Federated Identity Patterns in a Service-Oriented World

THE ARCHITECTURE JOURNAL Input for Better Outcomes

by Jesus Rodriguez and Joe Klug

Summary: Throughout the last decade, identity-federation techniques have been a fundamental complement of distributed programming technologies such as COM+ or CORBA. However, it is not a secret that the use of identity federation on those technologies never gained a wide adoption in real-world implementations, mostly due to some of the limitations that those technologies present in terms of interoperability or the use of proprietary protocols. Some of those challenges had a big influence on the adoption of Web services as the architecture style of choice for implementing distributed solutions. Consequently, during the last few years, Web services has emerged as the new battlefront for identity-federation solutions. In this article, we examine several identity patterns, and consider the strengths and weaknesses of each model.

Contents

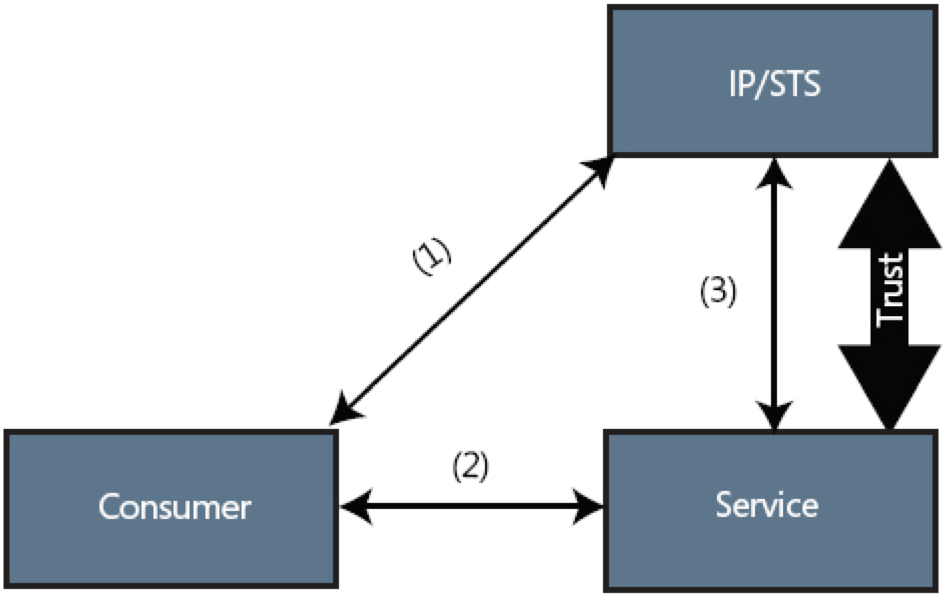
Introduction

The goal of identity federation is to enable resource access across completely unrelated security domains by sharing a limited amount of information such as security identities and policies. When it is implemented as an architectural style, identity federation is often positioned as an alternative when identity centralization is not a viable solution. In federated environments, clients are able to use a single set of credentials and identity information, issued under a specific security domain, to access resources on a completely different security domain. Behind those simple principles, identity federation has been one of the most challenging aspects of IT solutions for the last few decades. However, despite its inherent complexity, the fundamental identity-federation concepts have powerful analogies in our everyday lives. Identity documents such as driver licenses, passports, and credit cards are normally used as the main artifacts of robust identity-federation procedures like the ones that are used to interact with national institutions, such as, in the U.S., the Department of Homeland Security and Internal Revenue Service. Obviously, software-based identity-federation solutions present some significant differences, as we can't always rely on humans for resolving and mapping identities. This is precisely the case with distributed programming technologies on which clients and servers need to be able to establish secure communications across different security domains.

A lot of work has been done in the Web services security and federated identity fields. As a result, the industry has produced a series of Standards that represent the fundamental building blocks of Web services federation solutions. Among those Standards, we can list WS-* protocols such as WS-Security, WS-SecureConversation, WS-Trust, WS-SecurePolicy, and WS-Federation, identity-representation languages such as SAML and XACML, and federation-specific specifications like the ones that are provided by the Liberty Alliance. All those Standards and their implementations can certainly become vehicles to help us build Web services federation solutions. However, the key for successfully implementing those solutions comes down to understanding the principles of Web services federation, as well as the patterns and best practices (including the correct use of Standards) that can facilitate the applicability of those principles.

Principles of Web Services Federation

Imagine that you work at a large company. For your last project, you created a set of Web services that expose some business information to multiple applications. In order to access those services, the different applications need to be authenticated against a central directory service that contains all the user and application credentials of your domain. Given that you were creating multiple services and that the different clients needed to use different credential representations, you decided to remove the dependencies between your services and the authentication mechanisms by isolating those into a separate service, called a *Security Token Service (STS)*, which can be accessed by all the client applications. Once an application authenticates to the STS, it will receive a set of security tokens that can be used as proof of its identity in order to communicate with the services. In this case, the STS is acting also as an Identity Provider (IP/STS). Figure 1 illustrates the basic architecture:



Note: Even though this architecture is not tied to any particular Standard, it can be relatively straightforward to implement it using WS- security protocols such as WS-Trust.*

Figure 1. Basic brokered-authentication pattern

As part of the authentication process, the IP/STS provides the consumer with a set of *Claims (ClaimSet)* that, fundamentally, are a set of assertions about the consumer identity such as Name, Age, or E-Mail Address, which can be used for other security tasks such as authorization and policy enforcement. Additionally, the IP/STS also issues a set of Security Tokens, which can be used to prove the identity of the consumer as well as all the assertions made on the *claims*. When a service receives a request from a consumer, it will use both the Security Tokens and the ClaimSet to validate the identity of the caller as well as to perform other tasks such as authorization and policy enforcement. This is possible, because the services trust the security information that is issued by the IP/STS. In other words, we can say that the services and the IP/STS are part of the same *Trust Domain*.

After a few months of using this solution, your company acquired a business that also provides some Web services by using its own identity provider. For agility purposes, your management would like to keep both security environments working independently; but, at the same time, they would like to leverage both sets of Web services in the next generation of applications, without having to be concerned with the intrinsic complexities of the identity providers. They want to "*federate*" both environments.

In spite of the obvious advantage of using federation for this scenario, there are some aspects that you should take into consideration to implement this solution correctly, without affecting the security boundaries of both Trust Domains. In addition to all of the security requirements of the IP/STS, you need to consider specific aspects of the federation process, such as the strategies for mapping and preserving identities across Trust Domains, safeguarding the security mechanisms that are implemented on the IP/STSs, and abstracting the federation complexities from consumers and services (see Figure 2).

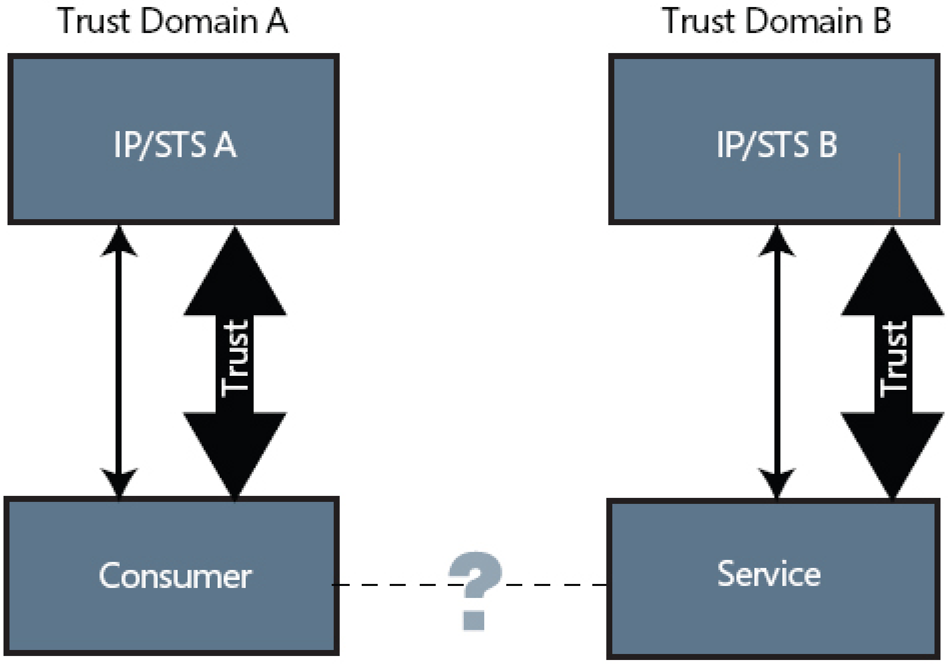


Figure 2. Initial federation challenge

While the use of standards like WS-Federation and SAML state the basic principles that are needed to implement this solution, no standard can provide a silver bullet that addresses all of the aforementioned considerations. Instead, those answers always rely on our knowledge of Web services federation patterns and techniques that can be applied to implement Web identity-federation solutions correctly.

Web Services Federation Patterns

As a result of the evolution of the Web services identity-federation architectures, the industry has produced a series of patterns, principles, and techniques that summarizes the experience that is gained through real world implementations. In this section, we will explore some of those patterns from a pragmatic standpoint, without introducing any dependencies on a particular standard or technology.

Inter-Domain Token Exchange

Context-Problem

This scenario is very similar to the one presented in the previous section in which a consumer who is living in a trust domain needs to interact with a service that is developed in a federated trust domain. The service accepts only identity information that is issued by the Identity Provider of its trust domain, which at the same time has no knowledge of the authentication mechanisms that are used by the consumer application.

Forces

This scenario presents two characteristics that justify using the solution described in this section:

- Multiple Identity Providers: For management and scalability reasons, your company needs to maintain multiple trust domains using different identity providers.
- Trust boundaries: Services should only accept security tokens issued by the identity provider of its trust domain. This will keep the security boundaries that prevent any application from interacting with the service unless it can present the proper security information.

Solution

By using some of the fundamental principles of federation and trust, we can design a model that addresses the previously explained scenario. The key to the solution is to establish a trust relationship between the two Identities Providers, so that security tokens that are issued in one trust domain can be used in the other. Figure 3 illustrates the basic model.

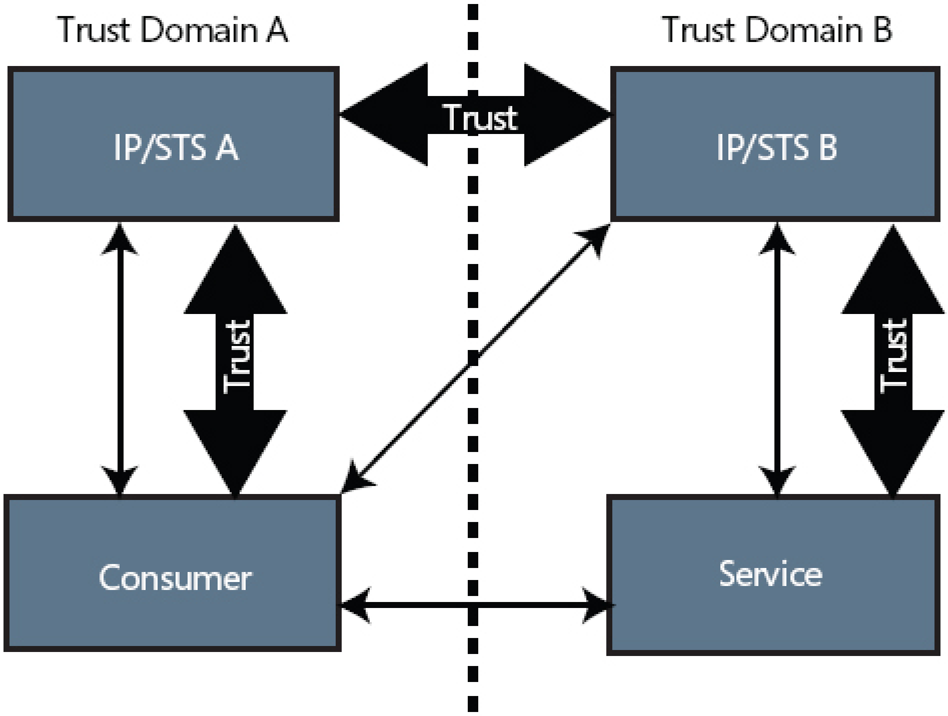


Figure 3. Basic federation-trust scenario

Initially, the consumer will communicate with IP/STS-A and express its intentions of accessing a resource on Trust Domain B. Consequently, IP/STS-A will issue a security token and a set of identity and attribute claims that need to be presented to IP/STS-B in order to obtain a new security token and identity claims that can be used to access the resource.

This approach keeps the flexibility of maintaining isolated trust domains, while introducing trust relationships between the IP/STSs to facilitate federated interactions between services and consumers across domains. Additionally, this model can be extended incrementally in order to address more complex federation scenarios.

One of the challenging aspects during the implementation of this model is the fact that it introduces a certain level of dependency between the consumer on one Trust Domain and the IP/STS on the other. For instance, if IP/STS-B is using certificates as the security token, the consumer application must be able to handle the certificates before it communicates with the service. In order to eliminate the dependencies between consumers and an IP/STS on a different security domain, we can apply a variation of this pattern, called an *intra-domain token exchange*.

Intra-Domain Token Exchange

Problem-Context

The scenario for this pattern is very similar to the one that is presented in the previous section, except that we would like to avoid any dependencies between the consumers on one Trust Domain and the IP/STS on another Trust Domain.

Forces

In addition to the forces that are listed on the previous section, this scenario presents one difference that is worth highlighting:

- Different token-claims representation: On a federated scenario, one of the IP/STSs uses a security token representation that can't be handed by the consumer applications on the federated trust domain.

Solution

The solution for this scenario is a small variation of the one that is presented in the previous section. Specifically, the consumer will present the token that is acquired from IP/STS-A to the service on Trust Domain B. The service, probably using some sort of interception mechanism, will forward the token to the IP/STS-B for validation. After that, IP/STS-B will use the mechanisms that are specified in the trust relationship to validate the security token and the claims, so that the request can be processed. Figure 4 illustrates this model:

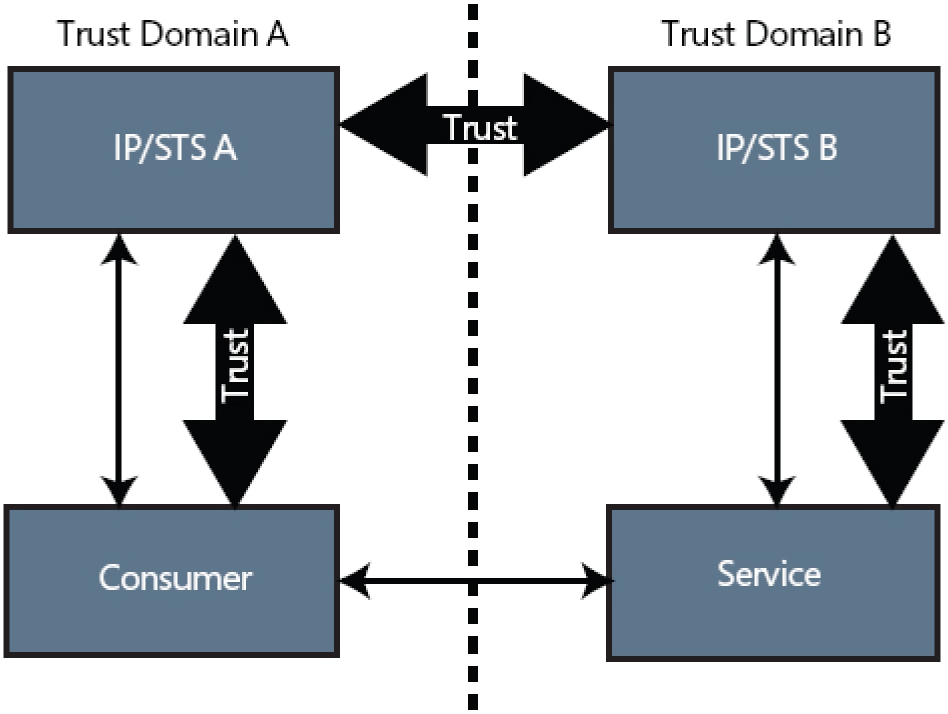


Figure 4. Intra-domain token exchange

Although this model removes the dependencies between the consumers and IP/STSs on different Trust Domains, it does introduce an extra level of complexity on the IP/STS—particularly, in the mechanism that is used to model the trust relationships with other IP/STSs, given the extra considerations that are needed for the security token and claims validations. Additionally, from a security standpoint, this model makes some concessions allowing the services on Trust Domain B to accept messages with identity information that has not been issued by IP/STS-B.

The two Web services federation patterns that we have discussed so far can address a large variety of the identity-federation scenarios that are presented in real-world applications. The main challenge of implementing those patterns relies on establishing the correct trust mechanisms between the IP/STS. Sometimes, it is not possible to establish a direct trust relationship between two trust domains. For instance, an IP/STS for the credit department of a financial institution might be issuing identity claims containing information that can't be shared with a partner company involved in the federation process. For those complex scenarios, the use of a third-party component that can model that trust relationship is often a good solution.

Third-Party Trust Establisher

Problem-Context

Sometimes, the creation of trust relationships between IP/STSs can present complexities that can derail into a tightly coupled interaction with the consequent challenges around management and versioning. For those scenarios, developers should implement a model that, although more complex, provides the flexibility that is required by the trust relationships, without affecting the default behavior of the IP/STS.

Forces

This scenario could present some specific conditions that justify the proposed solution:

- Policy change frequency: The policies that regulate the identity information that is associated with a consumer are constantly changing and, consequently, those changes can influence the policies of the second IP/STS.
- Different token-claims representation: Similar to the previous pattern, there are some scenarios in which the different IP/STSs use completely different claim representations, which can make the process of mapping the claims and validating the security tokens between those services very complex.
- Loosely coupling IP/STSs: Based on numerous reasons, one of IP/STS should not have knowledge of the security mechanisms that are used by the other IP/STS.

Solution

The typical solution for implementing a complex trust relationship between two or more IP/STSs, without introducing any dependencies, relies on establishing that trust relationship by using a third-party IP/STS to act as the bridge between the other IP/STSs. Figure 5 illustrates that concept:

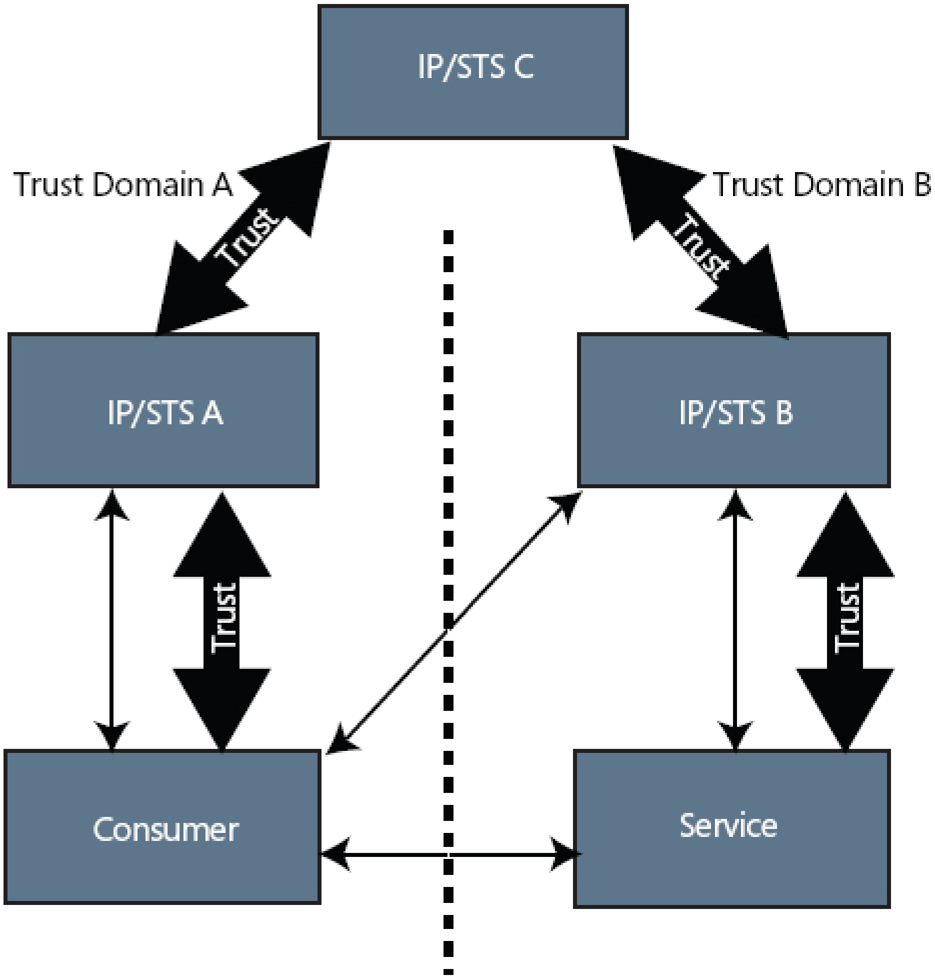


Figure 5. Third-party STS federation

In this scenario, when the consumer requests a security token to access a service in another Trust Domain, IP/STS-A will contact IP/STS-C to issue a security token that will be accepted by IP/STS-B. The security token and identity claims that are returned to the consumer will be in the native format that is used by IP/STS-A, but will contain all of the necessary information to guarantee its use on Trust Domain B. After that, this model will follow one of the token-exchange patterns; and, when the token is received by IP/STS-B, it will contact IP/STS-C to enforce the trust relationship.

One of clear advantages of this pattern is that it isolates the complexities of the federated environment from the different trust domains. As a disadvantage, this model introduces a new complex component that needs to be versioned and maintained as part of this infrastructure.

Like the first two Web services federation patterns, this one also requires propagating identities across Trust Domains. Some scenarios will require that the IP/STS in one Trust Domain should not have access to the identities of another. In this case, a pseudonym claim service can be used to map identities across federated domains.

Pseudonym Claim Service

Problem-Context

Federation scenarios quite often require identity mapping between entities that are deployed in different Trust Domains. In some scenarios, the privacy requirements dictate that the identity should not be propagated across domains.

Forces

This scenario presents several considerations that justify the solution proposed in the next section:

- Identity protection: For some scenarios, the IP/STS of a Trust Domain should not have access to the principal identity of the entities of a federated Trust Domain.

- Identity mapping: Although the identity should be protected across federated domains, these domains still need a mechanism for identity mapping.

- Identity collision: For some scenarios, the identity of services within an IP/STS can collide with other identities in a federated Trust Domain.

Solution

A *pseudonym claim service* is a service that maps principal identities to pseudonyms that can be used in the federation process with different Trust Domains. Figure 6 illustrates the concept of the pseudonym service:

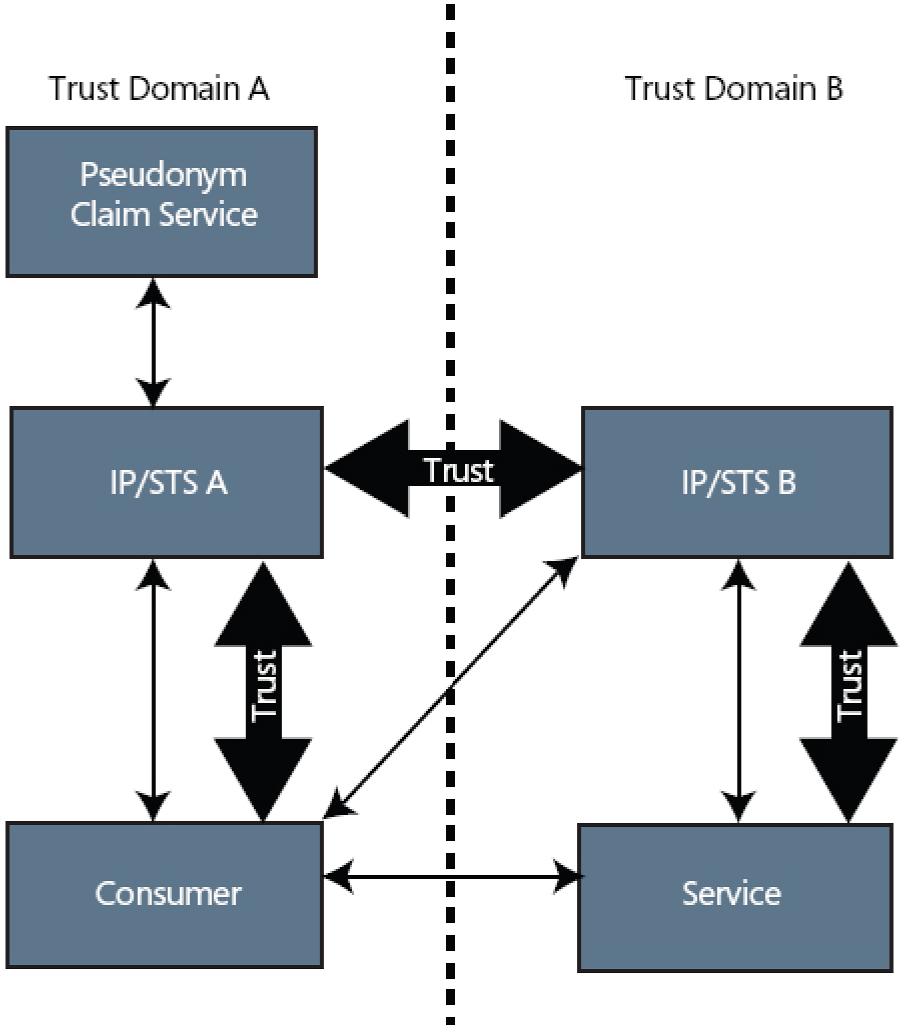


Figure 6. Federation using a pseudonym claim service

Initially, the consumer requests a security token from IP/STS-A, which, in turn, gets its identity aliases from the pseudonym service. After that, IP/STS-B generates the claims and the security token by using the consumer pseudonyms, instead of the principal identity. From there, the process can continue using any of the patterns that are explained in the previous sections.

The fact that we are using aliases instead of the principal identities preserves identities within their Trust Domains. Also, given that the aliases are applied on the initial claim set, the identity-mapping processes can be executed against the aliases without any impact on the normal process. Although this pattern is focused on pseudonyms for requestor identities, you can think of variations that use other types of identity aliases, such as service pseudonyms.

Until now, we have explored scenarios in which the security tokens and claims that are issued by the IP/STSs contain all of the information that is needed to execute federation tasks, such as identity mapping and policy enforcement. In practice, this is not always the case, and there are many scenarios in which the federation procedures need more information than what is contained in the claim sets that are issued by the IP/STS.

Attribute Claim Service

Problem-Context

Some of the federation tasks require extra information about the requestor that is not included in the security claim set that is generated by the IP/STS. For instance, the services on a specific Trust Domain require an e-mail claim that is not part of the claim set that is generated by the federated IP/STS.

Forces

Some of the following details justify the solution presented in this section:

- Privacy: Based on privacy requirements, only a subset of the consumer claims is used as part of the identity information that is issued by the IP/STS. The remaining consumer claims may be accessible to security entities such as IP/STSs, but should not be propagated to the different consumer applications.

Solution

The attributes that are needed for Web services identity federation can be exposed via an Attribute Claim Service, which can be invoked from the IP/STS and services on a Trust Domain. Using this service, either the IP/STS or the service itself can request more information about the requestor in order to complete the proper tasks. Figure 7 illustrates this concept:

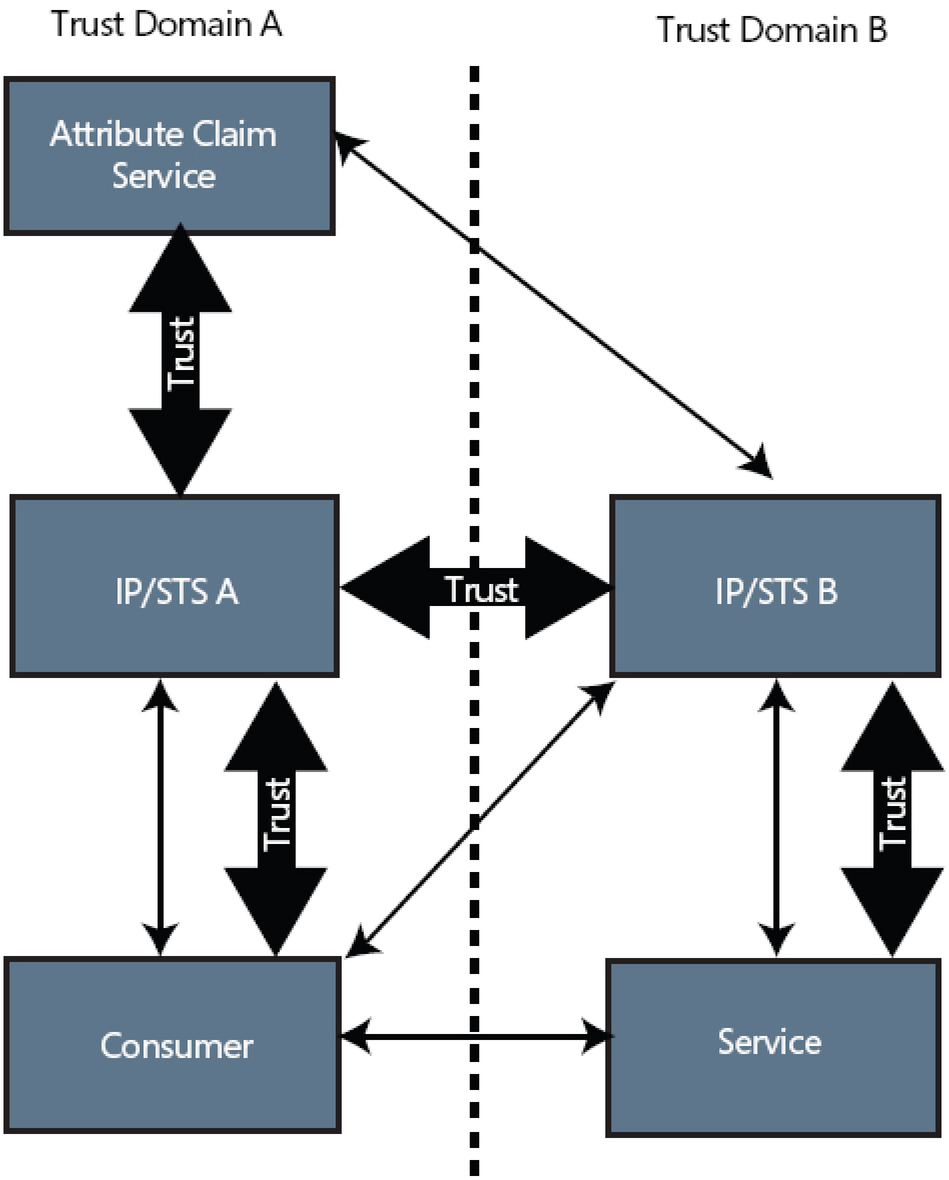


Figure 7. Federation using an attribute claim service

In this scenario, the consumer requests a security token from IP/STS-A, which issues a security token and the claim set that are required for Trust Domain A. Then, it presents that security token and claim set to an entity in Trust Domain B, which is either the service or IP/STS-B, depending on the federation pattern that we are using. In order to complete the federation process, IP/STS-B might need some extra claims that are not included as part of the claim set that is issued by IP/STS-A. In this case, IP/STS-B would obtain those claims by querying the attribute claim service, and presenting the security token that is issued by IP/STS-A and its own security identity. This access is possible, because the trust relationship that exists between IP/STS-A and IP/STS-B allows the latter to present a valid set of claims and security tokens to the attribute service. This last step is required, because the attribute claim service is typically part of the trust relationship between the two Trust Domains.

Although optional, the use of attribute claim services is becoming very popular in Web services federation scenarios. Based on its flexibility, this pattern can be applied in a number of different topologies and is occasionally combined with the pseudonym service pattern. For instance, in some scenarios, the attribute claim service and the IP/STS could be the same physical Web service, and, in other scenarios, they can be implemented as two separate Web services in order to maximize the flexibility of the solution.

Defederation

Problem-Context

Web services federation solutions are typically complex models that involve a relatively large number and variety of services, consumers, and identity providers. In that context, the governance and management of the entities that are involved in the federation is absolutely vital to the correct functioning of the model. Specially, the ability of dynamically "defederated" entities can become a big challenge in these scenarios.

Forces

The following conditions justify the solution that is proposed in this section:

- Defederating services: When a service is removed from the federation, we need the mechanisms for dynamically updating the IP/STSs and other services in the federated Trust Domains that have a relationship with that service.

- Defederating consumers: When a consumer is removed from the federation, we need the mechanisms for notifying the federated IP/STSs that the identity claims of that consumer are no longer valid in the federation.

- Updating policies: When the security policy of a service changes, we need the mechanisms for propagating those changes to the federated IP/STS.

Solution

Undoubtedly, the process of defederating entities is one of the more challenging aspects of Web services federation solutions. An interesting way to approach this problem is to provide the entities that are involved in a federation with a publish/subscribe "defederation protocol" that can be used as the main mechanism for defederating entities. For instance, when a service is removed from the federation, a "Defederate Service" message is sent to the IP/STS of its Trust Domain and, from there, propagated to the IP/STSs and other services that are involved in the federation. After that, a consumer from a different Trust Domain application will be unable to acquire a token to access that service. Figure 8 illustrates this concept:

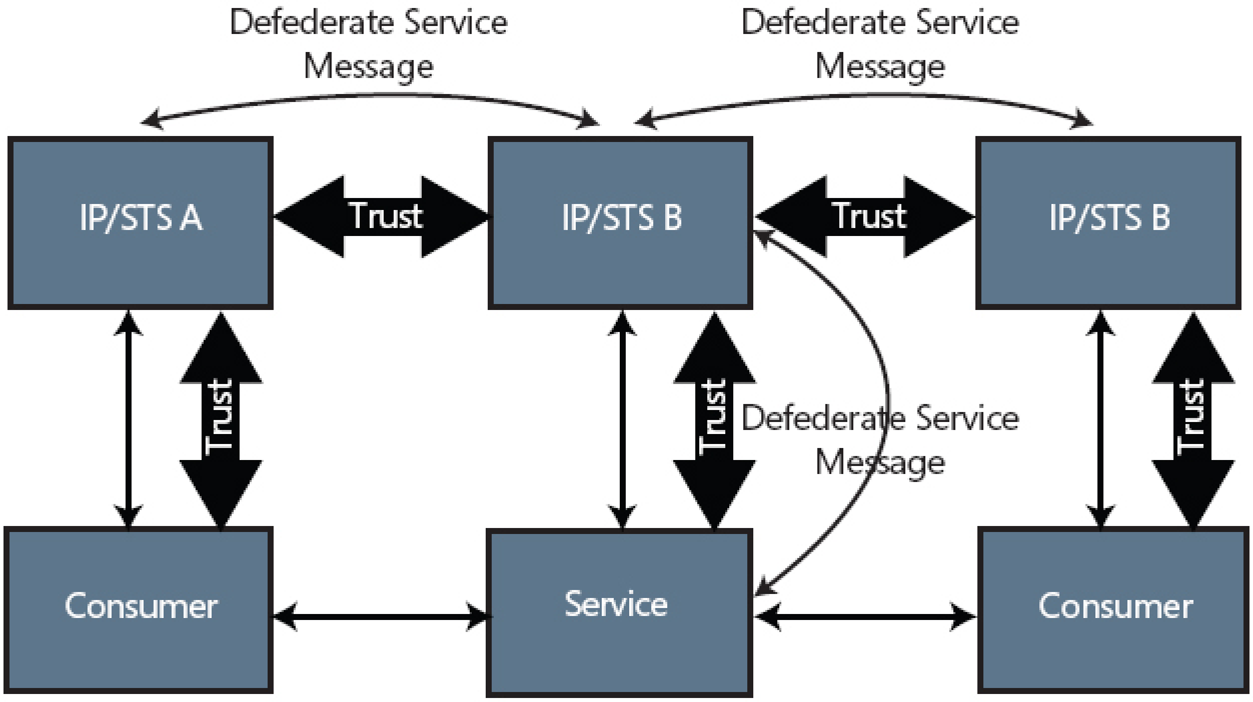

Defederate Service Message
Defederate Service Message
IP/STS A
Trust
IP/STS B
Trust
IP/STS B
Trust
Trust
Defederate Service Message
Trust
Consumer
Service
Consumer

Figure 8. Defederate service model

Indirectly, the use of a defederation protocol helps to enforce the security and trust boundaries that are enforced in the Web services federation solution. As explained in the previous section, there are multiple aspects that can be implemented with the use of a defederation protocol, without interfering with the normal functioning of the Web services federation.

Standards

During the last few years, the Web services community and software vendors have produced some standards that address some of the most common challenges of Web services federation. Although the patterns that are presented in the previous section are not dependent on a specific standard, certainly most of the practical implementations are based on some of them. The basic set of Web services federation standards are based on WS-Trust and WS-Federation. WS-Trust is a great solution for implementing brokered-authentication scenarios, as it provides mechanisms for codifying claims (assertions) about a consumer as security tokens that can be used to protect and authorize Web services requests in accordance with a policy. WS-Federation extends that model by describing how the claim transformation model that is inherent in security token exchanges can enable richer trust relationships and advanced federation of services. Although both WS-Trust and WS-Federation rely on WS-Security and, consequently, can use multiple security token representations, the use of the Security Assertion Markup Language (SAML) can considerably improve the flexibility of a Web services federation solution. SAML provides a natural way of expressing identity assertions (claims) and other metadata information that can be exchanged between the different entities on a federated environment. Finally, the Liberty Alliance project and, specifically, the ID-FF specification propose a very pragmatic approach that facilitates the implementation of federated identity scenarios. Although there is some overlap between the WS-* protocols and the Liberty Alliance project, there are lessons and best practices that can be leveraged from both in order to implement robust Web services federation solutions.

Conclusion

These patterns for solving Web services federation scenarios are based on real-world experiences. This is not an exhaustive list; these patterns solve the most common identity-federation challenges. Each pattern evolved, as the requirements for Web services federation grew more complex. Even though they have been distilled to their most basic model, they do not have to be implemented as stand-alone models. Several patterns can be combined to solve even more complex requirements. Understanding these patterns and the principles of Web services federation improves the correct use of Web services federation standards and technologies. Over time, these patterns will evolve and new patterns will emerge to keep up with the ever-changing landscape of identity federation.