OIO Identity-Based Web Services

Scenarios

National IT and Telecom Agency
Ministry of Science
Technology and Innovation

OIO Identity-Based Web Services

Scenarios

Version 1.1

Danish National IT and Telecom Agency
May 2010

Content >

Introduction

>

This document contains a number of scenarios for Danish eGovernment illustrating identity-based web services (IDWS). The term “identity-based web service” in this context means web services that act on behalf of a user or are personalized with the user’s data in contrast to normal web services which do not execute in the context of a particular user. Our usage of the term “identity-based” web services both encompasses what Liberty Alliance Project calls “identity-based” and “identity-consuming” web services. Identity in this context simply refers to some claims (e.g. attributes) about the user. Claims can tell properties about the user (e.g. age or sex), identifiers used in systems (e.g. pseudonyms or user names), authorizations (e.g. “user is administrator of system X” or “user is a medical doctor”) or directly identify a physical person (e.g. using a social security number). It is only necessary to include person identifiers if the service being requested requires it[1].

The first scenarios below cover basic functionality which is progressively expanded into more advanced scenarios dealing with aspects such as privacy and user consent. In the current version, only basic scenarios are covered but other scenarios can be added later.

The scenarios serve illustrative purposes and are primarily described via the interaction (message flow) between different system components. Many other scenarios and architectures are possible and are by no means disallowed. However, when using the below profiles, the requirements of the profiles must of course be followed.

To enable the scenarios, a number of interoperability profiles defining message formats and other normative requirements have been developed:

- OIO WS-Trust Profile [OIO-WST]
- OIO WS-Trust Deployment Profile [OIO-WST-DEP]
- Liberty Basic SOAP Binding [LIB-BAS]
- OIO Bootstrap Token Profile [OIO-BOOT]
- OIO SAML Profile for Identity Tokens [OIO-IDT]

A secondary goal of this document is to illustrate how these profiles can be combined to achieve business goals in the form of scenarios. It is important to note that the above profiles can be combined in many different ways depending on the specific architecture and solution context, and that the examples given only serve as illustrations. For example, it is anticipated that other profiles for bootstrap tokens will emerge or that sub-profiles of the OIO SAML Identity token profile will be developed by sectors / domains which e.g. need special identifiers.

[1] A more precise term instead of “identity-based” may be “claims-based” web services. However, experience shows that casual readers have difficulty comprehending what “claims-based” means. Until this changes, the term “identity-based” will be used.

>

The reader is assumed to be familiar with the existing OIO SAML profile for Web SSO described in [OIO-SAML].

Business Goals and Requirements

The goal of the scenarios and associated profiles are to realize the following high-level business goals and requirements:

1. Specify how a web service consumer (WSC) in a secure and privacy-respecting way can invoke an identity-based web service provider (WSP) on behalf of user.
2. Outline an open and de-coupled architecture that can later be extended with new components as standards evolve to support more advanced scenarios.
3. Ensure interoperability and COTS support by leveraging international standards and profiles – including SAML, WS-* and Liberty Alliance.
4. Support requirements from advanced eGovernment applications including interaction across organizational boundaries while complying with laws and regulations.

The legal aspects and security properties of using identity-based integration have been analyzed in a Danish context in [PD].

Traditional versus identity-based integration

To illustrate what's new we start by considering how traditional web service integrations used to be secured. Normally, a user would authenticate to an application, and the application would invoke foreign web services by authenticating with its own credentials (e.g. using a secure transport such as TLS with a client certificate or by signing the SOAP request). The external service would authenticate the request and return a response if the partner was allowed to invoke the requested service.

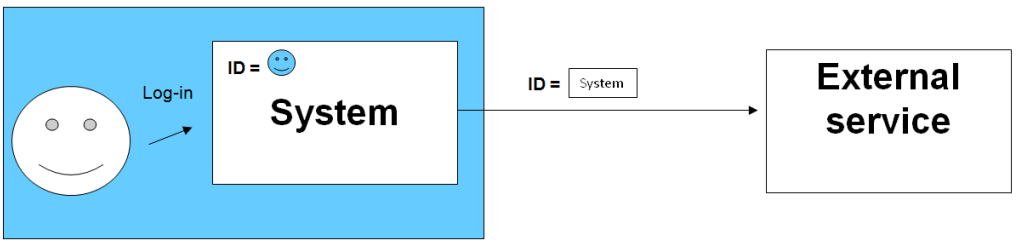

Figure 1: Traditional integration

This model may be suitable in situations where there is a great degree of trust between the service consumer and provider, and when the service consumer is allowed to represent all users (e.g. internal applications). However, it may not be suitable when users are external to the invoking system (e.g. citizen self-service applications), when there is limited trust between the two organizations, or when a more fine-grained security model is desired.

Identity-based integrations are different in that the user identity is transferred in the web service invocation by means of a security token (often issued by a trusted third party). Of course, the invoking system will also have to authenticate (not shown in the

>

figure). This model provides the external service with assurance that the user is actually logged in and that the requesting system has obtained a security token for the web service invocation. Thus, the external service does not have to trust the invoking system to the same degree - but instead needs to trust the issuer of the security token (a trusted third party).

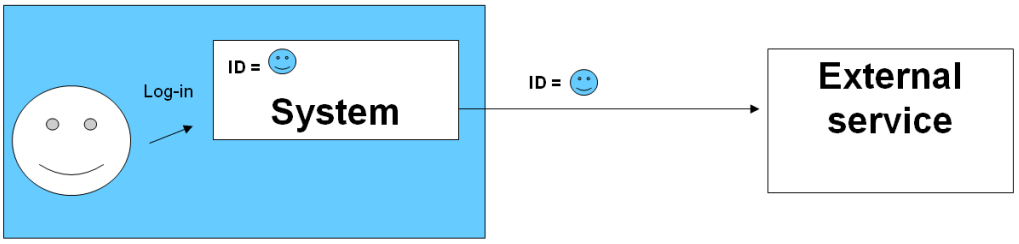

Figure 2: Identity-based integration

Architecture Overview

>

This chapter describes the individual components involved in the scenarios. The architecture is later refined as the scenarios become more advanced.

Technology choices

A number of standards have been selected for various parts of the scenarios:

- SAML 2.0 tokens and protocols are used for web single sign on and single logout. This part of the architecture has already been established.
- The WS-Trust protocol is used for requesting, issuing and validating security tokens.
- WS-Security combined with elements from Liberty WSF 2.0 SOAP Binding is used to secure web service invocations (SOAP messages) including transferring security tokens, handling message integrity, authenticity, confidentiality etc.

The rationale behind these choices will not be discussed further in this document.

Architecture components

The figure below illustrates the main architectural components involved in the scenarios:

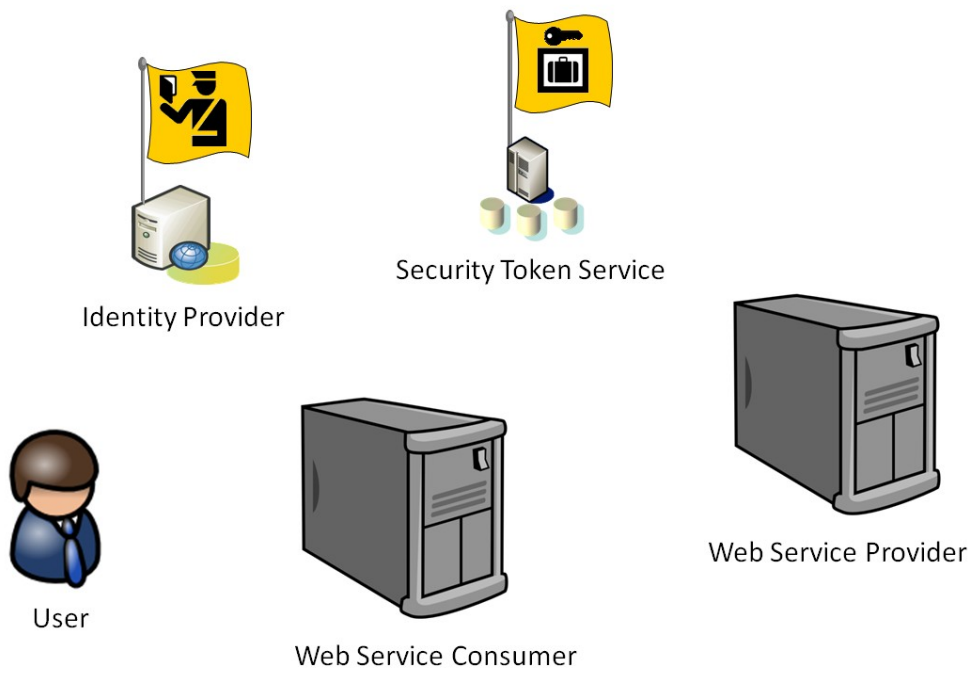

Figure 3: Architecture Components

The components are:

- The user
- The user's client: in most scenarios it is a standard web browser, but it may also be a rich client

>

- A service consumer who offers a personalized application to the user. In order to service the user, the application needs to call an identity-based web service in another domain (i.e. a web service provider).
- A web service provider who offers an identity-based web service to other service providers. The service may for example allow data about a particular user to be retrieved.
- An Identity Provider who runs a login service. In most scenarios it will be a (OIO) SAML 2.0 based Identity Provider offering login using digital signature.
- A Security Token Service (STS) who issues tokens for access to web services.

The components can be deployed and hosted in various configurations depending on the local context. For example, the user may or may not belong to the same organization as the web service consumer, the STS may or may not be co-located to the IdP, and both STS and IdP may be internal / external to the user's organization.

Scenario 1: STS co-located with Service Provider

>

This scenario describes how a user can first access a normal web application using the single sign on service of an Identity Provider. Subsequently, the application requires information from an identity based web service hosted at another service provider. It therefore takes the role of a web service consumer (WSC) and invokes a web service provider (WSP) on behalf of the user. In order to identify the user, the WSC contacts a local security token service (STS) to retrieve an identity token, and the token is subsequently placed in the web service call.

Business Context

The scenario can be used when one government agency provides a service exposing citizen data which is used by other agencies in their self-service applications. A citizen self-service application may for example need data about the citizen located with another agency or must perhaps send an event to the other agency on the user's behalf.

Prerequisites

The scenario can be used when the agency providing the web service trusts the calling agency only to invoke the service on behalf of users who are actually logged in and wish to perform a transaction. This implies trust at the organizational level as well as the ability of technically being able to validate the other party's digital signatures.

Furthermore, the agencies must share the same identifiers for the user – for example the social security number such that no identity mapping is needed.

We further assume that the interaction required between SP1 and SP2 is a simple request-response pattern and not a long-running conversation.

Lastly we assume that Service Provider 1 knows the end-point of Service Provider 2's services at deployment time such that dynamic discovery is not necessary.

Message flow

The figure below illustrates the message flow in the base scenario:

>

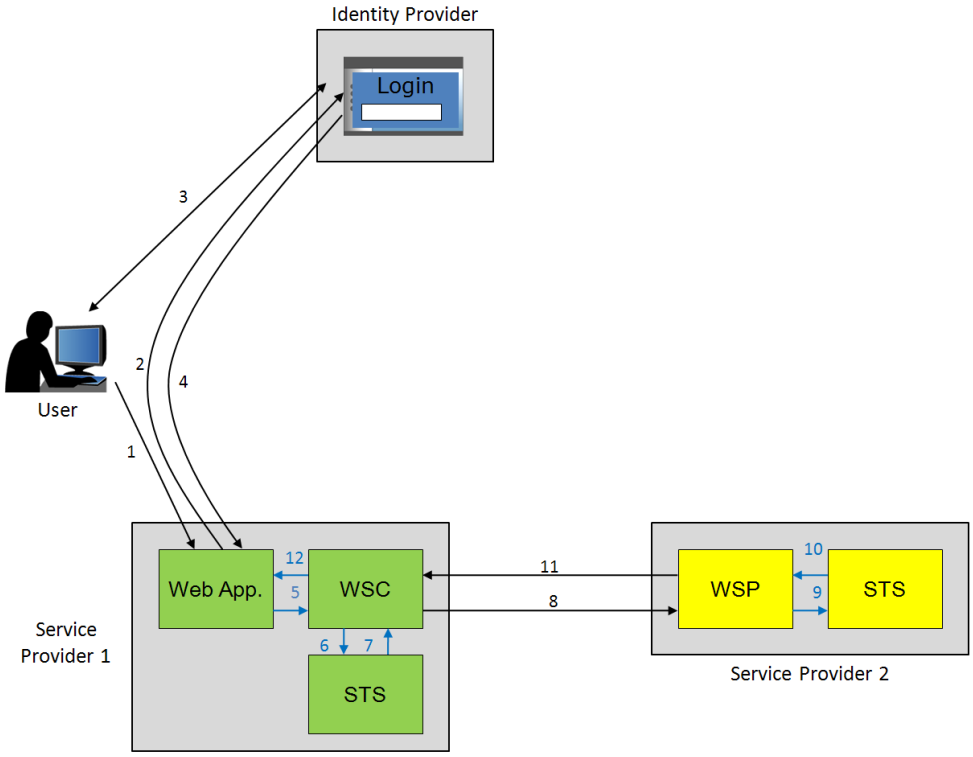

Figure 4: Local STS Scenario

Steps 1 to 4 correspond to a normal web SSO flow as described in [DK-SAML]:

1. The user requests a protected resource in the web application at SP1 which requires logon.
2. The user's browser is redirected to the Identity Provider for logon.
3. The user authenticates to the Identity Provider using his credential (e.g. a digital signature bound to a digital certificate).
4. A SAML assertion is issued and the user is redirected back to the service provider with the assertion. The SAML assertion is validated and the service provider creates a session with the user.

At this point the web application needs to invoke a remote identity-based web service. For example, it might need to retrieve some of the user's personal data residing with another service provider.

5. The web application transfers control to the web service consumer (WSC) component. In the call it passes the SAML assertion received during the web sign-on flow.
6. The web service consumer contacts a local security token service within the same security domain to have an identity token for the user issued.
7. The STS issues the assertion and returns it.
8. Service provider 1 issues a web service call to service provider 2 passing the user's identity token.
9. The web service provider (WSP) at SP2 invokes a security token service to get the identity token validated.

>

10. Since SP2 trusts the SP1 to issue identity tokens, the security token can be validated internally (signature/certificate validation). The user identity is extracted and returned to the WSP.
11. The WSP fulfills the request in the context of the user's identity and returns the answer to the WSC at service provider 1.
12. The WSC parses the response and returns the requested data to the web application. The application can now serve the user's original request.

Note that steps 5, 6, 7, 9, 10 and 12 marked with blue color are considered private to the implementation at SP1 and SP2 and will not be considered for profiling below.

Profiles and requirements identified in the scenario

The scenario requires a number of message / protocol profiles to specify the exact message content, processing rules and security requirements:

- A SAML identity token profile specifying the format of the tokens (assertions) returned by the STS. The identity token profile specifies how the user is represented via attributes and whether the assertion is signed, encrypted etc. Further, a policy should indicate whether the token can be used for multiple web service invocations, the lifetime of the token etc.
- A WS-Security profile (or Liberty WSF-profile) for how the user's identity token is passed in the web service call and how/if it relates to securing the web service call (e.g. using a subject confirmation element with a holder-of-key).
- The parties need to authenticate each other so a trust mechanism must be established.

The input to the local STS is considered private to SP1. Later scenarios will identify further profiles needed when the STS is external.

Limitations of the scenario

A number of issues are not considered in the above simple scenario:

- SP2 must trust SP1 to issue identity tokens and call the web service only on behalf of legitimate requests from the user. In some real life situations it is not realistic to assume such trust. However, when possible the scenario represents a simple pattern for identity based web services.
- SP1 must know the identity of the user at SP2 in order to issue an identity token. Thus, if the user has a different identity at SP1 and SP2 it will not work. An external STS (described in next scenario) may perform identity mapping such that the two agencies do not have to share the same user identifier.
- There is no fine-grained authorization shown in the scenario. SP2 may perform internal authorization according to local policy.

Scenario 2: STS issued identity token

>

This scenario describes how a user can first access a normal web application using the single sign on service of the Identity Provider. Subsequently, the application requires information from an identity based web service hosted at another service provider. It therefore takes the role of a web service consumer (WSC) and invokes a web service provider (WSP) on behalf of a user. In order to do this, the WSP contacts a Security Token Service to retrieve an identity token which describes the user identity.

Business Context

The scenario can be used for example when one government agency provides a service exposing citizen data which is used by other agencies in their self-service applications. For example, the citizen web application needs data about a citizen located with another agency or needs to send an event to the other agency on the citizen's behalf.

In contrast to scenario 1, the user does not need to have the same identity at SP1 and SP2 since the Identity Provider / Security Token Service can perform identity mapping. The STS can further encrypt attributes in the identity token or encrypt the entire token under SP2's public key such that SP1 does not learn the user's identity at SP2. Thus, this scenario supports stronger privacy than the previous scenario.

The coupling of the Identity Provider and STS makes it possible to ensure that the user actually had a valid session with Identity Provider / SP1 at the time where the identity token was issued. SP2 will have to define a local policy regarding how recently the identity token must be issued compared to the time of the web service invocation. Furthermore, a policy for the STS should govern life time of identity assertions and what checks that are performed on the user's SSO session when the identity token is issued (e.g. identity tokens are only issued when the user has an active browser session).

Prerequisites

We assume that both service providers trust the Identity Provider / Security Token Service and will accept the tokens they issue. Furthermore, we assume that Service Provider 1 is able to authenticate messages to the Identity Provider and that the two service providers can authenticate messages from each other.

Lastly, we assume that Service Provider 1 knows the end-point of Service Provider 2's services at deployment time such that dynamic discovery is not necessary.

Message flow

The figure below illustrates the message flow in the base scenario:

>

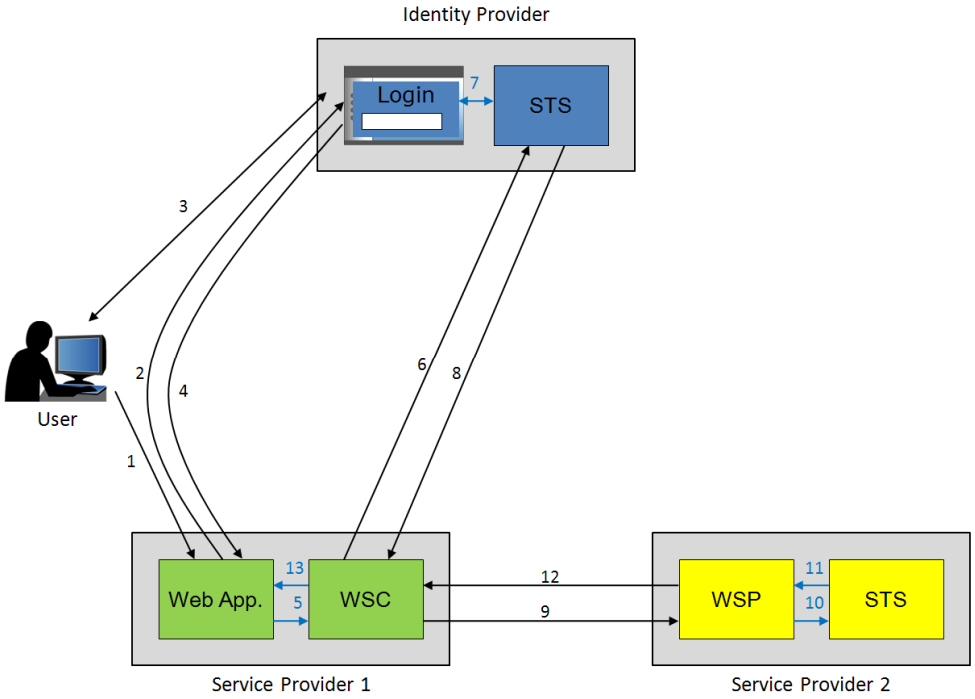

Figure 5: External STS Scenario

Steps 1 to 4 correspond to a normal web SSO flow as described in the previous scenario and they will not be repeated here. Note that the SSO service at the Identity Provider may do some internal housekeeping to facilitate later issuance of identity tokens by the STS.

After step 4, the web application needs to invoke a remote identity-based web service:

5. The web application transfers control to the web service consumer component. In the call it passes the SAML assertion received during the web sign-on flow.
6. The web service consumer extracts the end point for the Identity Provider's STS and the bootstrap token from the authentication assertion. It contacts the Identity Provider's Security Token Service to retrieve an SAML identity token for the user which can be given to service provider 2. In the call to the STS SP1 authenticates itself by signing the request, passes the bootstrap assertion and indicates which service or service provider, the token is intended for.
7. The STS validates the request issues a SAML Identity Token containing the user's identity (exchanges the bootstrap token for an identity token). It may cooperate with the IdP SSO component depending on the internal implementation.
8. The identity token is returned to service provider 1.
9. Service provider 1 issues a web service call to service provider 2 passing the user's identity token.
10. The web service provider at SP2 calls a security token service to get the identity token validated.

>

11. Since SP2 trusts the Identity Provider / STS, the security token can be validated internally (signature/certificate validation). The user identity is extracted and returned to the WSP.
12. The WSP fulfills the request in the context of the user's identity and returns the answer to the WSC at service provider 1.
13. The WSC parses the response and returns the requested data to the web application. The application can now serve the user's original request.

Note that steps 5, 7, 10, 11 and 13 marked with blue color are considered private to the implementations and will not be considered for profiling below.

Profiles and requirements identified in the scenario

In addition to the message / protocol profiles already identified in the last scenario, this scenario needs the following:

- A WS-Trust profile for requesting and returning a security token. It needs to specify how SP1 locates the Identity Provider's STS, how it informs the IdP / STS about the user identity (for example using the bootstrap assertion), which service providers (or fine-grained services) he needs a token for, how the request / response messages are to be secured. Furthermore, bindings should also be profiled.

The figure below illustrates how the various OIO profiles are combined in the scenario:

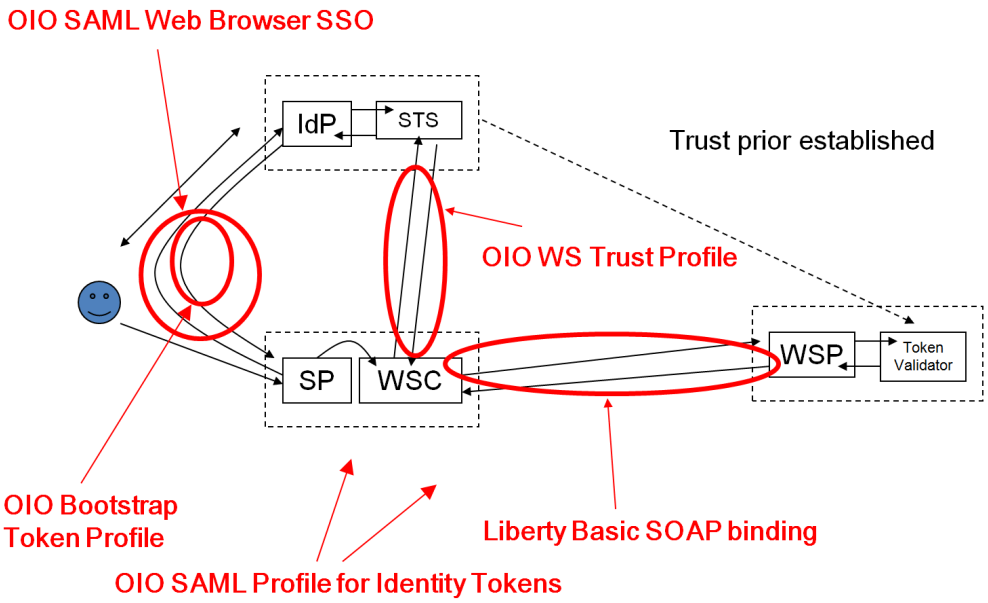

Figure 6: Profiles and messages flows

>

Limitations of the scenario

A number of issues are not considered in the above simple scenario:

- There is no mechanism for the user to provide his explicit consent to SP2 that SP1 may access the service. It is assumed, that the user accepts general terms and conditions of the Identity Provider which allows the Identity Provider to issue identity tokens to other service providers; the STS may expose a user interface where the user can control which service provider that can request tokens on the user's behalf (and in which situations). However, these trust assumptions may not be sufficient in some scenarios for example when the web service exposes sensitive data such criminal records or medical data. Here explicit consent to data release may be required which is described in the next scenario.
- There is no fine-grained authorization shown in the scenario regarding who can get which of the user's attributes or access which of the user's services. However, SP2 may perform internal authorization according to local policy for example taking the service, user identity and SP1 identity into account.
- We have implicitly assumed that the Identity Provider knows which attributes to include in the identity token for SP2. In a more advanced scenario, SP2 may define and advertise a policy regarding which attributes it requires, and SP1 may ask the IdP / STS for these when requesting the token.
- There is no discovery mechanism described – we have assumed that SP1 knows the location (end points) of the IdP, STS and SP2.
- The scenario does not show what happens when the user logs out of the IdP. For example, the identity token may have a lifetime that exceeds the time of single logout.

Scenario 3: Explicit user consent

>

This scenario extends the previous by enabling Service Provider 2 to ask the user for his explicit consent to the invocation of the identity-based web service. This is achieved through Service Provider 2 requesting Service Provider 1 to redirect the user's browser for direct interaction.

Business Context

In some situations a web service provider (or the corresponding user) may have defined a policy stating that a particular identity-based web service cannot be invoked without the user's explicit consent. Furthermore, the user's consent may be context-dependent in such a way that the user must be asked for every invocation. This can for example be relevant for services exposing criminal records, medical data etc.

Requirements for consent may be necessary to comply with regulations, it may be the result of lacking trust between the parties (SP2 not trusting SP1), the wish by SP2 to reduce liability, or it may originate from the desire to give users direct control over who can access their data in order to enhance privacy aspects.

Prerequisites

Service Provider 1 (WSC) has an active browser session with the user and is willing to redirect the user's browser to Service Provider (WSP) for direct interaction.

Further, the user must be able to authenticate to Service Provider 2 as part of giving his consent.

As in the previous scenarios we assume that Service Provider 1 knows the end-point of Service Provider 2's services at deployment time such that dynamic discovery is not necessary.

Message Flow

The figure below illustrates the message flow. The first eight steps are identical to the previous scenario and are not shown.

>

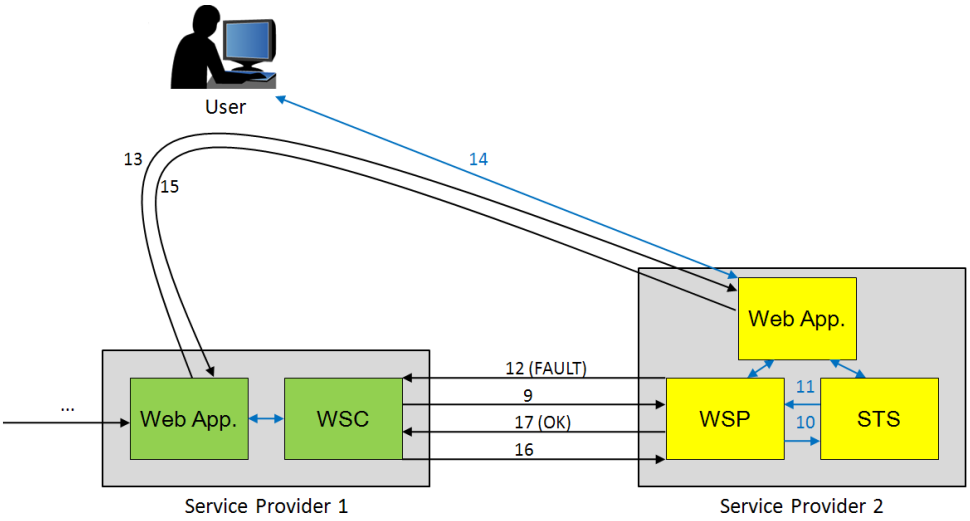

Figure 7: Requesting explicit user consent

The steps are:

9. Service Provider 1 (WSC) invokes the identity-based web service passing the user's identity token from Identity Provider. In the call SP1 indicates that it is willing to redirect the user to SP2 if necessary.
10. The web service provider at SP2 calls a security token service to get the identity token validated.
11. Since SP2 trusts the Identity Provider / STS, the security token can be validated internally (signature/certificate validation). The user identity is extracted and returned to the WSP.
12. Inspection of local policy reveals that explicit user consent is required for the current transaction. Therefore, a fault is returned indicating that the user must be redirected along with a re-direct URL.
13. Service Provider 1 redirects the user to the URL specified by Service Provider 2. In the call it passes a re-direct URL where the user must be re-directed back.
14. Service Provider 2 authenticates the user and subsequently asks the user for consent to the current transaction.
15. After having collected consent (or the opposite) Service Provider 2 redirects the user's browser back to Service Provider 1.
16. Service Provider 1 (WSP) now re-submits the original request.
17. Service Provider 2 (WSC) receives the request, processes it provided user consent was given, and returns the response.

Note that steps 10, 11 and 14 marked with blue color are considered private to the implementations and will not be considered for profiling below.

Furthermore, the following aspects are also considered private to Service Provider 2:

- How user authentication is performed. It can be a direct authentication or via the Identity Provider (i.e. a normal web SSO flow).
- The policy determining when consent is required.

- The form where consent is requested and granted – e.g. the specific questions the user is asked, which options he has, whether he is required to digitally sign consent statements etc.
- The scope of the consent – e.g. whether consent applies to just this service invocation, a transaction consisting of multiple web service invocations or all subsequent invocations by Service Provider 1 at Service Provider 2. The right balance between user control / privacy and bothering the user with too many questions will probably vary considerably with applications and organizations.

Profiles and requirements identified in the scenario

In addition to the message / protocol profiles already identified in the last scenario, this scenario requires the following specifications:

- How Service Provider 1 indicates in the web service call (e.g. in a SOAP header) that he has an active browser session with the user and is willing to perform a redirect. Such a header is defined in Liberty WSF 2.0 SOAP Binding [LIB-SOAP].
- A Redirect request protocol where Service Provider 2 can request Service Provider 1 to redirect the user and control can be transferred back. Such a protocol is found in the Liberty WSF 2.0 SOAP Binding [LIB-SOAP].

Limitations

- The above scenario assumes that the user has an active browser session with Service Provider 1. There may be situations where applications need to invoke identity-based web services without having a session with the user owning the service / data – e.g. applications for employees in government agencies supporting case processing. If user consent is required in such situations a permanent interaction service is needed which provides a reliable means of contacting the user (e.g. via the user's mobile phone). Such an interaction service is specified in [LIB-INT].

Scenario 4: Rich Client and local IdP / STS

>

The next scenario involves an employee in one organization using a rich client to invoke services in another organization. For example, within eGovernment this might be a case worker in one agency who needs information provided by a foreign agency. The employee will authenticate to the application using the local network login, and when the application client needs to invoke the foreign service, it will use a *local* Security Token Service to obtain a security token. The Security Token Service relies on the network login (e.g. Kerberos) for user authentication and may further fetch information on the user's roles from a directory and put them in the token as well.

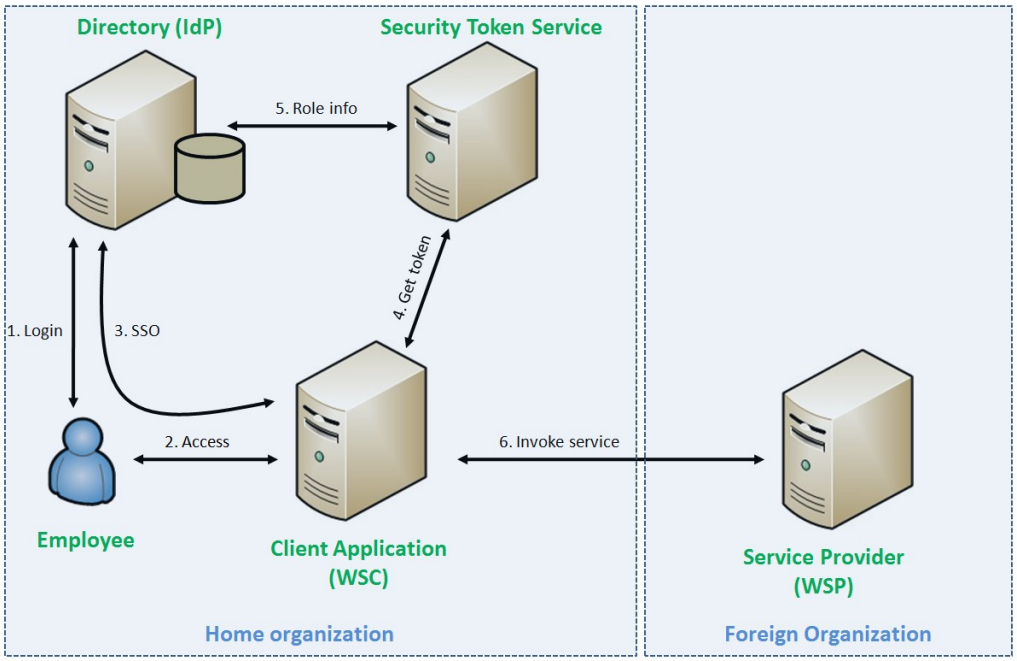

Figure 1: Scenario with Rich Client

The steps are as follows:

1. The user logs in at his local network in his home organization at the beginning of his work day (e.g. Active Directory). Thus, the local directory plays the role of Identity Provider.
2. The user starts his client application.
3. The client application automatically logs the user in (directory-based single sign-on in the home organization).
4. The client application needs to invoke a foreign identity-based web and therefore requests a security token from the local STS.
5. The STS authenticates the user (local SSO) and performs a lookup in the local directory to find which roles / rights / privileges the user has. The STS may perform a transformation from organizational roles to system roles required by the WSP.
6. The application invokes the foreign service using the security token returned by the STS.
7. The invoked service verifies it trusts the STS who has signed the token, the organization signing the request, and if everything is ok it allows the service to execute.

>

Profiles and requirements identified in the scenario

No new profiles are needed for this scenario. The Liberty Basic SOAP Binding [LIB-BAS] governs the SOAP call from client to Web Service Provider. The authentication of the user to the local application, and the interaction between the local application and local Security Token Service is considered local / private, since it will be based on the local directory / SSO implementation.

Prerequisites

The prerequisites for this scenario are:

- A local SSO solution within the user's home organization has been established and the user has appropriate credentials.
- The WSP trusts the local STS / home organization to: correctly authenticate local users, manage credentials issuance and revocation processes, assignment of roles and stating correct assurance level.
- The STS may have to know which system roles the WSP expects in the token (if roles are needed).
- The STS must state the user's assurance level and it must be appropriate for the WSP service. For example, some services might require that the user is strongly authenticated (e.g. assurance level 3) which means it is usually not sufficient to authenticate using a username and password.
- An administrator has assigned roles to the user in the directory (if needed).
- The client (WSC) and service (WSP) both have a digital signature which can be used for authentication of the request and response.
- The STS has a digital signature which can be used to sign tokens.

Scenario 5: Rich client and external IdP / STS

>

This scenario is very similar to the previous except for the Identity Provider and Security Token Service being external to the user's home organization. This situation is believed to be central to the deployment of identity-based web services in the Danish health care sector.

The scenario still involves an employee in one organization using a rich client to invoke services in another organization. The employee will authenticate to the external Identity Provider using an employee digital signature[2], and he will get back a SAML token for the application which contains a bootstrap token. Once the application client needs to invoke the foreign service, it will exchange the user's bootstrap token for an identity token at the external STS, and then use the identity token in the invocation of the web service.

The STS may fetch user roles, privileges or attributes (claims) from a local repository and add them to the issued token, such that they can be used in access control decisions at the web service provider. This is however outside the scope of this scenario and identified profiles.

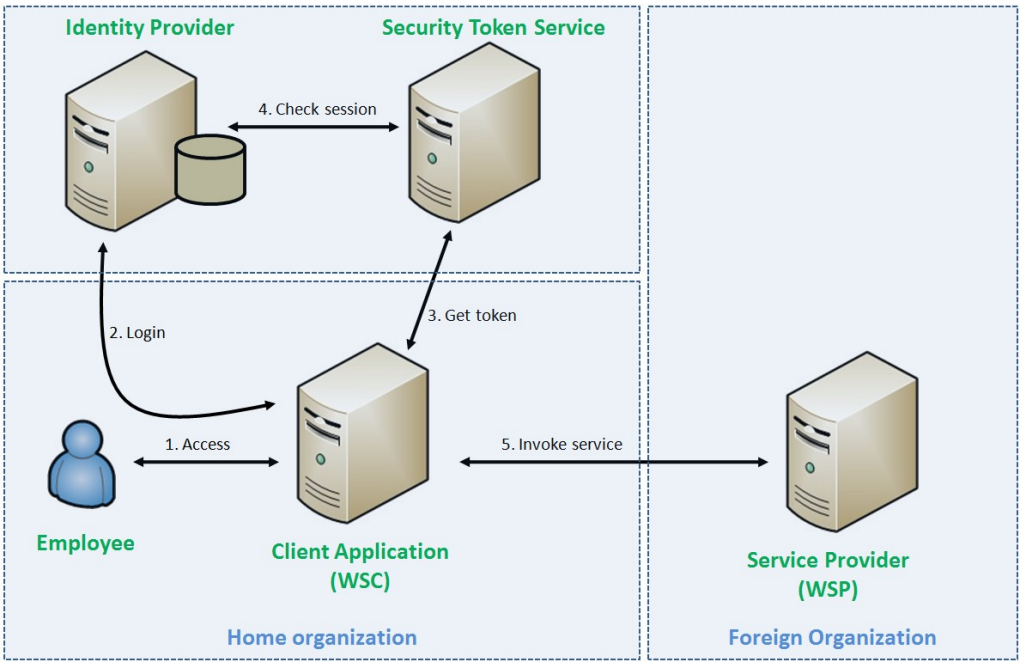

Figure 8: Rich client with external IdP / STS scenario

The steps are as follows:

1. The user tries to access a local application.
2. The application requires the user to be authenticated via the external Identity Provider. In order to achieve this, it requests the user to sign an authentication token using his employee certificate. This token is sent to the IdP, which

[2] A so-called MOCES certificate.

validates the user signature and exchanges the authentication token for an identity token to the client application. The client application validates the IdP-issued token and gives the user access. The authentication token includes a bootstrap token which can be used later at pre-defined Security Token Services (see below).

3. The application needs to invoke a foreign service at the Web Service Provider. It therefore invokes the external STS to exchange the bootstrap token for a token targeted at the service.
4. The STS validates the request and bootstrap token, and optionally contacts the Identity Provider (issuer of the bootstrap token) to determine whether the user is still logged in. This last step is optional and probably only possible when IdP and STS are co-located (deployed as part of same system). In any case, the STS-IdP interaction is considered private. If everything is ok, the STS returns an identity token to the application for the desired service.
5. The application invokes the foreign service using the new security token returned by the STS.
6. The invoked service verifies it trusts the STS who signed the token, the organization signing the request, and if everything is ok it allows the service to execute.

Profiles and requirements identified in the scenario

Compared to the previous scenario, we now need an authentication token profile for authentication of the user (via the rich client) to the Identity Provider. The OIO WS-Trust profiles can be used for conveying token request/response messages between application client and IdP / STS. Note that the IdP in this context is actually an STS which exchanges an authentication token (signed by the user) for an application-specific identity token. Here the WS-Trust requests from the WSC are signed by the WSC as specified in the OIO WS-Trust profile.

Interactions between co-located Identity Providers and Security Token Services will not be profiled as part of OIO IDWS as it is considered private to deployments.

Prerequisites

The prerequisites for this scenario are:

- The WSC trusts the IdP to authenticate the user (accepts tokens issued by the IdP).
- The WSP trusts tokens issued by the STS, and the STS in turn trusts tokens issued by the IdP.
- The STS may have to know which roles (claims) the WSP expects in the token (if roles / claims are needed).
- The STS must state the user's assurance level and it must be appropriate for the WSP service.
- When the Identity Provider authenticates the user, it knows which potential Security Token Services that can later be contacted, and includes bootstrap tokens for these in the assertion for the client application.

>

- The STS, client (WSC) and service (WSP) both have a digital signature which can be used to sign messages (and tokens in case of the STS).
- The user has an employee certificate which can be used for authentication (signing the authentication token being sent to the IdP).

Advanced scenarios

>

This document is work-in-progress and additional scenarios can be added later including:

- User client includes an identity selector
- Required claims defined via policy.
- SP2 has no trust relationship with IdP / STS (brokered trust)
- Logout
- Provider Chaining
- STS at IdP resolves generic address of SP2 to specific endpoint in the same go as it converts the bootstrap token to a SP2 specific token (aka Liberty discovery service).

References

>

[WST-OIO] "OIO WS-Trust Profile 1.0", Danish National IT and Telecom Agency.

[OIO-WST-DEP] "OIO WS-Trust Deployment Profile Version 1.0", Danish IT and Telecom Agency.

[LIB-BAS] "Liberty Basic SOAP Binding Profile", version 1.0, Liberty Alliance Project.

[OIO-BOOT] "OIO Bootstrap Token Profile", Danish IT and Telecom Agency.

[OIO-IDT] "OIO SAML Profile for Identity Tokens, V1.0", Danish IT and Telecom Agency.

[OIO-SAML] "OIO Web SSO Profile V2.0", Danish IT and Telecom Agency.

[LIB-INT] "Liberty ID-WSF Interaction Service Specification", Version 2.0, Liberty Alliance Project.

[LIB-SOAP] "Liberty ID-WSF SOAP Binding Specification", Version 2.0, Liberty Alliance Project.

[PD] "Identitetsbaserede webservices og personlige data", Version 1.0, IT- og Telestyrelsen.

<