obey the management rules on which all federated providers and a user agree. The bridge is also required to interact with other providers under the management rules on which it, the providers, and a user agree. Therefore, it regulates the attributes to be exchanged between them based on the rules and regulates the interaction between providers and the bridge. Three models of deployment for the federation bridge are presented in this paper because the best pattern of the deployment will vary depending on the use cases. The bridge federates multiple providers, so the relationship between the bridge and providers are different. The models are also different according to the relationship. This paper also discusses the interoperability of the federation bridge. The bridge needs interoperability because it proxies multiple providers. Though the federation bridge can manage the rules, it cannot interact with them without an interoperable manner, so it must interact in interoperable ways with the other providers.

The rest of this paper is organized as follows: Section 2 describes the identity federation models. Section 3 describes the federation bridge. Section 4 describes deployment models of the federation bridge. Section 5 discusses interoperability and issues related to it. Finally, a brief summary is given in Section 6.

2. IDENTIFY FEDERATION MODELS

Identity federation is necessary in many domains for user convenience because providers can share user information by the federation without his or her additional operation. It can also reduce the work providers have to do for attribute maintenance because they can get attributes from other entities though they do not maintain them. This section shows the examples of identity federation scenarios, especially cross-protocol federation, and current models of identity management. After that, the requirements for the federation are described.

2.1 Example of Multiple Identity Federation

2.1.1 Federation for User Registration

The first case is user registration at a service provider using identity federation. Figure 1 shows the flow of the registration.

In this case, a user registers attributes to make his or her account on a shopping portal site. Usually, when the site obtains these attributes, the user inputs them into the site. Though other providers have obtained the attributes, they do not share them. The user has to input the same attributes numerous times at the providers, so the load of user’s operation is heavy. Thus, it is convenient for providers to share the user’s attributes that they have already obtained without his or her having to input them again. In this scenario, the shopping portal site receives them from the ISP.

For the shopping portal site to get the attributes from the ISP, it requests the necessary attributes for registration from the authentication service. This request message is sent in one protocol such as OpenID. When the authentication service receives the message, it deals with the requests, i.e., it confirms the user information. After that, if the authentication service obtains all the attributes that the site requests, it returns them. However, when it does not have all necessary attributes, it requests them from the attribute service that has the necessary information. The protocol between them may be different from that between the shopping portal site and the authentication service. The authentication service converts the request format and forwards it using the converted protocol. When the attribute service receives the request from the authentication service, it returns the requested attributes. After the authentication service receives the response, the service communicates it to the initial request from the portal site. The protocol of
the response is the same as that of the request. Next, the site receives the attributes and makes a user account automatically.

During the interaction between the authentication service and the attribute service, they have to correlate user IDs that they manage for themselves in addition to message conversion. If they use a unique ID to identify one user, they do not have to correlate IDs. However, they may use different IDs to identify him or her, because they manage users independently. To correlate different IDs, SAML specifies the name identifier management protocol and profiles, which can connect different IDs by a federation ID [17]. In the name identifier management protocol, each service manages local IDs that the service issues for itself and connects them with the federation IDs that it and the federating service agree to use for federation. The service uses the federation IDs for communication with other services and uses the local IDs for the user management in itself. The above idea is adaptable to protocol translation. If the services contain both the local IDs and the federation IDs and they use the federation IDs for communication, they can correlate their user IDs with those of the other services. In the situation depicted in Figure 1, for example, if the authentication service manages local IDs, federation IDs for the shopping portal site and federation ID for the attribute service, it can convert IDs when it translates the message. Therefore they can federate even if they use different federation protocols.

2.1.2 Federation for Attribute Exchange

The next case is where identity providers manage all user information instead of other service providers. For the service, the operation cost to manage the user information is high. Thus, the service does not have user accounts. Instead, other providers manage user authentication and payment for content. Sharing the information is also convenient for users because they do not have to input the information several times.

![Figure 2. Attribute exchange.](image)

In this situation, depicted in Figure 2, a content provider (CP) site offers video content to a user. Nevertheless, it does not have user accounts, though it needs authentication and authorization for the content charge. Therefore, it gets the information from the others.

When a user accesses content at the CP, the CP requests ISP1 to authenticate the user, authorize him or her, and make the payment for the content instead of asking to the user to do it manually. It sends a request message using one protocol that the CP and ISP1 agree on for the interaction. When ISP1 receives the request message, it deals with the request. If it can authenticate the user, authorize the access, and charge a fee for the content, it returns the result to the requester. However, if it cannot process all tasks, it sends a request to ISP2. For example, ISP1 can authenticate the user, but it cannot charge a fee instead. Thus, ISP1 sends a request to ISP2. In this interaction, ISP1 may use a different protocol from that between the CP and ISP1. ISP1 converts the format of the request message. It also converts the user ID because the providers may use different IDs for user identification. The ID federation is the same as described at section 2.1.1. When ISP2 gets the request message from ISP1, it deals with the request and returns a response. The response is sent by the same protocol as the request. When ISP1 receives the response, it makes another response to the CP and returns it. This interaction protocol is also the same as the initial request from the CP to ISP1. Finally, the CP receives the response and offers content to the user.

2.2 Current Identity Federation Models

For these use cases, providers must federate the user identity, though they manage it independently. When they federate the identity, they consider both information federation and system federation because they are divided into some groups.

Information federation means grouping of user profiles such as an employee profile or a private profile. The profiles are divided into silos. The identity profile silos are shown in Figure 3. A user may have different kinds of profiles. For instance, a user accesses enterprise systems using his or her employee profile, and he or she logs into a blog site using a private profile. He or she changes the profiles according to the sites accessed. The enterprise system, for example, requires only employee profiles for its business, and the shopping portal site uses private profiles. Each profile contains different set of attributes because the necessary attributes for each profile are different. It is possible that the profiles contain the same attributes but that the value of them is different. For example, an enterprise profile and a private profile contain e-mail addresses but the addresses in the two profiles are different because a user may separate the addresses for work and private. It is also possible that they may contain the same attributes of which the value is also the same such as the legal name or age. In this case, they are independently managed at each profile. Some studies have focused on such profile management [12].

![Figure 3. Identity profiles.](image)

System federation connects multiple providers for account linkage. The providers contain user information in their account, so account linkage is necessary for the information to be exchanged. The model of system federation for account linkage is shown in Figure 4. The accounts are independently managed by many providers; hence, one user has many accounts. For example, one provider creates a user account based on the assured attributes by a trusted third party, whereas another provider creates an account based on the users’ claim. These accounts have different means of authentication, such as biometric ID, a smart card or ID/PW authentication, and attribute management.

To federate the accounts, several protocols have been proposed such as SAML [17], OpenID [21], ID-WSF [24], or WS-* [23 and...
These protocols specify the federation methods, security mechanism, and attribute exchange. By using these protocols, providers can link user accounts. Such identity federation is based on the trust relationship between providers. The relationship differs between providers if the providers use different federation protocols. In SAML model, for example, providers federate based on a business contract or agreement. OpenID does not have to premise the preconfigured relationship between providers; instead a user selects the providers to share his or her information and they interact based on the user consent.

A service can obtain user information from its account using the federation. If a service needs information in multiple accounts through multiple protocols, the protocols must be bridged. Federation between different protocols has been proposed [15] because the providers are operated independently and they may use different protocols for federation. [27] also describes the identity assurance for federation between different protocols.

Figure 4. Multiple system federation layers.

2.3 Requirements for Bridging Protocol

Providers are required to manage attributes based on the rules described by a user and the providers to accomplish the cases in section 2.1 for the privacy protection specified at the OECD guidelines [18]. Though the providers use the cross-protocol federation [15], they have to manage attributes under the same rules on which all federated providers and a user agree. Therefore, federation between providers must satisfy the following two requirements for connecting providers:

One is an attribute exchange filter. Though providers can convert the message format from one protocol to another, they must check the rules that they obey for attribute management. They may have their own rules or policies to manage the attributes. If the attribute exchange does not satisfy the rules that the attribute sender and receiver decide, they cannot share them. Moreover they have to satisfy the requirements of a user for privacy protection, when they exchange the attributes. In order to obey the attribute management rules that satisfy the requirements of the providers and the user, they have to regulate the attribute exchange if the exchange does not satisfy the rules.

The other is interoperability for attribute exchange between providers because there are many federation systems. As described in [28], when providers federate with others, they have to exchange attributes in the interoperable manner. They cannot interact with others without such interoperable ways. Therefore, interoperability is needed for federation.

In this paper, we suppose that providers have a trust relationship beforehand and that they possess the rules of the other provider for managing the attributes. We also suppose that a user has set his or her requirements about attribute management at providers that manage the attributes. Under these conditions, when providers exchange attributes, they can check user requirements in addition to the preconfigured trust relationship between them. The exchange depends on the user requirements and cannot be determined only by the preconfigured trust relationship. In OpenID, for example, a user decides the federated providers when the attributes are exchanged. Even if they have the relationship beforehand, they cannot judge whether they can share user’s attributes only by the relationship. The attribute exchange needs to satisfy both the rules between providers and the user requirements. Therefore, in this paper, providers check all the necessary rules and requirements whenever they exchange the attributes though they have the preconfigured relationship.

3. FEDERATION BRIDGE ARCHITECTURE

To satisfy the requirements in section 2.3, a federation bridge for a proxy that exchanges attributes between federated providers using cross-protocol federation is proposed in this section. The federation bridge that converts protocols and regulates attribute sharing, thereby enabling federation between different protocols, is described in this section. The bridge operates these two when it receives a request or response from providers. Hence, it processes different functions in the request and response.

A federation bridge forwards messages between providers. In the federation, there are an identity provider (IdP) and a service provider (SP). The IdP manages user information on authentication or attributes. The IdP offers the information to another IdP or SP. The SP does not maintain such information, but it receives the information from the other IdP. The federation bridge connects different IdPs that use different protocols, and it converts messages and protocols between the IdPs.

This section describes the functions of the federation bridge that proxies the messages and the sequence of federation.

3.1 Functions in a Federation Bridge

A federation bridge has three functions to connect providers that are protocol conversion, attribute filtering in the request message, and validation of exchanging attributes in the response message.

A federation bridge connects multiple protocols between IdPs because the bridge federates multiple systems that use different protocols. The situation of federation is the same as shown in section 2.1.1 or 2.1.2. When one IdP receives a request from a SP, it forwards the request to another IdP using different protocols. The conversion and regulation of the request are needed between the IdPs. Therefore, the bridge must be located between the IdPs.

The federation bridge needs to filter the requested attributes. Because inappropriate exchange of attributes between providers may cause privacy leakage, the bridge limits the attributes based on the rules. IdPs may use different rules for attribute management. For
example, when the IdP receives a request from the SP and forwards it to the destination IdP, the intermediate IdP cannot always return all the attributes that are sent from the destination IdP because the rules of the intermediate IdP prohibit sending the attributes to the SP. When the rules are different among the destination, intermediary, and original requester, the exchanging attributes between the two of them also differ. Therefore, the bridge must limit sending attributes based on the rules. The bridge does not request all the attributes that are sent from the SP to the IdPs, but it limits the kinds of attributes in a request message and forwards the request to the destination. The providers exchange only the attributes that all of them agree to exchange.

The federation bridge validates a response message from the destination IdP in the response process. Because providers must exchange the attributes based on the rules, the bridge checks that the response message satisfies the rules.

The details of the functions and components in the request and response process are following:

### 3.1.1 Bridge Components in the Request Process

When a federation bridge forwards a request between IdPs, it validates the received message, filters the requested attributes, and converts a protocol. The components in the request process are shown in Figure 5.

![Figure 5. Request flow.](image)

The message verification part in the bridge checks the sender and receiver of the request, the requested attribute, and the protocol. By using the checked information, the message verification part selects the rules of protocol conversion and filtering of attributes. The bridge determines whether it converts the message and filters the requested attributes by the rules. The rules of protocol conversion are described by the sender and receiver (IdP1 and IdP2 in Figure 5). The bridge must follow the rules when it transfers a request message. The filtering rules are also defined by IdP1 and IdP2. For the bridge to intermediate such federation, it must contain the rules for conversion and filtering.

The request filter limits the attributes in the request message from IdP1 to IdP2 based on the filtering rules. When IdP1 requests the necessary attributes to IdP2 instead of SP, IdP1 does not always receive them because a user or IdP2 does not allow exchanging them. To exchange the attribute under the rules defined by a user and IdPs, the bridge checks that the SP’s requirements satisfy the rules of the user, IdP1 and IdP2. The requirements and rules include the kinds of requesting attributes or the conditions of attribute management. As the format for expressing the rules and requirements, Liberty IGF specification [26] or P3P [19] can be adaptable. The attribute filtering between IdP1 and IdP2 is defined as follows:

If IdP1 requests the attributes $Att_{SP}$ that SP has requested to IdP1 and that satisfies the following $pol_{SP}$ to the bridge,

$$Att_{SP} = \{ a | \text{set of all attributes in a request message from SP satisfying } pol_{SP} \}$$

$$pol_{SP} = \{ x | \text{set of SP’s requirements of attribute exchange in a request message} \}$$

then the bridge determines a requesting attributes $Att_{req}$ that satisfies the following $pol_{bridge}$:

$$Att_{req} = \{ a | \text{set of requesting attributes that satisfy } pol_{bridge} \}$$

$$pol_{bridge} = pol_{SP} \cap pol_{fed} \cap pref_U$$

$$pref_U = \{ p \mid \text{set of rules that user permits to use attributes} \}$$

$$pol_{IdP1} = \{ y \mid \text{set of rules that IdP1 follows when it handles attributes} \}$$

$$pol_{IdP2} = \{ z \mid \text{set of rules that IdP2 follows when it handles attributes} \}$$

The bridge requests the attributes that satisfy the requirements and rules of the user, IdP1, IdP2, and the SP.

The other is protocol conversion, which changes the protocol and format of the request messages to that of the other. The changed protocol is determined by the agreement between the bridge and IdP2. If a message from IdP1 uses the agreed protocol, it does not convert the message.

When the request from IdP1 to IdP2 does not satisfy the rules and requirements of IdP1, IdP2, and a user, the federation bridge does not request any attributes to IdP2 but returns the error message to IdP1. This means that the SP’s requirements do not satisfy the conditions for attribute exchange described by IdP1, IdP2 and the user. In order for the SP to get the necessary attributes, IdP1 may request them to another IdP again. Because another IdP will have another rules for attribute exchange, the result of rule confirmation may be different from the previous result. The bridge checks whether IdP1 can get the attributes from another IdP. IdP1 may also repeat the request process until it gets them. This repeating process by IdP1 is beyond the scope of the process of the federation bridge. The discovery of another IdP is also out of the scope of the bridge. (ID-WSF Discovery Service [24] or OpenID discovery mechanism [21] can be applied to the discovery.) If the IdP1 cannot receive the attributes from any IdP, it returns the error. This means that the SP’s requirements do not satisfy any conditions that all IdPs and the user describe. In this case, the SP requests the user to handle the attributes. This process requires the user to be involved in the attribute exchange but it is needed in order for the SP to offer services to the user.

An example of attribute filtering and conversion is the case shown in Figure 1. For that case, a SP (a shopping portal site) requests user’s authentication information and attributes (e.g., address, name, and age) to an intermediate IdP (an authentication service that is the same as IdP1 in Figure 5). If IdP1 can authenticate the user but does not have the attributes, the request is transferred to a federation bridge. The bridge limits the requested attributes for this step using filtering rules. For instance, it requests only limited attributes (e.g., address...
and name) from the attribute service (the same entity as IdP2 in Figure 5). The two services may describe the rules. After the limitation, the bridge converts the request to another format that the bridge and the attribute service accept.

3.1.2 Bridge Components in the Response Process
The federation bridge receives a response message from IdP2 and forwards it to IdP1, as shown in Figure 6. When the bridge receives a response, it verifies the response, converts the response message, and verifies the attributes in the converted message.

Conversion Rules
Conversion Rules
Conversion
Rules
Conversion
Rules
Conversion Rules
Conversion Rules
Conversion
Message Verification
Conversion
Rules
Conversion
Rules
Conversion
Rules
Conversion
Rules
Conversion
Conversion Rules
Conversion
Rules
Conversion
Rules
Conversion
Rules
Conversion
Conversion Rules
Conversion
Conversion
Conversion
Conversion Rules
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conversion
Conver
request message is sent by an agreed protocol with the SP and IdP1. Figure 7 shows that the OpenID Request is sent to IdP1.

3. When IdP1 receives the request, it reacts to the request. For example, it authenticates the user and retrieves the attributes. If it can process all of the SP requests, it returns the response.

4. If it cannot return a response to the SP in the previous step, it sends a new request message in which the necessary information is described. The protocol and message format in the request depend on IdP1’s requirements.

5. When IdP1 sends a request to IdP2, a federation bridge catches and checks the message. The bridge converts the protocol by checking the sender and recipient of the message, unless the protocol is satisfied with both of the interacting entities. In the sequence, the OpenID request is translated to the SAML assertion. During this process, the bridge limits the requested attributes if needed. This limitation is based on the rules on an attribute exchange determined by IdP1 and IdP2.

6. The bridge sends the converted request to the IdP2.

7. When IdP2 receives the request from the federation bridge, it responds to the message. For example, it retrieves the requested attributes.

8. IdP2 returns the response that includes user information to IdP1. In the scenario, IdP2 sends the SAML assertion.

9. During the response interaction from IdP2 to IdP1, the federation bridge catches the response and checks its protocol. When the protocol is not satisfied with rules between the bridge and IdP1, it changes the protocol. It also verifies that the response satisfies the user preference. It converts the SAML assertion to an OpenID assertion in the sequence.

10. The bridge forwards the response message to IdP1.

11. When IdP1 receives the response, it verifies the message and sends the response message corresponding to the original request message in step 2. The format and protocol depend on the original.

12. On the basis of the response received from IdP1, the SP judges whether user access can be permitted or not. After this judgment, the SP returns the access response according to the request in step 1.

4. DEPLOYMENT MODELS OF A FEDERATION BRIDGE

Several deployment models of the federation bridge are available. The bridge is between IdPs. It is related to an IdP because it controls attribute exchange, much like the IdP does, but is not in a SP because it converts and regulates messages based on IdP’s policy.

A federation bridge may be a proxy entity interacting with other IdPs, an integrated entity with an IdP, or an independent entity that federates with other IdPs (Figure 8, 9 and 10). This section is described about the operation of the three models.

4.1 Federation Reverse Proxy

Figure 8 shows the reverse proxy model of a federation bridge. In this model, one IdP interacts using one protocol for federation, so the proxy resolves the cross-protocol federation. The federation reverse proxy must federate with other IdPs using the multiple protocols. It has all kinds of information on the federation. This model premises the trust relationship between IdPs because the proxy in an IdP forwards the messages between the federated providers and federation is between providers.

![Figure 8. Federation bridge: Reverse proxy model.](image)

The reverse proxy has user information because it manages the federation instead of IdPs. When it converts messages exchanged between IdPs, it needs to change the user’s identifier. It also limits the request content in the messages to prevent information leakage. It has much information and functions of a federation. The IdP does not have to add its functions when it initiates a new federation with an unknown provider. Instead, the proxy must add the functions to federate with the unknown provider.

In this model, the sequence of message flow may be different from that depicted in Figure 7. The proxy is at IdP1 or IdP2. If IdP2 converts and regulates the massage, the sequence is the same as shown. However, if IdP1 does the functions, steps 4 and 5 are reversed. In this case, the reverse proxy at IdP1 catches the request before step 4. The conversion and regulation are the same as those of step 5. After this, the bridge sends it to IdP2 with a redirection to a user agent. The processes after step 7 are the same.

In the reverse proxy model, the functions of the federation bridge are in the proxy of the IdP. The IdP can federate with each other because the proxy resolves the difference in the protocols. Therefore, this model can be adapted where they independently operate the federation proxy and manage the user’s information.

4.2 Identity Provider Integration

Figure 9 shows the IdP integration model. An IdP has federation bridge functions. In this model, the IdP has all the information on federation, authentication, and attribute management. It can convert one protocol to another by using the information, so it deals with as many corresponding federation protocols as the federated provider uses. This model premises the trust relationship between providers because they federate directly.

![Figure 9. Federation bridge: IdP integration model.](image)

In this model, one IdP understands all conditions and rules of federation, and can control access based on all of the information instead of the others centrally.
When the integrated IdP initiates federation with a new IdP, it must add new federation processes that include attribute filtering and message conversion.

The interaction sequence in Figure 7 is changed when this model is adapted because the federation bridge is not the independent entity. Before IdP1 sends a request to IdP2 in step 4, it converts and regulates the request. Then, it sends the changed request without the intermediary process. When IdP2 sends a response to IdP1, the intermediary does not catch it. IdP1 obtains the response from IdP2 directly, and it validates and converts the response message. The processes after step 11 are the same.

The integrated IdP carries out all federation processes, so its operation cost is high. However, it can control all interactions between providers easily because the IdP can regulate all messages. The integrated IdP models are suitable when centralized control is necessary, such as in an enterprise system.

4.3 Independent Federation Provider

Figure 10 shows the federation provider model. In this model, each IdP has federation functions with a federation provider, but it does not federate with other providers directly. The federation provider is an intermediary between IdP1 and IdP2, so IdP sends one message to the federation provider, and the federation provider forwards the received message to the destination provider. An IdP uses only one protocol for federation because the federation provider resolves cross-protocol federation and because it communicates with other IdPs using multiple protocols. This model premises the trust relationship between the federation provider and IdPs. One IdP does not have to keep the trust relationship with the other IdPs.

![Figure 10. Federation bridge: Federation provider model.](image)

When an IdP initiates federation with a new IdP, it does not have to federate with the new. Instead, the federation provider manages the initiation of federation between the new and itself. Because the provider converts and limits request content instead of it, the provider has all information on the relationship between IdPs.

The sequence in Figure 7 is changed when the federation provider forwards all the messages between IdPs in step 6 and 8. In the figure, the federation bridge forwards the message to the IdP directly. However, the federation provider forwards them with a redirection through a user agent, because the destination IdP may checks the user by itself. If IdP does not interact with the user agent, it cannot check the user such as authentication.

The federation provider has all information on providers, so that it must manage the information properly. In addition, it has great responsibility because all the IdPs in this model use the information. Therefore the federation provider is operated by a public entity. This model fits federation between the public services.

5. DISCUSSION

5.1 Interoperability of Federation Bridge

In the deployment models of the federation bridge, each bridge has different features because the relationship with the IdP and the bridge is different. When the bridge and the IdP federate with other entities and exchange attributes, they must keep the interoperability with other entities. Liberty ID-WSF, which enables providers to interoperate an attribute exchange, has been proposed [24]. It specifies an attribute exchange framework. ID-WSF includes the functions on how to start the interaction based on the information on federation, discovery, security mechanism, attribute limitation management (that is called “interaction service”), and interface of the federated service. As for these functions, it also specifies conformance test features. By using functions of the conformant ID-WSF, providers can federate in an interoperable manner. If providers have the same functions as ID-WSF specifies, they can also interoperate.

Aiming for verification of whether a federation bridge is interoperable, the author compared the federation functions of the federation models on their information management, discovery, and attribute management with those of ID-WSF. The security mechanism depends on the protocol that providers use, and the bridge does not decide the mechanism, so the security mechanism of the models is not considered in this section. The federated service interface is also out of the scope of the comparison because it also depends on the protocols of interaction and because the bridge does not specify it. The comparison between the three models is described in Table 1.

**Federation Information**

Federation information contains information on the ways of interaction, the conditions of attribute exchange, and the IdPs to be federated. Each model has information about federation, but the entities that maintain the information are different.

In a federation reverse proxy, the IdP has the information on other IdPs to be federated. However, the IdP does not have to store the ways and conditions because the reverse proxy checks and converts the IdPs. In this model, the IdP has part of federation information, and the reverse proxy for the federation keeps the other information instead of other IdPs.

The IdP integrated with a federation bridge has all the information for the federation. It centralizes the control.

A federation provider has all the information on the federation instead of the IdP, though the IdP has the information to interact with the federation bridge. In this case, the IdP federates with other IdPs via the provider, so that the bridge has all the federation information.

**Discovery**

Discovery means that a provider finds the destination provider to send a request. The three models have the discovery process.

The IdP with a reverse proxy discovers the destination of the request, and the proxy uses the discovered destination when it regulates the request. The IdP has the information on the discovery.

An integrated IdP also has a discovery function because it manages all federation and because no entities intermediates messages.
A federation provider discovers the recipient of a request message instead of the IdP. The IdP only understands the federation provider and sends all requests to it, and the federation provider decides the proper IdP to interact with based on the requests. In this model, the IdP can reduce the number of operations.

### Attribute Control

When a provider forwards attributes to other providers, it checks whether it can accept the exchange. This function is carried out by a federation bridge.

When the bridge is at a federation reverse proxy, the IdP does not have a limit on the shared attributes. Instead, the reverse proxy regulates the request message and limits the attributes to be exchanged. The IdP does not have to manage the attribute control.

An integrated IdP must check the attributes to be transferred because it controls everything. In this model, the IdP has to manage all the information on the federation, because it can control all interactions.

In the federation provider model, the IdP does not have to regulate the request as in the federation reverse proxy model. The federation provider controls all interaction between the IdPs, so the IdP can reduce the load of attribute control.

As described above, each model has the same functions as ID-WSF specifies to conformance test. The providers in every model can federate with other providers in an interoperable manner.

### 5.2 Validation of Messages

Providers may need the validation of received messages when they federate. In the situation, they check the digital signature in the message. Because a federation bridge mediates the interaction between them, it needs to manage the signature. In this paper, the validation of the signatures is not mentioned, but the bridge needs to manage the digital signatures.

In the federation bridge models, we suppose that a federation bridge and providers have a trust relationship; i.e. a receiver of the message trusts a sender. This means that the receiver believes that the content of the message from the sender are correct. Under this relationship, the receiver checks the signature generated by the bridge to validate the received messages.

When the receiver of the message from a federation bridge needs to check the original message that the bridge has received from the original sender, the receiver cannot validate the original signature by the forwarded message by the bridge. When the bridge converts a message, the signature in the message is also changed. Though the receiver checks the signature in the altered message, it cannot validate the original signature that the original sender generates. Several methods for the validation can be considered.

One method for the validation is that the bridge keeps all exchanging messages and offers audit services about message validation. When the receiver validates the message, it requests the validation to the bridge instead of validation for itself. It gets the result of the validation from the bridge. This method needs additional interaction between the bridge and the message receiver. The scalability of the bridge remains as a matter to be discussed.

Another method is that the bridge sends the original message and signature in addition to the altered message. In this method the receiver can check the original digital signature though it cannot understand the protocol of the original message. This model does not require the receiver to interact with the bridge. The receiver can check whether the signature in the original message is valid, but cannot always check whether the content in the altered message are the same as those in the original message because the receiver may not understand the original protocol. When it needs the validation of the message content, additional mechanism such as the audit service by the federation bridge is needed.

### 5.3 Extensibility of Federation Bridge

A federation bridge can have more abilities on the federation besides conversion between two protocols. In section 3.2, the bridge deals with the conversion between OpenID and SAML, but it can be converted to OAuth [25], Cardspace [22], and other formats, too. If it can accept the many protocols, the IdP can federate with more providers and share more attributes between them. The extensibility of the federation relationship is based on the number of federated providers and protocols.

### 6. CONCLUSION

A federation bridge that connects different kinds of identity federation protocols was described in this paper. The bridge can connect different providers though they use different protocols.

The functions for regulation of exchanging messages were specified. This regulation process limits the unnecessary attributes to be exchanged between providers; therefore, they can reduce the
risk of privacy leakage. Moreover, they can satisfy the requirements where they and a user describe the rules because the bridge limits the attributes based on the requirements when it transfers a message. The author also specifies the sequence to exchange user’s attributes through the federation bridge. Using this bridge, an identity provider can federate with other providers in an interoperable way though they have a different manner of interaction.

The deployment models of the federation bridge were also described. The bridge is an intermediate entity, so it may be a proxy, deployed within the IdP, or it may be deployed as an independent provider. Each model has its own features; hence, the best domain of applicability is different. The bridge can have more functions for federation such as the message validation or a conversion of multiple protocols, though the scalability of the federation bridge remains as a matter to be discussed. We intend to investigate multiple-protocol federation to manage the attributes properly, enabling greater convenience for the user.

7. REFERENCES


