Learn the discipline, pursue the art, and contribute ideas at www.ArchitectureJournal.net
Resources you can build on.

THE ARCHITECTURE JOURNAL™
Input for Better Outcomes
Journal 16

Identity and Access

The Evolving Role of the Identity

Federated Identity Patterns in a Service-Oriented World

Managing Identity Trust for Access Control

Architecture Journal Profile: Kim Cameron

Federated Identity and Healthcare

Claims and Identity: On-Premise and Cloud Solutions

Enterprise Identity Synchronization Architecture

Microsoft®

Contents

THE ARCHITECTURE JOURNAL™
Input for Better Outcomes
Journal 16

Foreword 1
by Diego Dagum

The Evolving Role of the Identity: From the Lone User to the Internet 2
by Fernando Gebara Filho
A report on how identity technologies have evolved to accommodate current needs, and what the challenges are from here to the future.

Federated Identity Patterns in a Service-Oriented World 6
by Jesus Rodriguez and Joe Klug
A sequence of strategies intended to make applications trust each other. How scenarios challenge the real success of each strategy and what can we do to address those challenges.

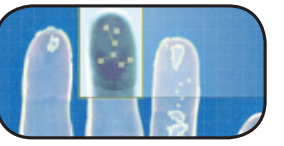
Managing Identity Trust for Access Control 12
by Gerrit J. van der Geest and Carmen de Ruijter Korver
A reference architecture for the management of Identity Trust within the context of Identity and Access Management.

Architecture Journal Profile: Kim Cameron 18
Kim Cameron is an Identity architect at Microsoft Corp. Learn what the founder of "the Laws of Identity" has to say about his career.

Federated Identity and Healthcare 20
by Mario Szpuszta
A real-world example of federated identification implemented in the Austrian National Healthcare System.

Claims and Identity: On-Premise and Cloud Solutions 26
by Vittorio Bertocci
How the lessons learned from current efforts on federated identities are determining upcoming trends in cloud-hosted applications.

Enterprise Identity Synchronization Architecture 33
by Mike Morley and Barry Lawrence
A case study on smart provisioning strategies for controlled and legacy environments.



Foreword

Dear Architect,

In the previous issue of the Journal, we explored the role of the architect across a number of dimensions. After being the editor of the Journal for 10 issues, I myself have accepted a new role leading the Platform Architecture Team here at Microsoft. I would like to introduce the new editor-in-chief of the Microsoft Architecture Journal, Diego Dagum. Diego has a long career as an architect, and is the current editor behind the MSDN architecture center. Please join me in welcoming Diego to the new role as editor-in-chief, and as always, we welcome all your feedback at editors@architecturejournal.net.

Simon Guest

Two years ago, when an article of mine about evolving architectures was published in an independent IT magazine, a colleague said to me, "You should write for *The Architecture Journal*." I couldn't have predicted that I would now find myself writing for this magazine as its editor. I want to thank Simon Guest, for this opportunity and these big shoes to fill—during his tenure, readership has more than doubled, increasing from 30,000 to 62,000+.

In this issue, we invite you to think about the identity architecture in your organization. Identity management today is evolving from the single, isolated scenario to a federated one, in ways which may surprise you.

We begin this sixteenth journey with Fernando Gebara Filho's introduction to identity concepts and strategies, how they have evolved and the road ahead. Next, Jesus Rodriguez and Joe Klug examine an assortment of strategies for making identity a first-class citizen in the portfolio of federated applications. Gerrit van der Geest and Carmen de Ruijter Korver consider the challenge of establishing an application-level trust environment since user identities, in a service-oriented world, must flow from a service consumer to a provider.

For this issue's profile, we caught up with Kim Cameron, author of "The Laws of Identity," whose ideas on federated identities are shaping the next generation of Microsoft identity technologies. (A funny thing happened the day I visited Kim for this interview: I forgot my ID badge so I needed Kim to "certify" my identity to the lobby.)

Resuming our journey, Mario Szpuszta describes how the Austrian healthcare system turned an administrative provisioning crisis into a clear opportunity for creating an open identity federation. Then Vittorio Bertocci explains how architectural patterns allow us to build claim-aware solutions, so that when the cloud arrives to companies, identity management won't necessarily look cloudy.

Finally, Mike Morley and Barry Lawrence reveal how they synchronized identities on multiple systems and legacy applications from a single administrative console through a consolidating framework.

Dear reader, I'd like to be the first to welcome you to the issue, and hope that you'll *identify* with the articles within. Enjoy!

Diego Dagum

The Evolving Role of the Identity: From the Lone User to the Internet

by Fernando Gebara Filho

Summary

The purpose of this article is to show the readers how digital identities are evolving over time, from the needs of a single user running within a single computing platform to today's world of multiplatform identity federation and privacy concerns. It can be used to help infrastructure architects design effective identity infrastructures that are not over-engineered, but rather based on current and projected needs of the organization.

There was a time when very few people needed a digital identity. They were specialized workers in a brand new profession, nowadays known as Information Technology (IT). They used a very limited set of devices, accessing a very limited set of applications. There was no online access; the only way to make the machines understand what to do was to feed them tremendous quantities of punched cards, loaded with machine-like language constructs. There were very few bad guys those days; the worst were the ones who hustled you so all those carefully ordered punched cards were spread out over the floor.

This is not the reality today. We live in a globally connected world, the vast majority of people using the technologies once reserved for scientists and hard-core engineers are now lay users, with little to no understanding of all the processes that are running inside their computers. There are unknown numbers of very bad guys who want to spy on you in order to determine your password to your banking account, so they are free to steal from you without the need for proximity. These malicious users can live far way from our homes, but it takes just a few seconds for them to track all our activities. Some of them don't even want our passwords — they want to steal our home-processing power to run complex hacking algorithms or distributed attacks to carefully chosen targets in the Internet. And so computer programmers must constantly work on digital identity technologies in order to handle all those changing landscapes.

A (very) Brief (and simplified) History of Identities

In the beginning…

… there was almost no interest in creating and managing identities and their security contexts. Why? We lived in a world of mainframes and mini-computers, submitting huge computational jobs through punched cards and printing stacks and stacks of paper on mechanical printers (but only if we were IT professionals or attending University classes at that time). Our identity was nothing more than an identifier, determining who submitted the job and who owned that big amount of paper (usually printed on the first page of the paper stack).

There was no security context at all in our identities. The username/password pair was even printed in the punched card set, so there was absolutely no secrecy involved. But there was no need for it, especially in the commercial/academic world; except for a few individuals, there was no interest in stealing other people's jobs (JCL jobs, that is). The only necessary secrets were in the realm of military installations. Identities were used only in the context of a single machine. If you wanted to use another computer, another username/password pair had to be created, and there was no connection among the identities in the machines you were allowed to use.

Basically, identities were not used to really identify you. Their only purpose was to generate an identity under which a process was run and the results could be sent to you. There was a very weak connection between you and your digital identity.

With the advent of distributed computing, network logon became a necessity, and technologies and protocols were specially created to handle those needs. But they evolved from the context of the so-called workgroup computers to the full domain-based central directory. Workgroup computers were really a set of workstations with a "master" element that took care of presenting the individual members as a cohesive entity. But this was only a view of the reality: You could enumerate workgroup members and resources but, when trying to access one of them, you had to be registered in the local identity database of the member that held the resource; and this workgroup member was responsible for checking if the credentials you used were correct.

The workgroup concept evolved alongside the network. When file and print servers became popular, they were also responsible for holding the user identity database and running the algorithms that checked the presented credentials' validity. Initially, one server was enough for daily jobs, but as quickly as we could spell "network," the need for networked servers showed up in our lives. This brought the challenge of presenting the user identity as a unique entity among all those computational resources: If I wanted to use a printer, no matter which server held the printer queue, I had to identity myself using my single set of network credentials and get the job done. In the first years of the networked servers, a simple and effective (at that time) artifact was used: identity database replication. All servers that were part of a known and trusted set of servers replicated its user database, effectively implementing the concept of a single sign-on to network

resources. Obviously, this mechanism had its limitations. When dealing with a large number of servers, replication delays and even inconsistencies were commonplace.

This may have been one of the first times when there was a clear relationship between identities and the individual who held them, because the same set of credentials (username/password) were used to access a set of network resources.

Then came the concept of the network domain. In it, a set of workstations and servers are managed under a central credential database, effectively allowing the creation of a common security context among all domain network resources and processes. In the network domain model, not only users but printers, workstations, and services are assigned a set of credentials that allow the execution of processes and communications among them. A user's credentials will only validate on a workstation that is part of the same domain.

This model also allowed the creation of trust relationships between disparate domains. With it, users that are controlled by domain A can access resources from domain B if and only if domain B is set to trust the credentials from domain A. This allowed for more flexible identity management because there was no need to replicate or clone identities from domain A to domain B if the trust relationship has been previously established.

Unfortunately, this model requires that all domains that are part of the same trust mesh use the same set of technologies, making it very difficult to share resources among loosely coupled directories or directories from different technology platforms.

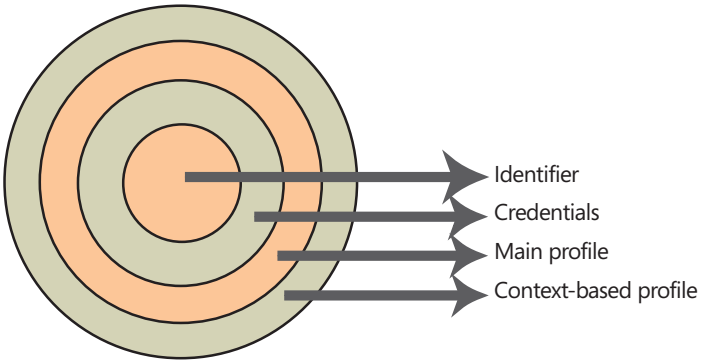
New sets of technologies were created and standardized to handle the transmission of user identities among loosely coupled network domains. They are collectively called identity federation systems: A predefined, cross-platform, standardized set of protocols designed exclusively to transmit user security contexts to allow one network domain to share resources with another network domain. These sets of standards-based protocols are friendly to the Internet infrastructure, allowing the sharing of resources even in the absence of dedicated network links.

As can be inferred from the preceding paragraphs, digital identities had to evolve from a single pair of username/password to a very complex set of protocols that transport lots of user-related claims and attributes.

A Nontechnical View of the Digital Identity Anatomy

Because of this evolution of the use and sharing of identities, a new understanding of digital identities was needed. In a simplified, nontechnical view, a digital identity can be seen as a set of at least four layers (see Figure 1):

Figure 1: Layered view of a digital identity



> **"IN THE NETWORK DOMAIN, A SET OF WORKSTATIONS AND SERVERS ARE MANAGED UNDER A CENTRAL CREDENTIAL DATABASE, EFFECTIVELY ALLOWING THE CREATION OF A COMMON SECURITY CONTEXT AMONG ALL DOMAIN NETWORK RESOURCES AND PROCESSES."**

1. Identifier;
2. Credentials;
3. Main profile;
4. Context-based profile.

In this model, the innermost layer is the unique identifier of this digital identity. There should be no identical unique identifiers in the same security domain. Users must have unique identifiers to be easily recognizable by any system to which they have access.

To have access to this identifier, the user must present to the security system a set of credentials that will be checked against the computing secrets stored in the identity database. This is the nearest we can get in terms of proof of possession. A user is entitled to access her own unique identifier if and only if she presents the correct set of credentials to unlock this magic number.

After the user unlocks the unique identifier, a set of common attributes (called the main profile) is accessible to a system that may require a little more knowledge than the unique identifier itself. This can be, for example, the user name, department, Social Security number, company name, and so on. These attributes do not vary from system to system; they are the same wherever the user logs on. They are a fixed set of values that are tied to the user during the lifetime of the logon session.

But not all systems need to share information. There are sets of information that are only meaningful in the context of a system or related systems. For example, frequent flier miles are meaningful only under the context of an airline carrier, losing most of its semantic meaning if moved from one carrier to another. But they share the same semantic context when used by the mileage program associates (restaurants, hotels, credit cards, and so on). These sets of attributes are stored in the context-based profile.

One digital identity can have only one unique identifier, a limited set of credentials (username/password pair, digital certificate/pin number pair, biometrical data), a unique set of main profile attributes, and an unlimited set of context-based profile attributes.

The separation of the unique identifier from the set of credentials used to access it allows us to evolve the security mechanisms without the need to create a different ID for the same user over time. It makes a unique identifier last for the lifetime of the individual that is the subject of the identity. This is also the base of the current identity federation systems: No matter what protocols and credentials are used to unlock the unique identifier, the same value will be presented any time it is required.

> "DIGITAL IDENTITIES FALL INTO TWO DIFFERENT, AND SOMETIMES INCOMPATIBLE, WORLDS: THE MANAGED AND THE UNMANAGED USER SPACES. MANAGED USERS ARE THE ONES TO WHOM YOU CAN APPLY CORPORATE SECURITY POLICIES DURING THE LIFETIME OF THEIR INTERACTIONS WITH YOUR NETWORK RESOURCES. UNMANAGED USERS LIVE IN A DIFFERENT WORLD."

The separation of the unique identifier from the multiple sets of profile attributes also helps us evolve the semantic and syntactic meaning of these attributes without affecting the user ID. One can use binary or XML-encoded data to store and/or present user profile attributes without interfering in the now stable unique identifier.

The Managed and the Unmanaged User Spaces

Today, digital identities fall into two different, and sometimes incompatible, worlds: the managed and the unmanaged user spaces. What are the differences between them and how can we build an effective digital identity strategy?

Managed users are the ones to whom you can apply corporate security policies during the lifetime of their interactions with your network resources. They are full- and part-time employees, vendors, trainees, and so on. They normally have signed a labor (or equivalent) contract with your company and so are subject to all corporate policies regarding working hours, safe Internet browsing, workstation and laptop security procedures, and so forth. The internal IT staff controls every device they use and is able to block the access to any network resource when they detect misbehavior. Also, the managed users are subject to full auditing on their activities and are not the owners of the data they store on local and network storage.

Unmanaged users live in a different world and have completely different security and privacy requirements. The unmanaged user is your customer or your business partner; it can be you, a managed user in your company's internal network, but a customer (unmanaged user) at an online bookseller site. Unmanaged users are the owners of the information they produce, the owners of their financial data and your IT staff has limited action, except in the case of misuse of the resources or fraudulent actions. Essentially, they use your systems when connected to the Internet and do not have direct access to the resources in your corporate network. They can imagine what your network looks like, but you have to create and manage all necessary resources (or views) that hide the internal network's topology and functional specifications from their eyes.

Unmanaged users pose another, different set of challenges than that of managed users: privacy. Privacy issues are difficult to handle and exposure of user data to theft can put your company's reputation at risk. These challenges resulted in the proposal of Kim Cameron's seven laws of identity (see http://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf for more information):

Law #1 ***User Control and Consent***
Technical identity systems must only reveal information identifying a user with the user's consent;

Law #2 ***Minimal Disclosure for a Constrained Use***
The solution that discloses the least amount of identifying information and best limits its use is the most stable long-term solution;

Law #3 ***Justifiable Parties***
Digital identity systems must be designed so the disclosure of identifying information is limited to parties having a necessary and justifiable place in a given identity *relationship*;

Law #4 ***Directed Identity***
A universal identity system must support both "omni-directional" identifiers for use by public entities and "unidirectional" identifiers for use by private entities, thus facilitating discovery while preventing unnecessary release of correlation handles;

Law #5 ***Pluralism of Operators and Technologies***
A universal identity system must channel and enable the interworking of multiple identity technologies run by multiple identity providers;

Law #6 ***Human Integration***
The universal identity metasystem must define the human user to be a component of the distributed system integrated through unambiguous human-machine communication mechanisms offering protection against identity attacks;

Law #7 ***Consistent Experience Across Contexts***
The unifying identity metasystem must guarantee its users a simple, consistent experience while enabling separation of contexts through multiple operators and technologies.

Competencies, Commoditization, and Real Value

Faced with so many choices and challenges in identity management, some users are asking: Should this be at the core of my IT services or should I delegate all these activities to a third-party? This question leads to a deeper business question: What is the value of all these identities for my company? How much time/money do I spend with identity management and what is the corresponding return on investment? In the world of Software + Services and Software as a Service, how do I create value from identity management? Is there value in outsourcing it? We must choose solutions to these problems carefully because, as expected, they are highly dependent on each company's business model and current technology infrastructure.

When drilling down the business justifications and current technology trends, one can reach the following conclusions:

1. Identity management is not a core business activity for the company;
2. There is no value that can be obtained from identity management;

"ONLY COMPANIES THAT ARE DEDICATED TO THE BUSINESS OF GENERATING AND MANAGING DIGITAL IDENTITIES (LIKE DIGITAL CERTIFICATE ISSUERS) GENERATE VALUE FROM THE TWO INNERMOST LAYERS. THE HANDLING OF THE PROCESSES OF AUTHENTICATION IS BEING COMMODITIZED, WITH LITTLE VALUE COMING FROM THEM. THE VALUE LIES IN THE INFORMATION THAT CAN BE GENERATED FROM USERS AND NOT THE PROCESSES THAT AUTHENTICATE THE USER."

3. Identity management is still a nontrivial technology so specific competencies must be nurtured inside the IT department;
4. Identities are becoming a commoditized IT function.

You have to understand how digital identities are being used by your systems and what kind of resources you are providing, allowing, or denying access to. When protecting internal network resources, it makes sense to retain the identity management competency in-house, mainly because it is being used to build and maintain strict access control lists (ACL) and is generating tons of logged data that will be used later when required by law or by internal auditing procedures. On the other side, customer access to internal resources is rare and you can get much more value from a consistent and updated user profile and historical interaction data than from the identity management activities. Customers usually have access only to application-generated views of data and business logic, and do not interact directly with internal IT data or equipment.

There is another aspect when dealing with customer-related data. Some countries restrict the storage location of their citizens to in-country hosted servers, demanding that you build and manage a local server infrastructure to handle their data and associated tasks.

This brings us back to the digital identity anatomy presented earlier. Companies can extract more value from the information contained in the main and context-based profiles than from the identifier and credentials layers. Only companies that are dedicated to the business of generating and managing digital identities (like digital certificate issuers) generate value from the two innermost layers. The handling of the processes of authentication is being commoditized, with little value coming from them. The value lies in the information that can be generated from users and not the processes that authenticate the user.

The Challenges of Identity Strategies

If you are building a new digital identity strategy for your company, first think about the audiences you will be dealing with and the infrastructure your company will be using.

There are two audiences you will be exposed to: internal users and external customers. You cannot get rid of your internal users: They will be the ones that will help the company be successful, the ones that will create value for the business. They are part- and full-time employees, vendors, and trainees. Depending on your business, you may not have a significant number of external online customers. Customers will always exist, but they may not always be represented by digital identities and may not be worth tracking online. But if you are part of the IT team of an online bookseller, for example, you will have to think about how you will handle the IDs for external customers.

For internal users, you may build an identity management infrastructure to handle all authentication, authorization, and internal network resource accesses, providing the correct auditing functions for future behavior and fraud analysis. But, if your company is a start-up and wants to minimize the costs of building and maintaining its own infrastructure (and if the regulations allow), you will probably use cloud services to host your computing needs. If so, use either state-of-the-art identity management systems that are able to federate with external systems or identity cloud services (like Windows Live ID) to handle all your internal users needs.

For external users, I tend to recommend the use of identity cloud services for managing IDs. As explained earlier, the value lies in the main and context-based profiles and not in the identifier and credential layers. You would have to build an infrastructure big enough to handle current and future customer audiences, which can lead to a big infrastructure dedicated to customer identity management. Also, think about the benefits of attracting more users because you are using a system that already has millions of preconfigured users that will not have to register a new set of credentials and will not have to learn new procedures for authentication.

Conclusion

Digital identities and their use are still evolving, based on the evolution of online services provided by the ubiquity of the Internet. Identity management is no longer merely a set of procedures for authentication, authorization, and provisioning of user accounts. If you understand how a digital identity works and how the protocols are evolving over time, you will be able to build a much stronger and lasting identity architecture for your company. Today, building a new identity architecture means working with systems and protocols that allow identity federation, further enhancing the use of your internal and/or external identities.

About the Author

Fernando Gebara Filho is an infrastructure architect in the Development and Platform team at Microsoft Brazil. He joined Microsoft in 1999 as an infrastructure consultant at the local Microsoft Consulting Services team, where he specialized in Active Directory and Exchange projects. In 2004, Fernando joined the Development and Platform group in the infrastructure architecture role, where he had a chance to get more involved in the Identity Management space and get a more agnostic view of the subject.

Federated Identity Patterns in a Service-Oriented World

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.

Introduction

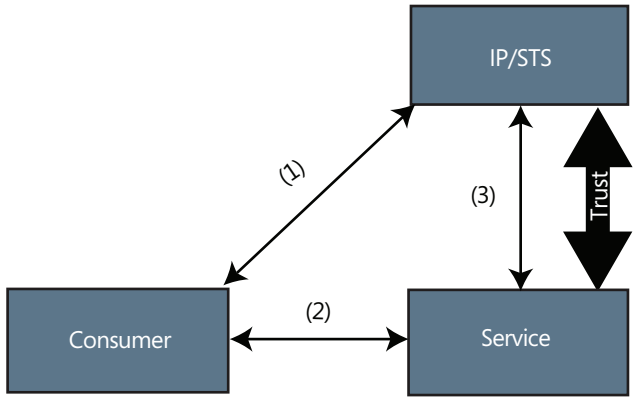
The goal of identity federation is to enable resource access across completely unrelated security domains by sharing a limited amount of information such as security identities and policies. When implemented as an architectural style, identity federation is often positioned as an alternative when identity 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 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 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

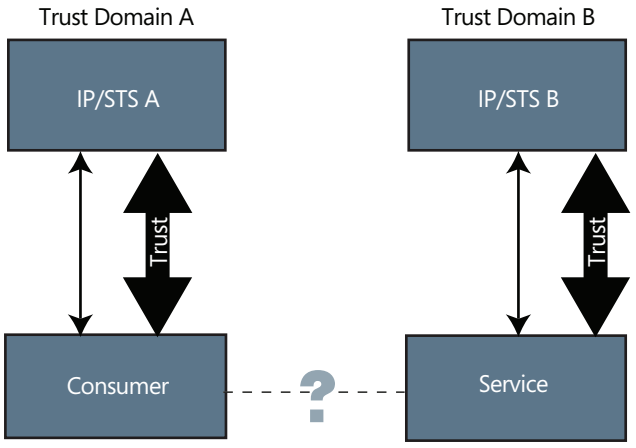
Imagine you work at a large company. For your last project, you created a set of Web Services that expose some business information to multiple applications. In order to access those services, the different applications need to be authenticated against a central directory service that contains all the 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

Figure 1: Basic brokered authentication pattern



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 2: Initial federation challenge



its identity in order to communicate with the services. In this case, the STS is also acting as an *Identity Provider (IP/STS)*. Figure 1 illustrates the basic architecture.

As part of the authentication process, the IP/STS provides the consumer with a set of *Claims (ClaimSet)* that are, fundamentally, a set of assertions about the consumer identity such as Name, Age, or Email 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 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 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 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 implemented on the IP/STSs, and abstracting the federation complexities from consumers and services (see Figure 2).

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

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 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 living in a trust domain needs to interact with a service developed in a federated trust domain. The service only accepts identity information issued by the Identity Provider of its trust domain, which at the same time has no knowledge of the authentication mechanisms used by the consumer application.

Forces

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

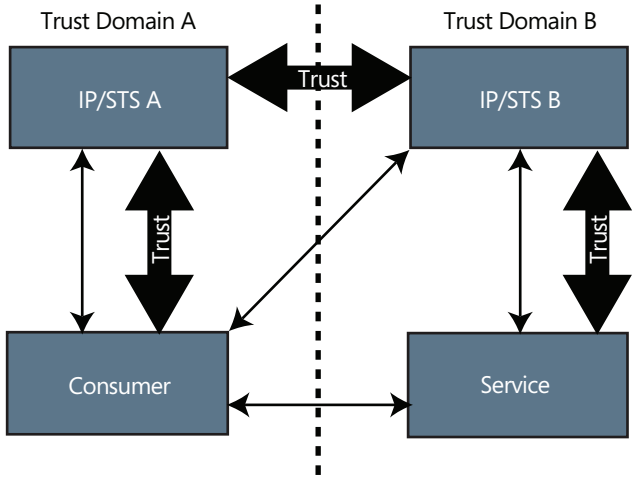
- *Multiple Identity Providers:* For management and scalability reasons, your company needs to maintain multiple trust domains using different identity providers.
- *Trust boundaries*: Services should only accept security tokens issued by the identity provider of its trust domain. This will keep the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

Solution

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

Initially, the consumer will communicate with IP/STS-A and express its intentions of accessing a resource on Trust Domain B. Consequently, IP/

Figure 3: Basic federation-trust scenario



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 incrementally extended in order to address more complex federation scenarios.

One of the challenging aspects while implementing 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 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

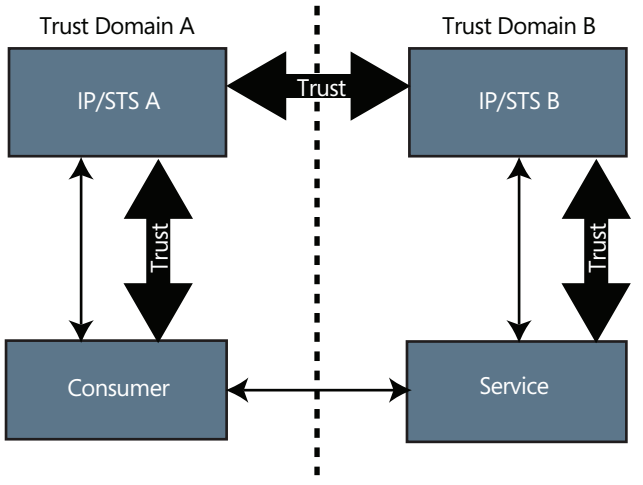
In addition to the forces listed on the previous section, this scenario presents one difference that is worth highlighting:

- *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 presented in the previous section. Specifically, the consumer will present the token 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 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 used to model the trust relationships with other IP/STSs given the extra considerations 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 discussed so far can address a large variety of the identity federation scenarios 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 may 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 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 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 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 using a third-party IP/STS to act as the bridge between the other IP/STSs. Figure 5 illustrates that concept.

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

Figure 5: Third-party STS federation

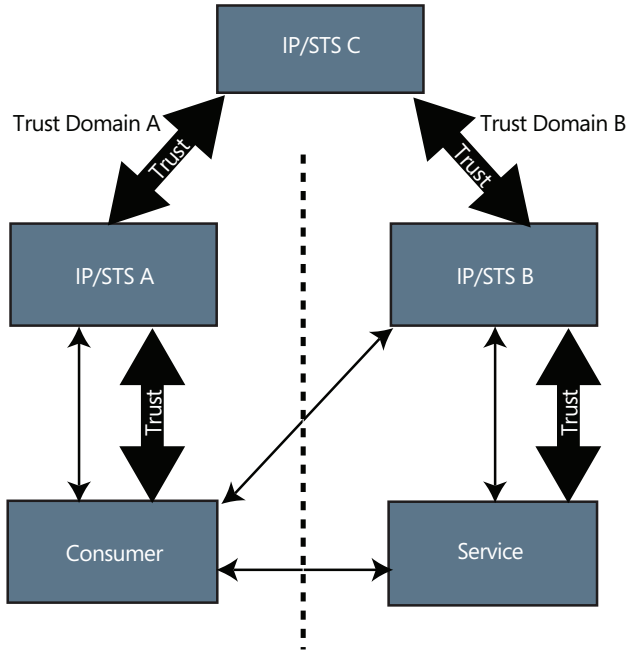
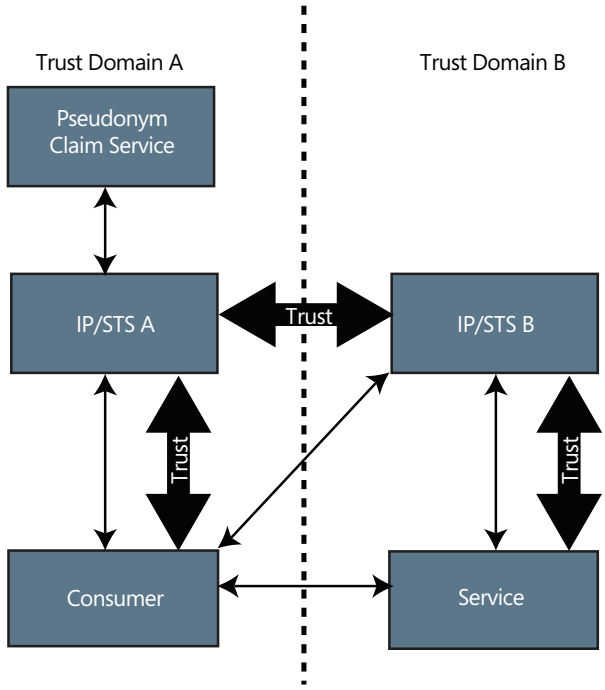


Figure 6: Federation using a pseudonym claim service



a security token that will be accepted by IP/STS-B. The security token and identity claims returned to the consumer will be in the native format used by IP/STS-A, but will contain all 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 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

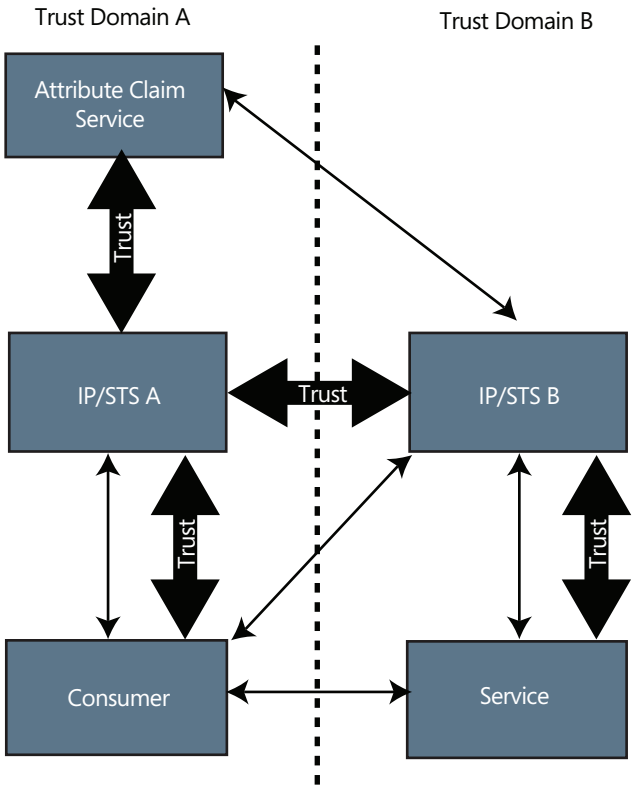
A pseudonym claim service is a service that maps principal identities to pseudonyms that can be used in the federation process with different Trust Domains. Figure 6 illustrates the concept of the pseudonym service.

Initially, the consumer requests a security token from IP/STS-A, which in turns gets its identity aliases from the pseudonym service. After that, IP/STS-B generates the claims and the security token using the consumer pseudonyms instead of the principal identity. From there, the process can continue using any of the patterns 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 issued by the IP/STSs contain all the information 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 issued by the IP/STS.

Figure 7: Federation using an attribute claim service



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 generated by the IP/STS. For instance, the services on a specific Trust Domain require an Email claim that is not part of the claim set 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 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 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.

In this scenario, the consumer requests a security token from IP/STS-A, which issues a security token and the claim set 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 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 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 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 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 proposed in this section:

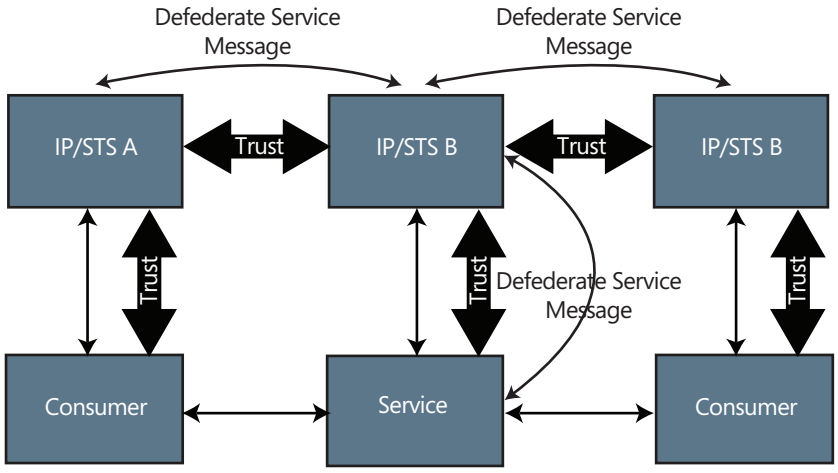
- *Defederating services:* When a service is removed from the federation, we need the mechanisms for dynamically updating the IP/STSs and other services in the federated Trust Domains that have a relationship with that service.
- *Defederating consumers:* When a consumer is removed from the federation, 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 most challenging aspects of Web Services federation solutions. An interesting way to approach this problem is to provide the entities 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 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.

Indirectly, the use of a defederation protocol helps to enforce the security and trust boundaries 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.

Figure 8: Defederate service model



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 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 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 standalone 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.

About the Authors

Jesus Rodriguez is the Chief Architect of Tellago Inc. He is also a Microsoft BizTalk Server MVP, an Oracle ACE, and one of a few Architects worldwide to be a member of the Microsoft Connected Systems Advisor team. As a member, Jesus has been selected to participate in a variety of Software Design Reviews with Microsoft's Product Teams including Windows Communication Foundation, Windows Workflow Foundation, and BizTalk Server.

Jesus derived his extensive experience with business process integration and messaging through numerous implementations of disparate systems founded on the principles of SOA and BPM. Jesus is an active contributor to the .NET and J2EE communities and an internationally recognized speaker and author with contributions that include several articles for various publications including *MSDN Magazine, Microsoft Architecture Journal, SOAWorld, and Web Services Journal* as well as speaking engagements at top industry conferences such as Microsoft TechEd, SOAWorld, Microsoft SOA and BPM Conference, Oracle Open World, Web Services Security Conference, and the Microsoft MVP Summit. Additionally, Jesus has conducted a number of Web Casts on varying SOA technologies.

Jesus is a prolific blogger on all subjects related to SOA and has a true passion for technology. You can gain valuable insight on leading edge technologies through his blog at http://weblogs.asp.net/gsusx.

Joe Klug is the Chief Technical Officer at Tellago Inc. Joe has an extensive background in enterprise application integration and business process management, which he derived though practical consulting engagements and enterprise-scale software development.

Prior to Tellago, he worked as a Product Manager in the BizTalk Server team at Microsoft Corporation. During his six years at Microsoft, Joe was a key contributor on several releases of BizTalk Server. Some of his key areas of responsibility include Web Services integration and adapter development, and he was one of the founders of the software development kit for developing Windows Communication Foundation-based adapters.

While at Microsoft, Joe became a nationally recognized speaker through engagements at Microsoft conferences including TechEd, SOA & BPM Conference, Office Developers Conference, and SharePoint Conference. He also worked as an interoperability consultant at J.D. Edwards, focusing on integrating J.D. Edwards OneWorld software to other third-party applications, mainframe solutions, and e-commerce gateways. Joe received a Bachelor of Science in Computer Science from Michigan Technological University and an M.B.A. from the University of Colorado.

Joe shares his insights on technology through his blog at http://geekswithblogs.net/jklug/Default.aspx.

Managing Identity Trust for Access Control

by Gerrit J. van der Geest and Carmen de Ruijter Korver

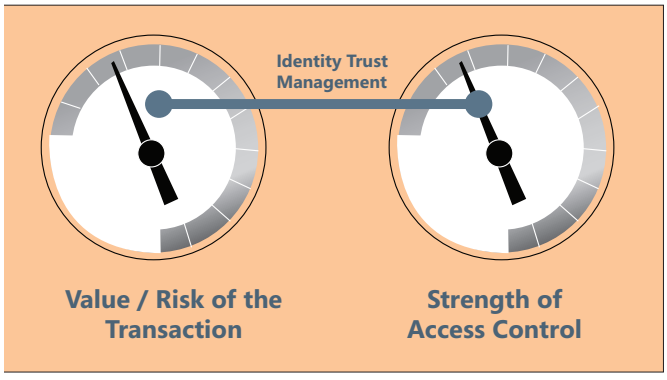
Summary

In the past, systems were used by a small set of users, mostly within the trusted circle of their organization. The circle of trust has, however, widened considerably. Today's computer systems are used by a variety of users across many organizations and geographical areas and via different channels. Organizations must make their information assets available to many users but simultaneously protect these against unauthorized access. However, each transaction that is used to access those information assets should bear just the appropriate level of access control, not too little in view of the security risks and not too much in view of user friendliness and high costs. How can we achieve an optimal balance between availability and protection?

The answer lies in the management of identity trust. We need to be able to create, maintain, and communicate various levels of trust. When we can manage identity trust effectively, we can subsequently tune the level of appropriate access control to the level of value or risk that is associated with the transactions (see Figure 1). An organization should therefore ensure that its strategic Identity and Access Management (IAM) architecture provides for mechanisms to capture and transport identity trust throughout the service layers within its IT infrastructure.

This article describes the management of identity trust as defined in an IAM reference architecture. It supports an organization's business requirements, which can range from providing low-threshold access for registration to performing high-risk financial transactions for high volumes of consumers. We'll introduce the concepts of identification trust, authentication trust, reputation trust, session trust, and trust level up- and downgrades. Additionally, we'll discuss how to implement such a model as part of a Service Oriented Architecture (SOA) by making use of the available industry standards.

Figure 1: Identity Trust Management



Challenges

Organizations face a number of challenges with respect to their Access Control:

Scalability. The e-business portfolio of organizations will strongly expand and new e-business services that require a high level of protection against fraudulent activities will have to be supported. This requires IAM functionality that is scalable from a quantity perspective and, even more important, from a security perspective.

Cost reduction. The implementation and management of adequate IAM functionality is a costly exercise. Organizations aim for sharing IAM functionality between their businesses.

Federation. Organizations will enter into more partnerships in which the partners may take responsibility for part of the IAM services based on a well-established trust relationship between the organization and these partners. On the other hand, identification processes are expensive and organizations that have implemented these processes can benefit by providing services to other organizations.

Client-centricity. Organizations want to be considered "easy to do business" with. Concepts like client-centricity are therefore important. Even organizations comprising extremely loosely coupled businesses and their partners must offer cross-business product portfolios to their consumers. Many cases require a group portal strategy for this purpose. Such a strategy can only succeed when it is supported by IAM services, which ensure that identity and authentication information can be exchanged between businesses without compromising their autonomy.

SOA compliance. Within an SOA architecture, the business processes are enabled via loosely coupled, coarse-grained, and reusable business services. The business services are exposed via technical interfaces based on industry open standards. The reusability of the services can only be achieved when appropriate controls are in place for protecting and ensuring service availability. Access Control information must

be communicated between service consumers, service providers, and between all enterprise components that make up the services. Traditionally, Access Control functionality was part of and integrated within several IT infrastructure components like operating systems, applications, and network components; the challenge will now be to implement this functionality as shared IAM services integrated within the Enterprise Service Bus.

Trust, Trust, Trust...

Only effective communication of IAM information such as identities, authentication assertions, or access policy decisions can address all of the challenges mentioned so far. Communication of IAM information requires the establishment of *trust* between the parties involved.

As described by the International Telecommunications Union, an entity can be said to "trust" a second entity when it (the first entity) has reason to assume that the second entity will behave exactly as the first entity expects.

Trust relationships are sometimes obscured but in essence exchange of information (not data) can only succeed based on some form of direct or indirect trust between the communicating parties. This applies in particular to the IAM information that is used for the protection of the resources, the assets of the organization. Transactions in IT systems, whether business, IAM, or technical transactions, represent some value or risk for the organization. Compromising a transaction may result in illegal access to a protected resource, which may lead to financial, reputation, or other types of losses.

In developing strategic IAM architectures, it became clear that we must formalize the trust concept in order to address the various challenges. In our model, the behavior as described in the definition is determined by policies established between the two entities. The level to which the receiver is able to trust the information received is dependent upon:

- the correctness of the way the policy was executed, and
- the policy that was agreed upon to generate the information.

Our Trust Model captures these two items by means of so-called Trust Levels. These Trust Levels are quantifiable and verifiable and can therefore be communicated between parties. The model requires an established governance framework.

Does Absolute Trust Exist?

This question should probably be answered by philosophers, but in the reference frame of Access Control, the answer is no. However, the absence of absolute trust does not bother us because we don't need it. In fact, we do not want it because of the disadvantages associated with obtaining higher levels of trust such as additional costs and inconvenience for users.

IAM's main objective is to provide Access Control to protected resources. The level of Access Control must be commensurate with the level of value/risk of the transactions executed against the protected resource. A level of protection that is too stringent for the type of transaction may result in high costs or bad user experience. A level of protection that is too low will increase the risks associated with the transactions. In other words, *we need to be able to create, maintain, and communicate various levels of Trust in order to tune the level of Access Control to the value/risk of the IT transactions involved.*

Trust Model

The IAM reference architecture includes a model that allows for such required commensuration of the level of Access Control to the value/risk of the IT transactions (see Figure 2). The core of this Trust Model consists of:

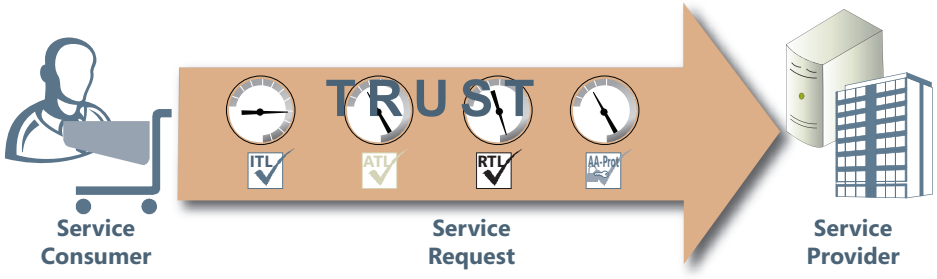
- A classification of the *IT transactions into value/risk categories* (four to six categories are often sufficient).
- Definition of *four IAM parameters* and appropriate levels to capture and communicate Trust:
  1. *Identification Trust Level (ITL)*
  2. *Authentication Trust Level (ATL)*
  3. *Reputation Trust Level (RTL)*
  4. *Level of protection of the Authentication Assertion.*
- *Associations* of the value/risk transaction categories with the parameter levels for the various Subject Types and Roles.
- Definition of mechanisms to *communicate* the parameter levels between the various infrastructure components involved in the execution of the IT transactions; and definition of mechanisms to *enforce* the required parameter levels by the resource in order to grant access.

Let's take a closer look at the four IAM parameters.

Parameter 1 - Identification Trust Level. To explain the ITL parameter, we'll first define some terminology. The terms are capitalized.

- ***Subject.*** Subjects access protected Resources. A Subject is a person, computer, device, service, or other entity that has, or will be provided, access to Resources, or has accessed Resources in such a way that audit records still have to be kept.
- ***Subject roles.*** In accessing Resources, a Subject can take on different Subject Roles. For example, a Subject of type 'Person' can assume the Subject Role of 'Personnel' but also the Subject Role of a 'Client'. In other words, employees can act on behalf of the organization in doing their daily work or act on behalf of themselves as clients of the organization procuring products or services.
- ***Digital identity.*** For a Subject to access a Resource, it must claim a Digital Identity (of type Security Principal). A Digital

Figure 2: Trust model



"IDENTIFICATION IS THE PROCESS OF VERIFYING THE CLAIMED IDENTITY OF A SUBJECT AND ASSIGNING A DIGITAL IDENTITY TO THE SUBJECT."

Identity is the representation of a Subject in the digital realm. A Digital Identity is a collection of related electronic data that represents the Assertions made by an Authority about the Subject. The Digital Identity is intended for use as a proxy of the Subject. There are various types of Digital Identities.

- ***Identification.*** Identification is the process of verifying (at a certain level of surety) the claimed Identity of a Subject and assigning a Digital Identity to the Subject. The Subject receives Credentials. Parties establish a contract (Identification Contract) during Identification.

An Identification Authority or Identification Agent performs the Identification. The Identification Authority applies an Identification Policy in order to identify a Subject. The Identification Policy dictates the steps that the Authority needs to perform. A typical example of such steps are: the Authority physically meets the Subject, verifies his or her passport, and makes a copy (please note that there is also a trust relationship here with the Authority that issued the passport). The Identification Policy also defines the way the Credentials are distributed to the Subject. This is a strong Identification Policy that may be applicable to high-risk business transactions. For a low-risk business transaction, another Identification Policy may be applicable that only prescribes that the Subject must provide a valid e-mail address and that the Credentials are sent to that e-mail address. Typical examples of Identification Authorities are a PKI Certificate Authority, an authorized human resources employee, front desk employee, or even automated processes.

Identification processes are expensive and may require manual interaction and involvement of third parties. They may also require involvement of the clients and tend to be experienced as unfriendly. So define and, if possible share, the most optimal process (do just enough) within or across organizations.

The strength of the Identification is mainly related to the strength of the verification steps (policy) performed by the Identification Authority. This strength is reflected as so-called Identification Trust Level and represents the level of assurance in the authenticity and integrity of a Subject's claimed (legal) identity (the assurance that the Digital Identity represents the Subject).

The ITL serves multiple purposes. In addition to the role it plays in Access Control, it is required for identity associations. Organizations often have multiple Digital Identities for the same Subject. In many cases, this is an undesirable situation. The processes to associate these Digital Identities — initiated by the organization but preferably driven by the Subject — will rely on the ITLs assigned to the various Digital Identities.

Parameter 2 - Authentication Trust Level. Authentication verifies and confirms a Subject's asserted Digital Identity with a specified or understood level of confidence. Credentials issued as a result of Identification are used as proof that the Subject has the right to claim the Digital Identity.

There are various strength levels for the Credentials, such as single factor username/password and more stringent forms like multifactor username/password combined with smartcards or biometrical traits. The Authentication strength is also dependent on the controls applied when establishing the session with the Subject and aspects like the channel, device type, location, and time also influence the strength of the Authentication.

The ATL represents the current session's Authentication strength. The Authentication Authority (which was involved in the verification of the provided Credentials) determines the ATL.

Parameter 3 - Reputation Trust Level. Reputation Trust represents a party's expectation that another party will behave as assumed, based upon past experience. Reputation Trust is bidirectional and can be split into Consumer Reputation Trust and Provider Reputation Trust.

The type of transactions performed, authentication history, and so on, can influence the Consumer Reputation Trust. The Consumer Reputation Trust is a dynamic parameter and can vary within a session. Consumer Reputation Trust will become an important parameter, and provisions in the IAM architecture capture, maintain, and communicate the RTLs; however, the exact definition is dependent on the specific situation and still subject to further investigation.

A client performing an e-business transaction will also implicitly establish a Provider Reputation Trust regarding the e-business provider. An e-business provider must strive towards a situation in which the Provider Reputation Trust as perceived by the client, commensurates the value/risk level of the transaction; for example, by the use of Extended Validation SSL Certificates and personalized welcome messages for higher risk transactions.

In this article, we focus on the consumer side of the Reputation Trust. The three parameters discussed so far (ITL, ATL, and RTL) need to be communicated between the service consumer and provider. Parameter 4 guarantees their integrity and authenticity at an appropriate level.

Parameter 4 - Protection of the Authentication Assertion. A protected Resource or Service Provider needs to receive an Authentication Assertion (security token), which allows it to enforce Access Control policies to determine if and to what extend it will grant access to its services. The enforcement of the Access Control policies may be delegated to a 'centralized' Policy Enforcement Point (PEP) supported by Policy Decision Points (PDP).

In the model, the Authentication Assertion contains Identity information combined with the ITL, ATL, and RTL and possibly other attributes such as privilege attribute certificates, to enable subsequent authorizations. The Identity information contained within the Authentication Assertion may reference various types of Digital Identities:

- ***Persistent identity.*** The Digital Identity originally claimed by the Subject at Authentication (also referred to as 'proclaimed identity').
- ***Implied identity.*** The Digital Identity used by intermediate services to access 'lower level' services in a trusted subsystem model.
- ***Initiating identity.*** A Digital Identity representing the person initiating the transaction. The need for such an identity is apparent in the following scenario: A client instructs a service desk employee to perform a transaction against his/her account. The persistent identity

represents the service desk employee. However, the identity of the client also needs to be kept and communicated. This identity is called the 'initiating identity'.
- ***Domain identity***. A Digital Identity associated with the Subject within a dedicated application or business domain.

The Authentication Assertion requires protection because it contains information that must not be compromised while stored or in transport. The authenticity and integrity of the Authentication Assertion is crucial, while the confidentiality of the Authentication Assertion is in most cases less important, regardless the fact that integrity may be enhanced by applying confidentiality (security by obscurity).

Protection of the Authentication Assertion is a considerable cost factor due to the required processing needed for encrypt and decrypt technologies. Therefore, the protection should be tuned to the value/risk levels of the transactions. In some cases, an SSL/TLS or IPsec transport layer security will suffice, while other situations require message layer security by applying WS-Security with various encryption levels.

The IAM reference architecture defines three levels for the protection strength of the Authentication Assertions. The service provider needs to enforce, by means of its policies, that the required level has been applied.

Associating Value/Risk Transaction Levels with the Parameter Levels

After they have been defined, it is possibe to associate the various transaction value/risk levels and parameter levels. This needs to be done for each Subject Type and Subject Role. For example, the mapping for a Subject of Type 'Person' in Role 'Personnel' will differ from a Subject of Type 'Person' in Subject Role 'Client'. Their identification policies, the way they authenticate, and the transaction types, may all differ. It is also necessary and possible to include transactions in the model that are not directly related to the core business of the organization. This concerns transactions such as IT administrator transactions and IAM transactions for Subjects in Role 'Personnel'.

Figure 3 shows how you can associate the transactions to the ITL, ATL, RTL and Authentication Assertion Protection Levels for the Subject Role 'Client'.

In order to execute this specific transaction in the value/risk category 'Medium,' the following minimum parameters must be met:

- ITL C30 stands for: an Identification Policy that includes documented proof, copy of passport, Credential distribution by out-of-band mechanism.
- ATL 4 stands for: username and strong password combined with one-time PIN via cellphone.
- RTL 50 stands for: no incidents, no history yet.
- Authentication Assertion Protection level B stands for: transport layer security, AES 256 bits encryption.

Before we discuss how the Trust Model should be implemented, we must introduce the concept of Trust Level upgrades and downgrades.

Trust Level Upgrades and Downgrades

A protected Resource will enforce an Access Control Policy, which dictates that certain minimum levels of ITL, ATL, RTL, and Authentication Assertion protection must be met. If the levels are not appropriate, it rejects the service call and requires higher Trust Levels. This triggers a process requiring additional Credentials such as a PIN. This is called an ATL upgrade. Some IAM product suites support ATL upgrades.

ITL up- and downgrades are also possible. A typical example of an ITL upgrade is when an existing e-business client enrolls for higher risk transactions and is required to come to the office and provide a copy of his/her passport. The security principal that he/she uses will stay the same but it gets a higher ITL assigned. An ITL downgrade may occur when an Identification renewal process was not executed in time, or when some of the verifications executed during the initial Identification are no longer sufficient.

An RTL upgrade may happen when there is regular use and no incidents are reported over a certain time period. A (temporary) RTL downgrade may be the result of the fact that the user has lost part of his/her Credentials.

Implementation

The ITL, ATL, and RTL are combined into what is referred to as the *Session Trust object*. Listing 1 shows the structure (see page 16).

Session Trust is the level of surety that the Digital Identity wanting to transact actually originates from the Subject, whose identity information is linked to the Digital Identity, and that this Subject will behave in the agreed manner.

The Session Trust object must be communicated as part of service requests and forms part of the Security Token embedded within these requests. There are various alternatives, but consider the implementation in a Web Services environment in which the Authentication Assertion is formatted as an SAML Authentication Assertion (SAML element <AuthnStatement>). The <AuthnStatement> element provides a means to capture the Session Trust object via its definition of the Authentication Context (<AuthnContext>) element.

According to the SAML standard: *"A particular authentication context declaration defined in this specification will capture characteristics of the processes, procedures, and mechanisms by which the authentication authority verified the subject before issuing an identity, protects the secrets on which subsequent authentications are based, and the mechanisms used for this authentication."*

The <AuthnContext> element is extensible and provides the rudimentary structure to capture Session Trust. The

Figure 3: Associating value/risk transaction levels with the parameter levels

IT Transaction	Value/ Risk	ITL	ATL (Credential value only)	RTL	AuthN Assertion Protection
...	...	...	...	...	...
Transfer funds from current to savings account	**Medium**	**C30**	**4**	**50**	**B**
...	...	...	...	...	...

Listing 1: Structure of the Session Trust object

```
<Session Trust>
        <IdentificationTrust>nnn</IdentificationTrust>
        <AuthenticationTrust>
            <TypeOfCredentials>nnn</TypeOfCredentials>
            <Channel>nnn</Channel>
            <DeviceType>nnn</DeviceType>
            <Location>nnn</Location>
        </AuthenticationTrust>
        <ReputationTrust>nnn</ReputationTrust>
</Session Trust>
```

architecture prescribes an extension of this structure to cater to the full Session Trust object.

Suppose parameter 4 indicates that the Authentication Assertion must be protected by message layer protection. In that case, use the WS-Security standard and place the SAML Authentication Assertion in a <wsse:Security> element of the SOAP header. To meet the required authenticity and integrity requirements, the issuer or attesting entity will sign the Authentication Assertion and the encryption algorithm and key length must meet the complexity as defined for parameter 4.

The Service Provider needs to implement (WS-Policy) service policies to enforce compliance of the four parameter values to the IT transaction value/risk levels of the services that it provides. It will validate the ITL, ATL, and RTL and will only accept service requests that meet the required level of protection of the contained Authentication Assertion. If these levels are not met, it will inform the service consumer, which may initiate a Trust Level upgrade procedure to meet the required levels. The Service Provider may delegate some of this verification to a Security Token Service (STS), which may also cater to Security Token transformations based on its established trust relationships with Authentication Authorities.

"IT ALL BOILS DOWN TO THE WILLINGNESS OF THE PARTIES INVOLVED TO COOPERATE, TO ESTABLISH FORMAL AGREEMENTS FOR THIS COOPERATION, AND TO ADHERE TO THOSE AGREEMENTS."

The Session Trust object contains consolidated information about the executed Identification and Authentication process and changes that occurred in the Reputation. The information contained in the object is condensed in such a way to enable communication in various technology sets and protocols. Web Access environments may require implementation of the Session Trust object within the HTTP header, which has its length limitations.

More elaborated Access Control mechanisms need additional information on top of the Session Trust object. The IAM services maintain a so-called *Trace object*, which reflects all the detailed information about the performed Identification and Authentication process steps and Reputation changes.

Listing 2 provides a possible structure for the Trace object.

The Identity Services will maintain the first and last part, while the Authentication Authorities will cater to the second part. The Trace object also plays an important role in auditing the IAM processes.

Note that the object caters to multiple Identification and Authentication steps and Reputation changes to register Trust Level upgrades and downgrades. The Trace object is associated with the unique identifier assigned to the Digital Identity as maintained by the Identity Services.

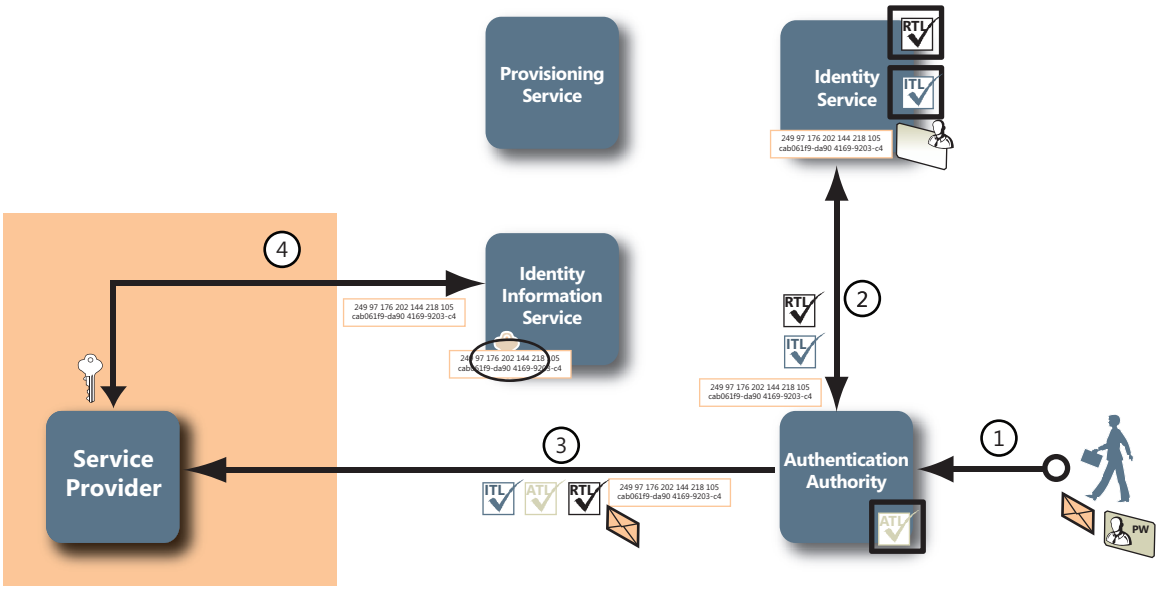
Let's consider a typical scenario for Authentication to illustrate the implementation of the Trust Model (see Figure 4, page 17).

(1) The Subject provides its Credentials to the Authentication Authority. (2) The Authentication Authority fetches the unique ID (a GUID in this example) and the ITL and RTL from the Identity Service. If needed, it may interrogate the Identity Service for more detailed information contained in the Trace object. The Authentication Authority determines the ATL and updates the Trace object to reflect the Authentication steps it has performed. (3) Subsequently, it propagates the service request to the Service Provider and embeds the Identity information and Session Trust object in the service call. (4) In this example, the Service Provider implements its own unique identifier (Domain Unique Identifier) for the identity and queries the Identity Information Service about the association of the unique identifier to the Domain Unique Identifier. The Service Provider can also fetch the Trace object from the Identity Information Service to get more detailed information if required.

Listing 2: Structure of the Trace object

```
Trace
    IdentificationProcess
    IdentificationStep (1-n)
            Policynumber (preferably an OID)
            ExecutedByWhom
                Name/IDno
                Institution
            DateTimeOfExecution
            ReferenceTo SourceDocuments
            AcceptanceOfIdentificationContract
    AuthenticationProcess
        AuthenticationStep (1-n)
            AuthenticationAuthority
            DateTimeOfAuthentication
            Credentials
            Medium/Channel
            Device
            Location
    ReputationHistory
        ReputationChange (1-n)
            PolicyNumber (preferably an OID)
            ExecutedByWhom
                InitiatedByWhom
                Name/IDno
            DateTimeOfExecution
            ReferenceTo SourceDocuments
```

Figure 4: Authentication



Conclusion

We started this by addressing the business and IT challenges. The ability to communicate Identity and Access Management information between parties is the key. Facilitating such communication requires quantifiable and verifiable Trust Levels. This applies to parties such as businesses within organizations, an organization with its partners, and also services in an SOA architecture.

Additionally, in order to meet the challenges, enable Access Control Policy enforcement at the to-be protected Resource at a level that is adequate for the situation at hand, tuned to the value/risk level of the transaction.

The architecture as developed — of which we discussed elements such as Session Trust object, Trace object and the mapping of the various Trust Levels to the value/risk levels of the business transactions — is capable of addressing these challenges from a technical perspective. However, it all boils down to the willingness of the parties involved to cooperate, to establish formal agreements for this cooperation, and to adhere to those agreements. But isn't that the main challenge in the whole Identity and Access Management domain?

Resources

International Telecommunication Union
ITU-T Recommendation X.509 (03/2000)

Web Services Security: SOAP Message Security 1.1,
OASIS Standard Specification, 1 February 2006

Web Services Security: SAML Token Profile 1,
OASIS Standard, 1 February 2006

Assertions and Protocols for the OASIS Security
Assertion Markup Language (SAML) V2.0,
OASIS Standard, 15 March 2005

Authentication Context for the OASIS Security
Assertion Markup Language (SAML) V2.0,
OASIS Standard, 15 March 2005

About the Authors

Gerrit J. van der Geest is an IT architect and program manager who has over 25 years experience in IT. Gerrit holds a master's degree in applied physics from the Eindhoven University of Technology in the Netherlands. He is cofounder of consultancy companies in the Netherlands and in South Africa. Prior to that, he held program management positions at Digital Equipment Corporation and Philips, where he was responsible for many international system integration programs.

Gerrit specializes in IAM, SOA, and infrastructure architecture and has been responsible for the development of IT strategies, domain, and solution architectures for large insurance and manufacturing organizations. Gerrit welcomes feedback at gerrit@navit.co.za.

Carmen de Ruijter Korver is a program manager with experience in IT infrastructure programs and architecture assignments. She worked for Compaq in the Netherlands, where she was responsible for the setup of a program management office and worked on many international IT infrastructure programs. Thereafter, she cofounded Navit (Pty) Ltd. in South Africa, where she specialized in the area of IAM with a focus on IAM migration strategies. Carmen is a PMI and PRINCE2 certified program manager.

Architecture Journal Profile: Kim Cameron

For this issue of *The Architecture Journal* on Identity and Access, we interviewed Kim Cameron, a Microsoft Architect whose thoughts on this area are the basis of one of the latest Microsoft initiatives in Identity Architecture: Windows Cardspace.

DD: Hello, Kim. Tell us who are you and what you do.
KC: I am Kim Cameron and I work as the Architect of identity at Microsoft. What I do is try and figure out what kind of problems can be solved with Identity and then how we can build the systems that respond to those problems. It's a very-wide ranging type of work where we have to think about all of the different experiences in the complete realm of computing and how they relate to Identity.

DD: Many of our readers know you from the paper on the Laws of Identity you published some years ago. [A condensed list of Kim's seven laws of identity appears in Fernando Gebara Filho's article on page 4.] Can you tell us how you went from Kim Cameron the individual to Kim Cameron the Identity Architect?
KC: Well, it took a long time. In the beginning, I worked on e-mail and it became clear after a certain amount of time that, this is back in the 1980s, the problem of getting e-mail from one person to another person wasn't really the problem of transporting a message—it was as much a problem of finding out the name or address of the person you were sending it to. In terms of routing things, it was a matter of how to find out where to route this, and so on. So it turned out the real problem with e-mail was more of a directory problem than an e-mail problem, and I started to become interested in directories. Once we started working with directories, it became clear that every little application had its own directory.

So directories through their multiplicity were as much of a problem as a solution. I started to see we had to have a new way of looking at directories, which I called Meta Directory, to unify the different directories at a logical level. And you know, it's a kind of a daunting problem. I started falling into this problem and no one else was looking at it, and that caused me to fall further and further. Basically. the problem hypnotized me and pulled me into being an architect.

This was long before I came to Microsoft. One of the reasons I came to Microsoft was because this is one of those huge infrastructure problems. This isn't the kind of problem that can be solved by one or two people. It can't even be solved by one or two companies. It has to be solved across the whole industry and you need to have a place to work where you can pull people together right across the industry. That was one reason why I was so interested in Microsoft.

DD: What advice would you share to those who want to be recognized for their abilities as an architect?
KC: I would have two pieces of advice. One is to fully explore a problem in a way that is extremely self-critical, so that you are willing to expose everything you think to a complete rethink all the time, and to make sure that you are aware of all the conflicting views and embrace those and embrace the knowledge in those views. In other words, be thoroughly scientific and nonemotional. Some people get on a kind of a hobby horse and then something else comes along that threatens their vision and there is a tendency for people to just cover their eyes. Don't cover your eyes—embrace the ideas because you have to really solve the problem. That is how you build your reputation —by solving the problem.

The other thing, which took me a lot longer to figure out, is that you have to really explain the story. The reason people do the wrong thing isn't because they are evil or stupid or something, it's because the story you know—the technology and science of it— hasn't been explained properly. So instead of arguing about the issues, you need to find ways to lay down the objective characteristics, and that's what the laws of identity represent. It really wasn't a change in my way of thinking, but a change in my ability to express my thinking.

DD: The architect role requires understanding current and future technical trends. How do you stay up-to-date?
KC: Well, one of the things I do is—I blog. When you talk about blogging, people often say, "I would like to blog, but I don't have time and besides no one would read it." In fact, when I started my blog, I definitely thought that no one would read it, but at a certain point, I realized it didn't matter.

What was useful was expressing my ideas. Once they go into the Internet contraption, they are there forever, so I had to express them in a way that wasn't too stupid, but also face the fact that I was going to be changing my ideas and this transformation of my thinking was going to be public. That was a starting point—a transformation of my thinking as a public thing, not a private thing. Then, other people who are interested in these issues may not sit there and read your blog on a daily basis, but every now and then they will sift through it. They'll have reactions to it and often they will write to you. They will either write to you or about you. As I was writing about these ideas, other people would comment on them and point out, "This isn't clear" or "This is not a good word because it implies the wrong things." So instead of sitting

in my attic and producing a paper that would be misunderstood by everybody, they actually helped me not only to improve entire points of it, but find a way to explain it. And of course, I wouldn't have to go out and find out about conflicting ideas. People would come to me with, "Why don't you look at this? Why don't you think about that?" It became a tremendous point of concentration of information.

It's paradoxical that by originating information, you actually end up consuming more information. So that's one thing. Another thing is that being at Microsoft, we are very lucky because we get to be in conversations with many people in all different governments, industry, and the academic world. So, putting those things together, I would say to the younger architects, make sure you talk to as many people as you can. Be open—don't avoid. I also try to read a lot.

DD: Name the most important person you have ever met in this industry. What made him/her so important?

KC: That's a hard question. I have met a lot of very interesting and inspiring people—but I guess I'll pick Craig Burton. Craig Burton is an analyst who used to be involved with Novell. He was very crucial to the original success of Novell's Netware—the early version of Netware. When I met him, I had been working this Meta Directory concept and had actually started to build the Meta Directory. My company was producing this thing, but of course we couldn't talk about it. We didn't have any words for it, we just knew we had to solve a technical problem. Craig introduced me to this problem of communication and helped me understand that the communication process was as crucial to the technology as, say, the analytical process. For example, what is the name of something? As you try and name it, it becomes much clearer than the original intuition that you have as an architectural thinker. Initially, it seems clear enough to you, but when you go and explain to somebody else, you may have to sit there and work on it for 20 minutes to get the point across.

The aim is to make everything clear enough that you can get the point across instantly. So it's a matter of sharpening the concepts and of not being afraid to be sophisticated; in other words, there is no need to condescend to the audience. You can be scientific and feed as much clarity as you want. For example, we had the question of what to call this thing we had invented, and I just laughingly said, "You know, we had thought of calling it a 'Meta Directory,' but of course you couldn't really call it a Meta Directory because everybody would think we are existentialists or something alien." And Craig said "No—if you call it what it is then people will start to use those words. Call it what it is, don't call it something else in order not to offend some people."

DD: Is there anything you did that you'd do differently if you could turn the clock back on your career?

KC: I am not a person to think backwards that way. I guess if I had known what I know now at the beginning of my career, a lot of things I had done would have been much more successful than they were—right? When I did my original e-mail system, I couldn't believe that I could actually withstand the competition, with others. It was only later, when I met these people, they told me that they were really worried about the technology that I had been bringing forward. In other words, I didn't have a large enough view of what I was doing. I think that's often the case. People look at a company like Microsoft and they say, "Well gee, if I am doing a technology and Microsoft is doing that technology—is it really worth it for me to be in the same area doing the same kind of thing?" And it is! Because it creates an ecology, and it creates room for these different products and points of view. The fact that Microsoft has a product creates the room for another product that may be specialized in some way, but that otherwise wouldn't have any chance at all. So I can understand the synergy part of the industry. I saw it as very much "dog eat dog," and now I am much more of a believer that "the other guy" is my best ally. Because we are both building this new world that hadn't existed, and by having two of us build it we can do a much better job of improving our products—and we'll actually sell a lot more of it.


Kim Cameron

Kim Cameron is chief architect of Identity in the Connected Systems Division at Microsoft.

Kim joined Microsoft in 1999 when it bought the ZOOMIT Corp. As vice president of Technology at ZOOMIT, he invented Meta Directory technology and led the development team in producing a range of SMTP, X.400, X.500, and PKI products.

Kim grew up in Canada, attending King's College at Dalhousie University and l'Université de Montréal. He has won many industry awards, including Digital Identity World's Innovation Award (2005), and Silicon.com's Agenda Setters 2007.

Kim blogs at identityblog.com, where he published *The Laws of Identity.*

DD: Do you see your ideas also being considered in other companies?

KC: Well, I don't want to pose as the source of all thought in this area, but certainly other organizations have embraced much of my work. I'm also trying to synthesize what others are doing right. This is the idea of keeping your mind open and embracing what surrounds you. But it's been very interesting because the identity area is unique in some ways: Identity is most important when you are reaching across to somebody else—including a competitor.

Clearly, we can't have a solution for the Internet that just works with Microsoft products. People live in a much bigger world and that whole world has to be aligned if technology is to be really usable. In this sense, I consider everybody else in the industry to be as much as an ally as my colleague in the next office here.

The industry is really coming together. The Information Card Foundation, for example, will have been launched by the time this interview reaches publication. So that's all these companies who've come together to produce the compatible software we have, like IBM, Oracle, Sun, and even smaller companies: all kinds of people participating in this new technology.

DD: What does Kim the Architect's future look like? What do you hope to accomplish in the next few years?

KC: Well, I would like to see the deployment of this Identity Meta System. We are currently at the stage where various vendors have started to produce software, but we are not yet at the stage that people have deployed it.

I also have a project that I worked on earlier, which is a new way of conceptualizing and building directories, based on what I call "Polyarchy." Polyarchy means that instead of having hierarchies, you can shoot it across these different dimensions. So I am trying to evolve the nature of directories in that direction.

Federated Identity and Healthcare

by Mario Szpuszta

Summary

Healthcare is one of the fastest growing markets in the IT industry. It is also one of the most delicate and challenging industries. That is because of the highly heterogeneous technical environment and the various political interests of involved parties – reaching from medical practitioners or pharmacies through hospitals to the ministry of healthcare. For architects, such environments mean we have to build bridges – our systems need to be able to deal with both technical diversity and political interests, especially when it comes to responsibilities and security. The concepts of the Identity Meta-System Vision and its federated approach help us architect and build such systems. In this article, we will discuss those core concepts and drill into parts of proposals we made during a prototyping engagement with the Austrian Medical Association, SVC GmBH. – an IT daughter of the Austrian national insurance responsible for the Austrian electronic healthcare ID-card infrastructure (e-card) – and finally, the Microsoft Innovation Center Copenhagen.

Federation and the Identity Meta-System Vision

Summarizing the concepts and the primary goal of the identity meta-system vision, originally invented by Kim Cameron, within one sentence is a challenge, as you can see by taking a look at Kim's article on MSDN (http://msdn.microsoft.com/en-us/library/ms996456.aspx; a condensed list of the seven laws of identity appears in Fernando Gebara Filho's article on page 4). One of the primary targets of the identity meta-system vision is the development of concepts, models, and processes to unify and structure the way we deal with digital identities in the future from all perspectives, including parties that issue digital identities, rely on digital identities and – especially important –deal with digital identities (we as end users).

Today such concepts, models, and processes do not really exist as a standardized common layer across all areas. This is the main reason we (end users) are losing control of our information, our digital fingerprints, in the cloud. We are sharing information about ourselves in an unstructured way with many different parties. (How many user profiles do you have?) We specify information for each and every registration with an Internet site and finally we lose control of what we specified where and for which reason. Even more interesting is the reality that we are protecting this information with very weak mechanisms such as simple username and password combinations that can be easily stolen (see www.antiphishing.org).

By defining standards that include common concepts, models, and processes with clear roles when dealing with digital identities, the identity meta-system vision tries to solve the aforementioned issues. These concepts target the missing "standardized Identity Layer" in typical layered models such as OSI. Today, these include standards for things such as communication or address resolution; they do not talk about identities. Figure 1 illustrates one model that has evolved within the identity meta-system. This model emphasizes the clear separation of responsibilities for different parties as well as the communication paths between them.

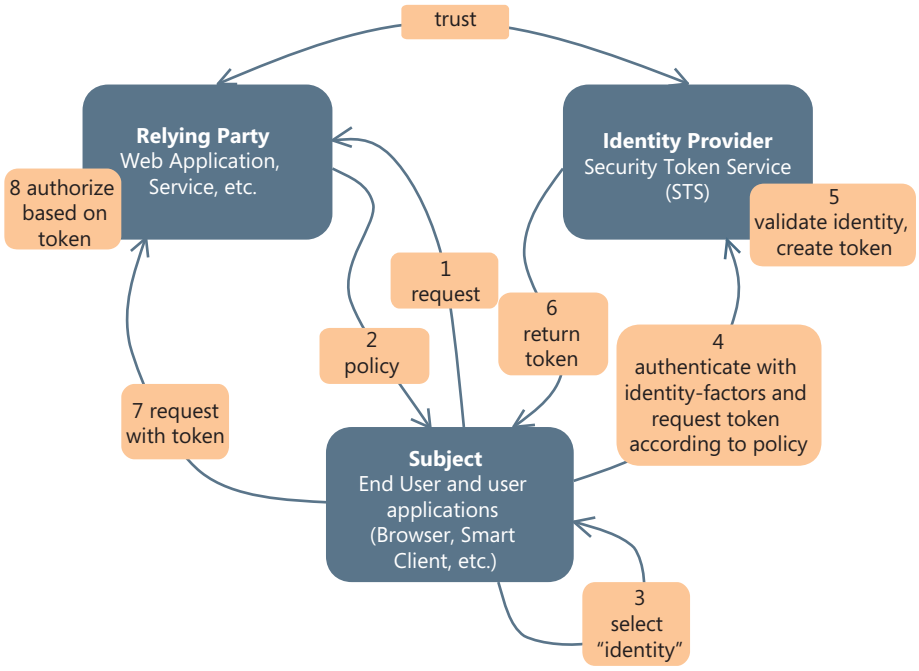
The model differentiates between identity providers, relying parties, and subjects. An identity provider (IP) essentially is responsible for managing information about us. Typically, we trust identity providers and therefore let them store our personal information. Identity providers also validate identities and verify the correctness of an individual's information. Technically, an identity provider is a web service called a "Security Token Service (STS)," which implements certain interfaces. Then we have relying parties (RP), implemented as applications or services. Relying parties typically have some offerings that they make available to subjects if those subjects can prove that they fulfill certain requirements. These requirements are statements about us, called "claims". These statements associate end users with properties, permissions, or anything similar. For that, the relying party trusts an identity provider who can verify the correctness of these statements. A relying party authorizes based on statements (claims) verified by an identity provider. This mechanism is typically called claims-based security. Finally, there is the subject, which represents the individuals in the model. Each individual needs to verify statements about them when accessing services of a relying party. If a relying party requests a verification, the subject is in control of which identity provider it uses for providing the required, verified list of statements. With that selection, the subject also decides which information is forwarded to the relying party. Essentially, a subject selects an identity with a so-called identity selector. The identity selector and its underlying standards provide the "common, unified user experience." Identity selectors are outside the scope of this article. We essentially rely on the role-separation as well as the protocols defined in the model you've seen in Figure 1.

There is one important fact to keep in mind about this model: It neither prescribes any specific type of authentication against the identity provider nor does it prescribe any specific security token type

that identity providers need to issue. It just specifies clear separation of responsibilities and defines communication paths based on standardized web service protocols based on leveraging the WS-* standards, as you will see in the last section of this article. Another important fact is that relying parties are no longer responsible for storing and managing information about the subjects. That is the task of identity providers, with the long-term goal that we as users have just a couple of trusted identity providers that are storing information about us –as is the case in the real world. This is a big difference from today's digital world, where nearly every web site creates user profiles and stores all the personal information we have entered for user profiles within other sites as well. In the real world this is much different: Only a few identity providers have information about us and we typically know what information they have. Furthermore, in the real world many identity providers do not store the same information about us again and again. They rely on other identity providers in cases where they require that information. That has a couple of interesting effects: There is no central data storage available that has all our information, which is important from a data protection and privacy perspective, and responsibilities in terms of verifying statements about subjects can be delegated between a set of identity providers. And that brings me to the most important aspect of the model in Figure 1 for our healthcare discussion: The flexibility of this model by its nature, due to the fact that the "authentication" against the identity provider is not prescribed in any way, allows us to do what we do in the real world: build trust chains and delegate responsibilities between identity providers. What does this mean? Well, authentication against identity providers can happen either via classic authentication mechanisms such as smart cards, Kerberos (bound to username, password – yes, that's still possible), certificates or – and this is the interesting part – through a token issued by another identity provider. Every identity provider can be a relying party of another identity provider, as well. This helps us build federated systems with trust chains and responsibility delegation, which in turn helps us bridge political and technical silos. Figure 2 illustrates this.

As you can see, identity provider 1 requires a token issued by identity provider 2 that proves certain information before identity provider 1 can prove other information required by the relying party. Every identity provider in this scenario can prove a certain set of statements about a subject and therefore can associate permissions with the subject, but none of the IPs can prove all the required information about the subject. The relying party then authorizes based on statements proved by one or both identity providers. Essentially, the identity providers are controlling the authorizations of the relying party by providing (or not providing) certain statements. So in Figure 2 you have two identity providers that can independently (or together) affect the authorization decisions made by the relying party.

Figure 1: A model for clear separation of roles when dealing with digital identities



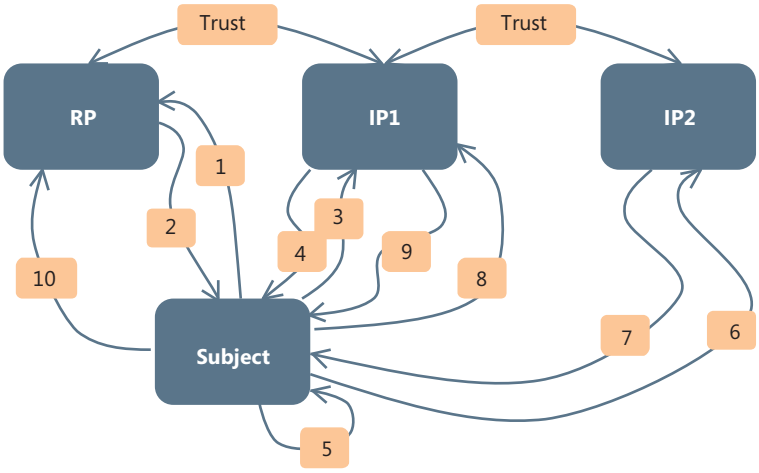
With that, we can now discuss how these concepts can be used to effectively structure security in a federated environment– based on the proposals we made in Austria.

Austria's Healthcare Infrastructure and Future Plans

Austria has an infrastructure in place available for every insured citizen and for a majority of medical practitioners called the e-card system. Originally, the e-card system was intended to support electronic processing for insurance-related tasks such as payments for treatments without the need for additional paperwork. For patients, this means they do not need to worry about any paper-printed health insurance certificates. The e-card system involves smart cards for patients and medical practitioners, some special equipment medical practitioners need to install in their ordinations to be able to use the e-card system, and back-end services for querying information about e-card holders and to start insurance-based back-end processes. For our purposes here, we don't need to cover the technical details about the e-card system and its deployment.

While the e-card system is seen as a platform for enabling many future e-health services, it is really just one system within a complex, heterogeneous landscape of systems in Austria. First, the availability of the e-cards is limited to patients who have a national insurance. It does not cover people without such an insurance, whether they are just visiting Austria or not. These patients need other means of authentication, such as citizen cards. Second and more important is the issue of political responsibility and ownership. The national insurance is willing to manage some, but not all, information about patients and medical practitioners. For example, they do not store complete healthcare records such as lab reports or analysis reports from hospitals because "that's not their job". The responsibility for managing these kinds of information is assigned to hospitals or other

Figure 2: Trust-chain between identity providers



types of medical practitioners (such as lab physicians). That means they are also responsible for securing information about patients. In order to support patients with and without e-cards, they need to accept multiple types of authentication. This brings about a situation where a patient's information (records) is spread around the country between several different parties. The advantage of this is that there is no single point of attack for those who want to compromise the patient's privacy and security. On the other hand, wouldn't it be great if a medical practitioner or hospital had access to all pertinent information about a patient in order to ensure the best possible medical treatment? Of course it would, and that's why the Austrian Ministry of Healthcare started a working group to design a system for electronic healthcare record management. This system should give medical practitioners, hospitals, pharmacies, and patients access to all necessary medical information. Access should be possible anywhere in the country whenever needed, but only if the patient agrees. A centralized solution is not an option for Austria due to its very strong data protection law that protects the patient's security and privacy in multiple ways. Therefore, storing information in a distributed fashion across several systems managed by different parties is a core requirement for an electronic healthcare record system. And that is one of the main reasons we have taken a federated approach into account when we designed our prototype. Furthermore, the architecture needs to support other types of authentication in addition to the e-card. The e-card system is a widely deployed and accepted standard and it will play a core role in Austria's future e-health developments. Therefore, we've taken the e-card as a starting point and worked on proposals for extending its service interfaces to support federated identity scenarios.

Data Protection of Patient's Data is Core

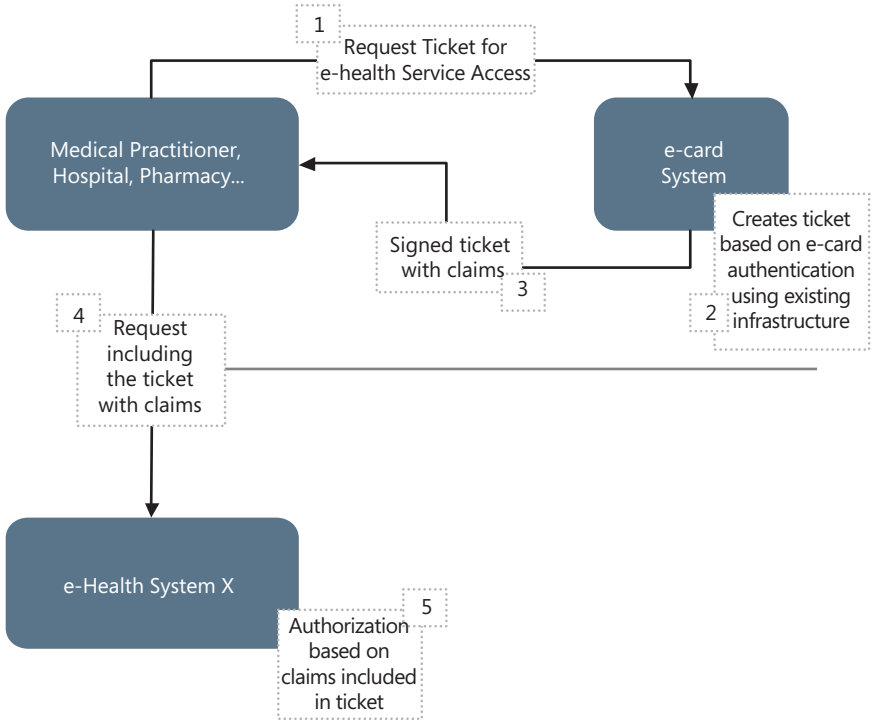
One of the major debates within all the electronic healthcare records discussions revolves around the patient's security and privacy. Austria has one of the strongest data protection laws in the world and acts as an example across the European Union. Short and simplified: The data protection law states that access to ANY information about an individual must be approved by that individual each and every time. According to the Austrian Medical Association, there must be a combination of an organizational and technical process in place to allow the following (while health service provider can be any party with healthcare offerings such as medical practitioners, pharmacies, hospitals, etc.):

(1) Patients must have the opportunity during every visit to a health service provider (e.g., doctor) to give their explicit consent for access to their personal data (health record). If patients do not give their consent, their data may not be accessed.

(2) Information about a patient in e-health systems may only be made available to other health service providers for other treatments with the patient's explicit, additional consent (beyond the general consent described in item 1 in this list) given in a confidential discussion with the health service provider (e.g., doctor).

Although some organizational details are still being discussed between the different political parties, they all agree that a process such as the one outlined in this section must be in place. This requirement definitely influenced the way we designed our prototype.

Figure 3: The e-card system as a Security Token Service (STS)



A First Step – E-Card System as a Security Token Service

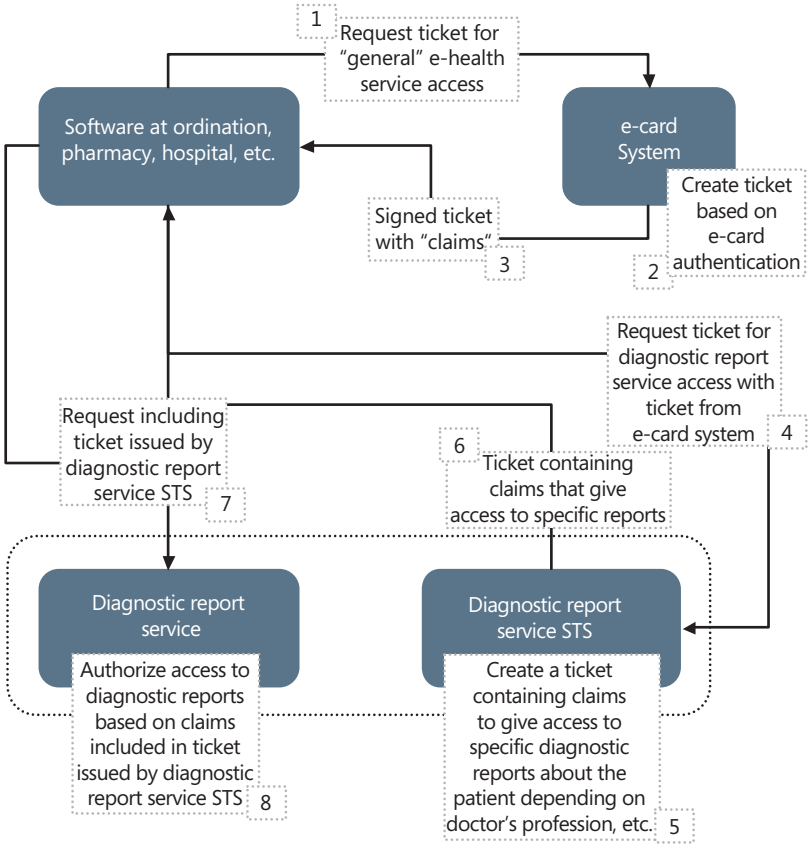
Based on the concepts of the identity meta-system vision, the versatile authorization decisions, and the need for a federated security architecture for e-health systems in Austria, we came up with the idea of extending the e-card system to a security token service. As such, the e-card system issues tickets based on the well-established e-card authentication process. These tickets include claims, which are essentially statements about the patient and/or the medical practitioner in a specific situation. These tickets, or the claims in the tickets, allow e-health systems (such as an electronic prescription system) to make authorization decisions. The tickets need to be digitally signed by the e-card system so that the relying e-health systems can trust the tickets and accept them as verification of specific information about patients, medical practitioners, and the current situation they are acting in. Figure 3 illustrates this simple but powerful concept.

The really nice thing about this solution is the fact that we are splitting the responsibilities – as the identity meta-system vision proposes with its models as well. The e-health service is no longer responsible for authenticating the request. It delegates this complex task to a service originally built for authenticating patients and medical practitioners – the e-card system. That means every e-health service just needs to tell its client counterparts which statements about the patient, doctor, and their current situation need to be verified and which identity provider is able to verify these statements. In our prototype, the identity provider is the e-card system, which then performs this complex task if the e-card authentication succeeds. But the architecture also gives the e-health system the ability to accept tickets from other identity providers that support other types of authentication such as citizen cards (e.g. from foreign countries). It even gives the e-health service the ability to accept tickets from identity providers, which verify additional information about a patient compared to that verified by the e-card system. In such a scenario, we would have a trust chain between another identity provider and the e-card system as an identity provider. Furthermore, the authorization within the e-health service itself is very simple: It just queries and analyzes claims stored in the ticket created and signed by the e-card system and forwarded through the software running at the doctor's ordination.

Now let's take the data protection requirements as an example of how claims can simplify the implementation of relying parties. Referring back to the previous section about data protection, it could be implemented with the claims-based architecture and the e-card system as a security token service as follows (simplified scenario):

- The patient goes to a medical practitioner and uses the e-card for authentication by pushing the e-card into a card-reader. During the process of authentication, the patient gets the chance to formally agree to give the doctor access to the electronic healthcare record.
- Depending on whether or not the patient agrees, the e-card system issues a different ticket. If the patient agrees, the ticket could contain a claim signed by the e-card system that states the doctor (its software) will have read-only access to the patient's electronic healthcare record for the validity-period of the ticket. In its simplest form, this claim could be a Boolean flag. As it is signed by the e-card system and issued during the authentication process, other e-health services can trust that claim within the signed ticket and grant access to specific parts of their services and information.
- Next, the patient has a private discussion with the doctor. During that discussion, the doctor's software can use the previously issued ticket.
- At the end of the discussion, the doctor creates a diagnostic report that would be useful to other doctors during the patient's medical treatment. According to our data protection requirements outlined in the previous section, the patient needs to have the opportunity to give an additional, explicit agreement for this step by pushing his e-card a second time and running through an additional authentication process. During that authentication process, the e-card system requires that the e-card be pushed, as well as the previously issued ticket for issuing another ticket that includes a claim for verifying that the patient gave an additional requirement.
- This new ticket is then used by the doctor's software to go to an e-health service that allows sharing diagnostic reports with other medical practitioners. This e-health service will share a report only if it gets a ticket containing the claim that verifies that the patient authenticated a second time. The mechanisms for implementing such a service are exactly the same as any other– the service just

Figure 4: Trust chain between the e-card system and an STS dedicated to specific e-health services



evaluates one additional claim in a ticket issued by the e-card system. The e-health service can trust this ticket and the claim because it is issued by the trusted e-card system and the trusted e-card system in turn issues the ticket only based on a previously issued ticket stating the first-time logon.

With that process and architecture in place, our architectural proposal would allow a very simple implementation of the two-phase agreement according to the data protection law. And furthermore, this complex and critical requirement would then be implemented in one single place and not again and again for every service, as the e-card system can do the patient-agreement verification for us as an identity provider. This demonstrates the power of claims-based security and how it simplifies the way we deal with rather complex requirements. My first thought was: "It can't be that simple." But when you think about how things work in the real world, we have to admit that the real world is nearly as simple. So often authorization decisions are made based on simple statements on a ID card or a conference badge (e.g. people wearing the conference badges with a red band are speakers and may walk into the speaker lounge, others may not).

Extending the e-card system to a security token service was just the first step. It turned out that the e-card system is able to proof a large set of information about patients, doctors, etc. – but not all information that might be necessary for any future e-health service. Also, the e-card system and its owners do not want to own all information for political reasons, which brought us to the next idea of extending the model to a federated security system based on the standards proposed by the identity meta-system.

The Next Step – Federated Security Model

The reasoning behind creating a federated security model was the fact that the e-card system will not be able to deliver all claims in tickets, which might be necessary for all the different e-health services in the future. Also, for political reasons, the owners of the e-card system don't want to provide all possible types of claims to drive (!!) authorization decisions. According to them, other parties will have the necessary know-how and political position to manage claims requiring more specific authorization decisions. Therefore, the idea was that if an e-health system requires additional information (claims) in order to carry out an authorization, and if this information cannot be made available by the e-card system, it should be possible to introduce additional security token services capable of issuing tickets for access to these particular e-health services (or a set of particular e-health services). Of course, these systems must have a trust-relationship with the e-card system and will issue tickets based on a ticket that has already been issued by the e-card system – if they rely on e-card authentication.

> **"THE POWER OF CLAIMS-BASED SECURITY IS THAT IT SIMPLIFIES THE WAY WE DEAL WITH RATHER COMPLEX REQUIREMENTS"**

Let's take a possible, typical scenario: A diagnostic report delivery system that, in order to be able to limit access to diagnostic reports, needs to make authorization decisions based on specifics defined by laboratories, laboratory specialists, hospitals, and doctors. These specifics are potentially not known by the e-card system and therefore the necessary claims for making an authorization possible are managed by the aforementioned parties (laboratories, doctors, etc.). For issuing such claims, an additional security token service is necessary. Figure 4 illustrates that scenario with an e-health service that requires its own security token service.

This architecture allows flexible future extensions of the security infrastructure in the system by adding new security token services as new flavors of authorization that require additional types of claims arise without losing or affecting existing investments. The model also allows splitting up "authorization responsibilities" for several reasons ranging from political, technical, up to know-how. Finally, the model also allows for the consolidation of security token services if the landscape changes, and simplification is possible for any reason without really affecting the implementation of the different e-health services. For the e-health services, just the identity provider changes, with some extensions or consolidations within the overall system. As long as the claim-types remain available, you only need to configure another STS for the e-health service and everything remains fine. Figure 5 illustrates this situation.

Figure 5: Future extensibility, flexibility, and agility due to federation

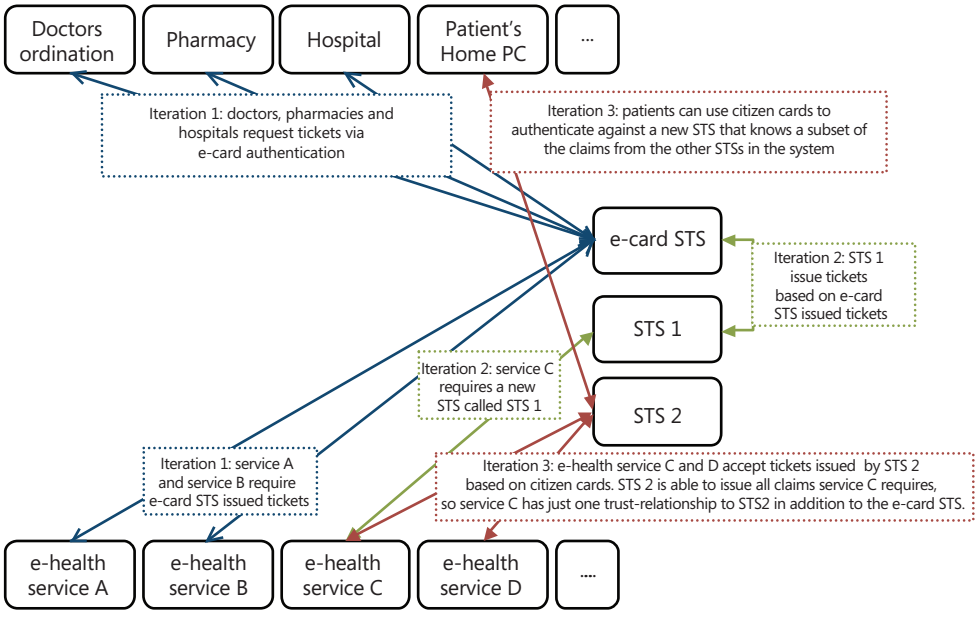
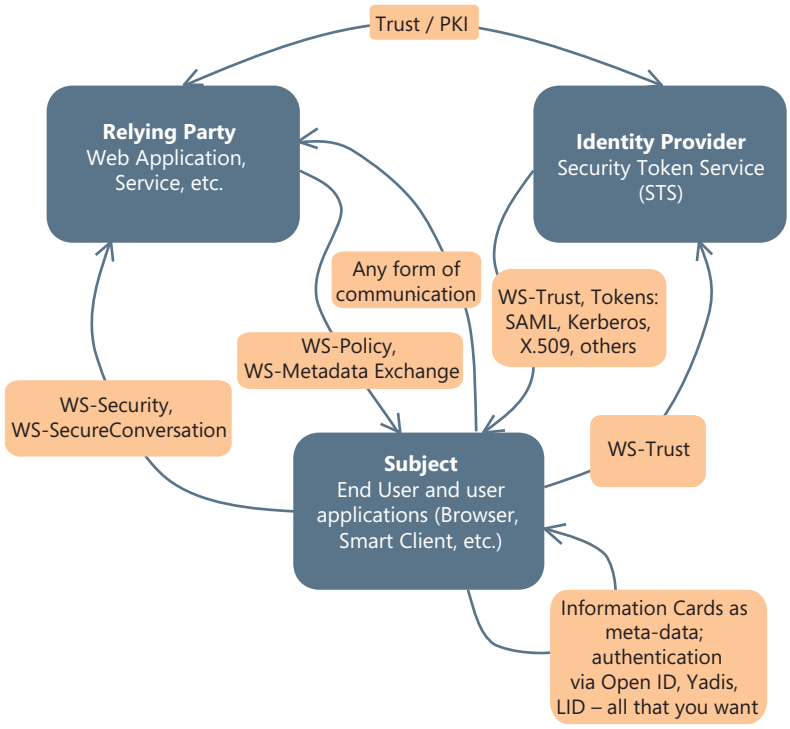


Figure 6: Standards in the identity meta-system vision



As you can see in iteration 3 of Figure 5, it is possible to include further authentication mechanisms and factors in the future while still reusing existing e-health services without affecting their implementations – as long as the claim-types remain the same. This is exactly the flexibility and agility we need for complex and heterogeneous environments such as the health-care industry. This is also why standardization efforts such as IHE are moving in this direction (while IHE leaves the used standards and protocols open and stays on an even more conceptual level).

Can We Implement the Model? Mapping Technologies!

An architectural model is worth nothing if we cannot implement it. Therefore, the next step is to assess how far along the technologies are and whether it is practical to implement a system as previously outlined. First, we need to define the standards for implementing the federated security model as proposed in the identity meta-system vision. Figure 6 highlights the most important standards:

All these standards are part of the WS-* stack, which is standardized by OASIS on top of the existing W3C standards for web services. But having standards does not necessarily mean we can easily implement the systems. So what about the support in development frameworks and runtimes? Generally speaking, the world of the identity meta-system is much more real and the best proof is the Catalyst Europe conference held last winter (see http://www.identityblog.com/?p=889). At this conference, several technology vendors (commercial and open source) were invited to test interoperability of their relying party, identity provider, and identity selector/subject implementations, and the results were very promising. Interoperability based on the standards shown in Figure 6 works between a variety of platforms including Microsoft .NET, Java, PHP, and even Ruby. At Catalyst Europe, vendors verified interoperability between several platforms including Eclipse Higgins, xmldap.org, Bandit (Novell), Verisign PIP, and the Microsoft .NET Framework. The .NET Framework has supported the necessary protocols since version 3.0 with the Windows Communication Foundation (WCF). Digital identity selectors are supported through Windows CardSpace, also part of the .NET Framework 3.0. The protocol binding you have to look for in WCF is the wsFederationHttpBinding (for more information take a look at the following article on MSDN: http://msdn.microsoft.com/en-us/library/aa395215.aspx).

Conclusion

Health care as a politically and technically complex, heterogeneous environment is definitely one of the places where the concepts and ideas of the identity meta-system vision are at home. We discussed the most important concepts of the identity meta-system that help you build services in such a complex environment. These concepts include a clear separation of roles and responsibilities. These are divided into identity providers, relying parties, and subjects. The meta-system also defines the communication paths and protocols based on WS-* web services standards. You can apply these concepts by taking a look at a prototype we've built in Austria together with the Austrian Medical Association, the SVC GmbH. (an IT daughter of the national insurance), and the Microsoft Innovation Center Copenhagen. Clearly defined roles and responsibilities help architects build loosely coupled systems where different parties with different interests are involved by decoupling authentication and assignment of permissions through claims from the actual authorization code in the business services – all based on open WS-* standards. These are the types of architectures that help us build bridges — whether they are technical or political!

About the Author

Mario Szpuszta works as an architect in the Developer and Platform Group of Microsoft Austria and supports software architects of enterprise accounts and top 10 web site accounts in Austria. Mario always focused on working with Microsoft technologies and started working with the .NET Framework very early. In the past years, he focused on secure software development, ASP.NET web development, web services, as well as integrating Microsoft Office Clients and Servers in custom applications using Microsoft .NET, Office Open XML File Formats, and web services. Mario co-authored "Advanced .NET Remoting 2nd Edition" with Ingo Rammer and participated in writing "Pro ASP.NET 2.0 in C#" as well as "Pro ASP.NT 3.5 in C#" with Matthew MacDonald.

Claims and Identity: On-Premise and Cloud Solutions

by Vittorio Bertocci

Summary

Today's identity management practices are often a patchwork of partial solutions, which somehow accommodate but never really integrate applications and entities separated by technology and organizational boundaries. The rise of Software as a Service (SaaS) and cloud computing, however, will force organizations to cross such boundaries so often that ad hoc solutions will simply be untenable. A new approach that tears down identity silos and supports a de-perimiterized IT *by design* is in order.

This article will walk you through the principles of claims-based identity management, a model which addresses both traditional and cloud scenarios with the same efficacy. We will explore the most common token exchange patterns, highlighting the advantages and opportunities they offer when applied on cloud computing solutions and generic distributed systems.

The Sky Is the Limit

When you look at a cloudy sky, your inner child probably sees dragons and castles; don't be surprised if your inner architect, after having read this article, will see dollar signs. Cloud computing promises to bring radical advantages to the way in which we think of IT: Its basic idea is that companies can host assets outside of their own premises, reaping the benefits of those assets without the burden of maintaining the necessary infrastructure. This is somewhat similar to the idea of SaaS, where companies can avoid the burden of maintaining on-premise applications that are not specific to their core business, buying the corresponding functionality as a service. Cloud computing, however, pushes the bar further. Instead of buying complete applications provided by third parties, such as the classic CRM and HR packages, the cloud offers the possibility of *hosting your own resources* in data centers that are exposed to you as a *platform*. You have all the advantages of retaining control of the resource, without the pain of CPU and bandwidth usage, dealing with the hardware, cooling the room; you don't even need to worry about patching your system. If your web application produces new data every day, using a data store in the cloud saves you from constantly buying hardware for accommodating growth. The best part is that you can expect to be charged an amount proportional to the usage you actually make of the resource, instead of having to invest in hardware and infrastructure beforehand. This "pay-per-use" pattern is one of the reasons for you will often hear the term "utility computing" instead of "cloud computing," and it is even more evident in CPU-intensive tasks. Imagine if, instead of sizing your data center for handling its maximum forecasted peek and underutilizing it most of the time, you could deploy your most CPU-hungry processes in a data center of monstrous proportions: The CPU utilization could grow as much as requested, and you would pay your cloud provider in proportion. Those are some of the advantages that will light a sparkle in the eyes of your IT managers, but the Cloud holds even more interesting properties for architects. Since the cloud provider hosts resources on a common infrastructure, it is in the position of offering services that can be leveraged by every resource simplifying development and maintenance. Obvious candidates are naming, message dispatching, logging, and access control. Once a resource uses the cloud infrastructure, implementing those functionalities can be factored out from the resource itself.

The literature on cloud computing is vast and grows every day: If the introduction above was not enough to convince you of its disruptive potential, simply search for the term with your favorite search engine to get a feeling of how seriously the industry is taking it. The diligent architect, at this point, is likely to wonder, "Is my company ready for this?" Not surprisingly, answering this question is a complex task and requires considering many aspects of your architecture and your practices. In extreme simplification: If you run your business according to solid service orientation (SO) principles, you are in the ideal position to take advantage of the new wave. After all, if you respected autonomy, exposed policies, and used standards, who cares where your services run? If you are in that position, you have my congratulations. In my experience, however, nobody ever applies SO principles in excruciating detail. For example, services developed with the same technology offer special features when talking with each other, and there are situations in which it makes perfect sense to take advantage of those.

Identity management and access control are most likely to be affected by this phenomenon. Enterprises typically have their directory software, and they rightfully leverage that for many aspects of the resource access control; sometimes it works so well that developers are not exposed to identity concepts, which is actually a good thing, but that rarely happens. When faced with tasks involving some form of access control management, such as federating with partners outside the directory or using different credential types, you can expect developers to come out with the worst swivel chair integration solutions. If identity brings out the worst from development practices,

why do we get away with it? The easy answer is that sometimes we don't. I am sure you have heard your share of horror stories of access control gone wrong. The subtler answer is that we get away with it because, until we own the majority of the infrastructure, if we exercise iron-fist governance, we can somehow handle it: We may use more resources than needed, we may deal with emergencies more often than needed, but somehow we go on. In fact, "we own the majority of the infrastructure" is a fact that is challenged by growing market pressure. When a lot of your business requires you to continuously connect and onboard new partners, where does your infrastructure end and theirs begin? Cloud computing is going to snowball this: Once the cloud is just another deployment option, crafting custom access code for every resource will simply be not sustainable.

The good news is that there is an architectural approach that can help manage identities and access control for generic distributed systems, and it works for on premise, cloud, and hybrid systems alike. The core idea is modeling almost everything as exchanges of claims, and model transactions in a much more natural fashion.

This article is an introduction to this new approach. Special attention will be given to the aspects that are especially relevant for the cloud, but the vast majority of the concepts and patterns presented can be applied regardless of the nature of the distributed system. Note that the problems we are trying to tackle here are not just the simple single-site, membership provider-based application. While the principles laid down here apply to any system, hence also to simple cases, their expressive power is best utilized for scenarios including partnerships, complex access rules, and structured identity information.

If the following definitions are hard to digest at first, think of how difficult it must have been to learn to use the digit '0' for the Europeans of the Middle Ages, used to sum up numbers using the Roman notation. In that case, there was an established practice to go against, but the payoff of adopting a better model was exceptional!

Claims-Based Solutions

The issue with classic identity management solutions can be summarized as follows: They presume too much.

The most common assumption is that every entity participating in a transaction is well known by some central, omnipresent authority that can decide who can access what, and it what terms. This is usually true in self-contained systems, such as enterprise networks managed via directories, but fails when business processes begin to require alien participant such as software packages with their own identity stores, partners and customers accessing your extranet, and consultants. Tactical solutions, like using shadow accounts, often have to do with pretending to be able to manage something we don't own; and as such, they are very brittle.

Another common assumption is that every participant in a transaction uses a consistent identity management technology. Again, this is a fair assumption for self-contained systems (think network software), but it fails as soon as you let aliens in the process. The common practice in accommodating different technologies is treating those cases as exceptions. As a result, the resources themselves end up embedding a lot of identity management plumbing code, written by developers that usually are all but identity experts. This is every bit as bad as the old taboo for embedding business logic in the presentation layer, perhaps even worse. Handling identity plumbing directly inside the resource not only makes the system brittle and hard to maintain, it also makes the life of system administrators miserable. How can you manage access control at deployment time if the logic is locked inside the resource itself?

The claims-based approach defuses these issues by assigning each task to the entities who are its natural owners, and avoiding introducing artificial dependencies and expectations by respecting the autonomy of all participants – nothing but good old SO architectural principles.

To follow, I will give a quick description of the claims-based approach. The topic in itself is huge; in fact, I coauthored a book on the subject. In order to fully understand the implications in this article, I recommend reading Chapter 2, freely available online on MSDN (see Resources).

Basic Definitions

Here I will present a bestiary of the various concepts and constructs you will encounter while exploring claims-based approaches.

Claims

A claim is a fact about an entity (the "subject"), stated by another entity (the "authority").

A claim can be literally anything that describes one aspect of a subject, be it an actual person or an abstract resource. Classic examples of claims are "Bob is older than 21", "Bob is in the group 'remote debuggers' for the domain Contoso.com", and "Bob is a Silver Elite member with one Star Alliance airline. A claim is endorsed by an authority; hence one observer can decide if the fact the claim represents should be considered true according to the authority's trustworthiness.

Trust

An entity A is said to trust an entity B if A will consider true the claims issued by B. While very simplistic, this definition serves our purposes here. Trusting what B says about a subject saves A from the hassle of verifying the claim directly. Entity A still needs to make sure that the claim is actually coming from B and not a forgery.

Tokens

A security token is an XML construct signed by an authority, containing claims and (possibly) credentials information.

Security tokens are artifacts, XML fragments described in (see Resources: WS-Security), which can fulfill two distinct functions:

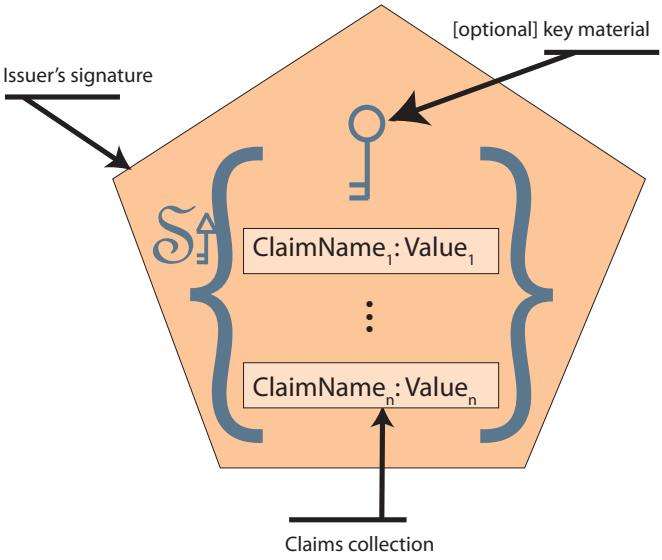
- they provide a means to propagate claims
- they can support cryptographic operations and/or have a part in credentials authentication

Thanks to the properties of asymmetric cryptography, the fact that a token is signed makes it easy to verify the source of the claims it contains.

Tokens can also contain cryptographic material, such as keys and references to keys, which can be referenced in encryption and signatures in SOAP messages; those operations can be used as part of credentials verification processes. In this context, we consider a "credential" any material that can be used as part of some mechanism for verifying that the caller is a returning user: Passwords and certificates are good examples (for more details, see Resources: Vittorio Bertocci's blog, The Tao of Authentication).

Tokens can be "projections" of specific authentication technologies, such as X509 certificates, or they can be issued (SAML, a popular token

Figure 1: Anatomy of a security token



format you may have heard mentioned in the context of web services security, is one example of an issued token). The system is future proof: As new technologies emerge, suitable token "projections" can be documented in profile specifications.

Security token services (STS)

A Security Token Service is a web service that issues security tokens as described by WS-Trust (see Resources: WS-Trust).

An STS (see Figure 1) can process requests for security token (RST) messages and issue tokens via requests for security token responses (RSTR). Processing the RST usually entails authenticating the caller and issuing a token that contains claims describing the caller itself. In some cases, the STS will issue claims that are the result of transformations of claims it received in the RST. (For more details, see Resources: Vittorio Bertocci's blog, R-STS.)

The identity metasystem (IdM)

The Identity Metasystem (IdM) is a model that provides a technology agnostic abstraction layer for obtaining claims.

This is a very simplified definition that does not render justice to the model; for example, it does not mention policy distribution and systems management. (See Resources: WS-Security, WS-Trust.)

Every identity technology tends to accomplish the same tasks, following common patterns and dealing with more or less the same functional roles. The IdM describes those patterns and roles in an abstract fashion, modeling the behavior of any system in term of claims exchanges but leaving the details on how those are implemented to the specific technologies. The necessary level of abstraction is achieved by taking advantage of open, interoperable protocols such as the WS-* family of specifications.

The IdM describes three roles:

- ***Subject.*** The subject is the subject entity we mentioned in the claim definition. It is whoever (or whatever) needs to be identified in a transaction.
- ***Relying party (RP).*** The RP is the resource that requires authentication before being accessed. Examples of RPs include web sites and web services. RPs derive their name from the fact that they rely on IPs for obtaining claims about the subject they need to identify.
- ***Identity provider (IP).*** The IP is the authority entity we mentioned in the claim definition. An IP possesses knowledge about the subject, and can express it in the form of claims: Any RP that trusts that particular IP will be able to rely on those claims for making authentication and authorization decisions on subjects. Note: Often the IP will use an STS for issuing claims in the form of tokens; that does not mean that every STS *is* an IP. An STS is a tool that the IP uses to get its job done.

Architectural Patterns of Claims-Aware Solutions

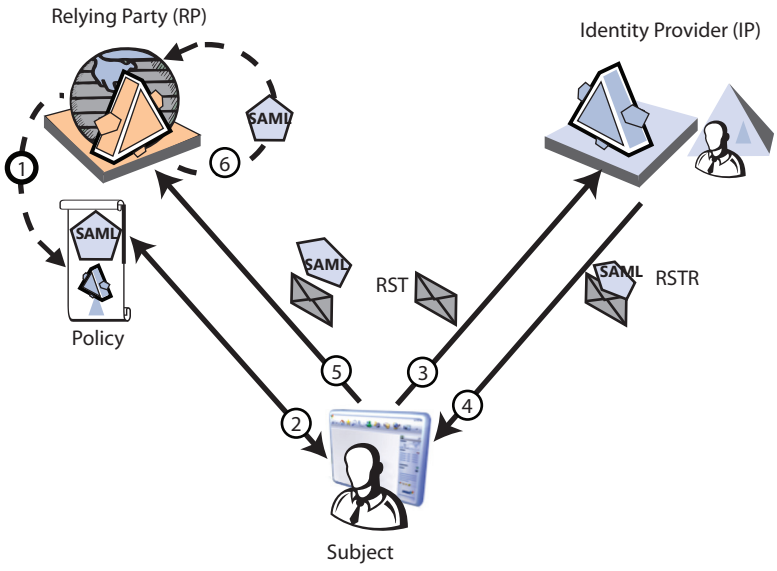
The basic aforementioned concepts constitute the basis for a new physic of authentication, and as such they can be combined to describe any system and transaction. In this new world, we can finally separate two functions traditionally conflated into one: obtaining information about the caller (via claims) and verifying if it is a returning user (via credentials). Thanks to this separation, we can now choose which function fits best our scenario; for more details see Resources: Vittorio Bertocci's blog The TAO of Authentication.

There are certain patterns that are especially useful for describing common scenarios. To follow, I will describe the three most common of those patterns, along with some of the advantages that the claims-based approach brings to those scenarios with respect to traditional practices. All the patterns are suitable both for on premise and cloud architectures.

The Canonical Pattern: Subject-IP-RP

This pattern describes the minimal situation in authentication management: a subject wants to access a restricted resource (the RP). This happens when a Windows user tries to access a network share, when a smart client invokes a secure web service, when a web surfer wants to browse restricted content, you name it (see Figure 2).
The key steps are as follow:

Figure 2: The Subject-IP-RP canonical pattern. The numbers refer to the text.



1. (out of band) The RP publishes its requirements in the form of a policy. Those requirements include
    a. A list of claims
    b. The IPs that the RP trusts as sources of those claims
    c. Details about the specific authentication technologies that the RP can use; for example, the token formats that the RP understands
2. The subject reads the RP policy and verifies whether it can comply with it. In practice, it means verifying if the subject can obtain a suitable token from any of the IPs that RP trusts.
3. The subject invokes a suitable IP, requesting a token that complies with RP policies. In practice, it usually means sending an RST message to the IP's STS.
4. The IP receives the RST and authenticates the subject. If the subject is known, the IP will retrieve the required claim values, package it in a token, sign it, and send it back to the subject.
5. The subject receives the token and uses it for invoking RP.
6. RP extracts the token from the invocation message and examines it: Is it signed by one trusted IP? Does it contain the right claims? If the check is positive,
    a. the claims values are used for feeding some access control logic
    b. If the access control logic is satisfied, the claim values are made available to the resource itself and access is granted

For the purposes of this explanation, let's assume that the resource is a web service. The pattern above exhibits many good properties.

Authentication externalization

Using tokens and claims lifts from the resource the burden of writing any explicit identity management code. The same infrastructure that takes care of publishing RP policies can also perform operations such as deserializing tokens, checking signatures, and making claims values readily available to the resource developer regardless of the token format that was actually used on the wire. The infrastructure can also perform some authorization decision based on claims values, reducing further the tasks that the developer needs to worry about.

This is an advantage that holds for any scenario, but especially for the cloud where the hosting technology is one of the variables. Using claims makes resources truly portable, by decoupling them from the details of the infrastructure that will host them and the authentication technology that will be available at runtime.

Resource-level policy

Being able to specify policies at the resource level allows for very agile deployments, fine-grained control, and dynamic negotiation of the authentication technology of choice. It allows for establishing a management strategy that will work regardless of the hosting environment, thanks to the use of interoperable standards. It decouples the resource itself from the execution environment, making it much easier to move the resource around (including to the cloud).

Autonomy

Every resource specifies its requirements in an autonomous fashion: It is the subject that matches those with its own capabilities. The interaction is an emergent property of the combination of the two. Both parties can change independently, and the set of subjects that can access the resource is defined solely by the ability to comply with the requirements as opposed to some out of band or infrastructural dependency. The system is very robust, since everybody needs to worry only about its own requirements and capabilities. Onboarding users is very easy, since it does not require any explicit negotiation or arrangement.

Centralization of some authentication and authorization logic

In this pattern, the IP performs all the attributes retrieval. The RP receives what it needs in the form of a token, without needing to query other systems. This is obviously an advantage for those attributes to which the RP would not have access (i.e., Airline A asks for the subject's frequent flyer status with partner Airline B), but it is also a great means of centralizing attribute retrieval logic in a single place. Not only does it reduce the number of endpoints that need access to attribute stores, with advantages for manageability and security, it can also reduce the number of accesses itself, since once the subject obtains a token it can cache and reuse it until it expires without getting back to the IP. The IP can also be used for authorization decisions, which can be expressed via claims as well; however, this can happen only when the IP has a tight relationship both with the subject and the resource. While this happens in important scenarios, such as when the IP represents a directory and the RP is one of the resources it manages, in the generic case no relationship can be assumed between RP and IP.

This is key to many cloud scenarios, in which resources need to rely on external IPs for verifying information about possible users; however, it is also very useful in more agile versions of classic federation, in which users of a partner organization can be represented by technology- and platform-agnostic tokens, and for crossing boundaries of any kind.

The Claims Transformer Variation: Subject-IP-Claims Transformer-RP

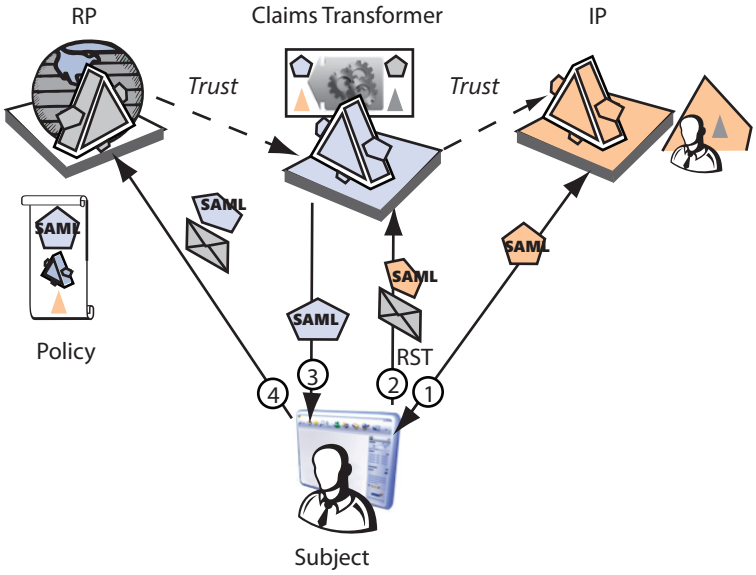
The pattern just described can be applied to a wide range of scenarios, from consumer (frequent flyer with Airline A accessing the website of partner Airline B) to enterprise (almost any Kerberos scenario can be modeled that way, with the added bonus of technology independence).

However, there are situations in which certain constraints may prevent the application of the pattern as it is:

1. **Indirect trust.** A RP may not directly trust the IPs that are in a relationship with the subject, but there could be a chain of inter mediate authorities that can be traversed for brokering trust. If you want to board a plane, you won't be able to do so using just the documents already in your wallet; the employee at the gate will trust only a boarding pass issued by the proper airline. How ever, you can use the documents already in your possession (pass port or driver's license, issued by the government) at the check-in counter to obtain the boarding pass (issued by the airline).
2. **Claims format.** The RP may not be able to understand the claims in the format issued by the IPs that are directly available to the subject. Sometimes it is a simple matter of format (my Italian ID says "nato il" instead of "birth date," a bartender in the U.S. would not be able to extract the information he needs even if it's there), other times the claims required by the RP are the results of some processing of the raw information received (the RP may need a "CanDrink?" claim, which may be the result of processing "age" and "Nationality").

The new constraints can be easily addressed by adding a new element, which we will call a *claim transformer*, that can process a token

Figure 3: Basic claims transformer pattern. The Subject (1) obtains a token from the IP and (2) uses it to get a token from the claims transformer. The R-STS of the claims transformer processes the incoming claims and (3) issues the subject a new token. The subject (4) uses the new token to invoke the resource.



before it reaches the resource and can be leveraged by the RP to broker trust and convert the incoming claims in a more suitable format. Between the subject and the RP, there could be an arbitrarily long chain of claim transformers. Dynamic negotiation of policy can automatically "choose" a path, if available, without any need to plan it at a high level.

The best artifact for implementing a claim transformer is, again, the STS. Instead of populating the issued claims from its own data sources, the STS used by the claim transformer will mainly manipulate the incoming information. Since it is often run by the same entity that owns the RP, as in the classic federation case, this kind of STS is usually referred to as Resource STS, R-STS or RP-STS. I recommend reading my blog for more details (see Resources).

This variant presents some powerful advantages and, in our opinion, will emerge as the dominant pattern for identity management in distributed systems (see Figure 3).

Resource clustering

Claims transformers can offer a granularity continuum between single resources and the directory at the enterprise level. An R-STS can be used for corralling multiple resources with similar requirements. It simplifies the policy management of single resources, moving the complexity of brokering trust and processing raw claims in a single point but without requiring it at the enterprise level. An R-STS can be a virtual boundary protecting legacy resources that may not play too well with the rest of the network. Those are just few examples of the expressive power of claims transformers (see Figure 4).

Authorization decision point

While an IP is usually an independent entity, very often the R-STS has some knowledge of the resources it fronts. Such knowledge may enable the R-STS to perform authorization decisions. The incoming claims can be processed together with the requirements of the resource the subject is trying to access, and the result may be a decision about whether the subject holds the necessary privileges. Such a decision can be expressed in different ways:

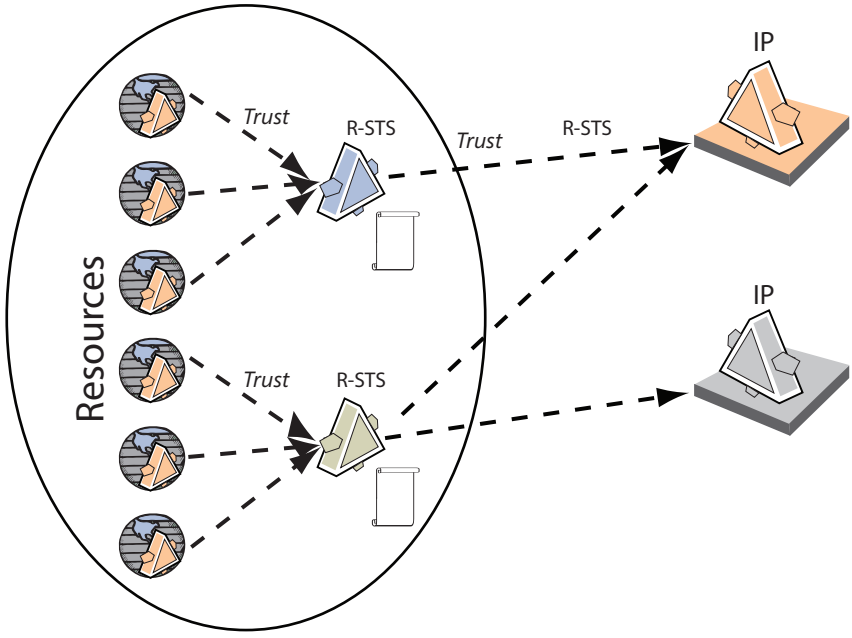
- If the subject does not hold any privileges, the R-STS may simply refuse to issue a token and block the invocation. In this case, it would enforce authorization.
- If the subject holds some privileges, the R-STS may formalize those authorization decisions in the form of claims. The RP is hence relieved from the need to run authorization logic, and will just enforce the directives it receives from the R-STS in the form of claims.

The use of tokens, claims, and IPs made authentication externalization possible; the R-STS can, in certain cases, help to do the same with authorization decisions.

Delegation

There are many resources that are not claims-aware: legacy systems, resources managed directly at the directory level, and so on. Often some investments have been already made to manage access to those resources, such as assigning ACLs and creating groups. The identity of a subject calling a front-end application via claims-based security would not be suitable to leverage those investments; if the front-end application could obtain a directory identity on behalf of the caller, however, the

Figure 4: R-STSes can be used to cluster multiple resources with similar policy requirements, simplifying trust management and providing a means to control the granularity of externalized authentication and authorization logic.



problem would be solved. In fact, the value of obtaining tokens on behalf of somebody else is clear even when the issued tokens remain in the realm of claims.

Claims and Cloud

The beauty of what I've described so far is that it adapts to *any* loosely connected system; and what is the cloud, if not the ultimate distributed system? When you reason at cloud scale, you deal with a world where every entity is managed independently: claims, tokens, trust, and identity metasystem roles can help you make the best of whatever little information you may have about the relationships among the parties you deal with. By describing a claims-based solution, we killed two birds with one stone, giving you tools that can be applied on premise and on the cloud alike.

We still know too little of how things will evolve to give detailed guidance on cloud scenarios. However, with some working hypothesis, we can certainly highlight the aspects of claim aware identity management that are especially well suited for the cloud.

Access Control via Claims Transformation

We have seen that a cloud provider may offer services such as storage and compute, messaging, and integration. If you recall what we described in the claims transformer section, you will see that an obvious access control strategy for the cloud provider is to offer an R-STS and have the resources trust it. Every tenant would then be able to govern access control simply by manipulating the claim transformation rules of the R-STS.

Let's say that you are an ISV, and you want to deploy your services at a certain cloud provider. You decide on the set of claims your services will accept, and you set your services to trust the provider's R-STS. Your access control strategy is already in place, regardless of who calls your services! When onboarding a new customer, you simply have to define how to map his claims to yours, by setting some rules at the R-STS level. This can be incredibly agile and easy to maintain (see Figure 5).

Externalizing Authorization

We have seen how an R-STS can conveniently externalize and aggregate authorization decisions; this is, of course, valid in cloud scenarios as well. In fact, there are cases in which this can be pushed further to include authorization enforcement. Getting back to the ISV example:

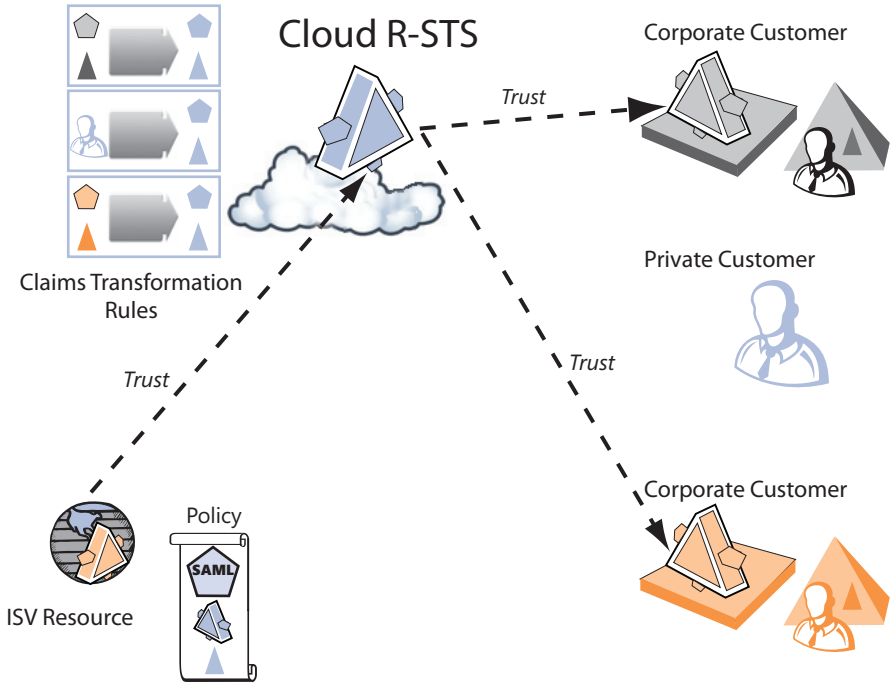
If the services are exposed by taking advantage of some messaging offering, the dispatch infrastructure itself can take care of enforcing authorization decisions by examining the authorization claims even before the message is routed to its destination. This relieves the ISV from yet another burden: If the services are not exposed using the provider's messaging infrastructure, the authorization decision claims have to be enforced by adding some processing pipeline in front of the resource. (For more details, see Resources: Vittorio Bertocci's blog, claims types.)

Managing Relationships

With the model described, an ISV can represent a customer relationship as a set of rules. This is much more agile than creating a federation relationship by traditional means, directory to directory. While coarser to manage, it also avoids the burden of extra management that is necessary in peer partners relations but would often be overkill for vendor-customer relationships.

The model also allows for accommodating individuals and customers without advanced identity capabilities. The ISV services will always see tokens issued by the R-STS, regardless of whether the customer authenticated with a token issued from his own IP or with simple, one-off username and password.

Figure 5: An R-STS in the cloud is a very powerful tool for managing access control. The resource always handles tokens and claims in the right format, from the same trusted source; the R-STS takes care of handling trust management and credential verification from different sources, using rules to perform the necessary transformations and isolating the resource from changes and unnecessary complexity.

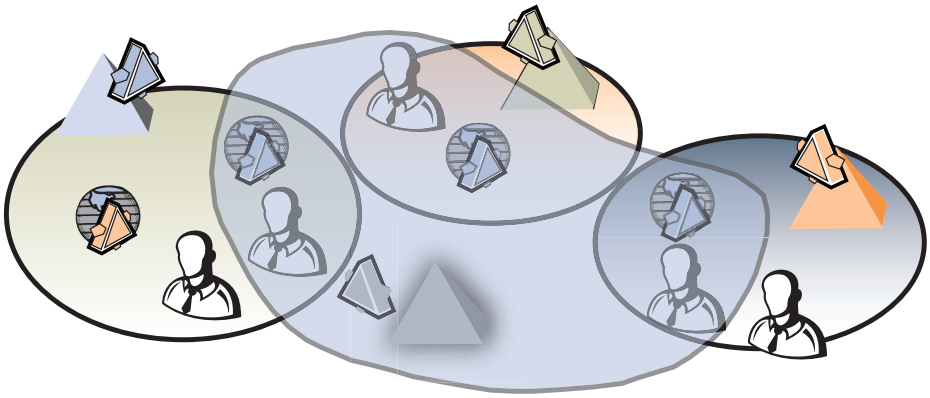


The Future

The shift toward the cloud will happen: the industry is fairly quick to acknowledge that it is not a matter of if, but when. Predicting the future is always a dangerous game, but the cloud is just too exciting for not venturing in a plausible development that, despite looking like an identity utopia today, is in fact perfectly plausible thanks to principles described so far.

Today, access control is often constrained in the railways of organization structure, with privileges reflecting rank rather than function. The number and the caliber of companies and open-source initiatives participating in interop events

Figure 6: Policies, claims exchanges, and virtual directory management make possible the semi-spontaneous emergence of virtual organizations (center).



(see Resources: RSA 2008 OSIS Interop event) and integrating support for claims in their lead products suggests that claims-based programming is on its way to becoming mainstream. Once that happens, it is easy to imagine how identities and access control structures may be corralled around tasks rather than organizations. Subject, resources, roles, and authorities from many different companies could all be described in terms of a specific project, with privileges assigned not according to organization rank but reflecting the function performed in the context of the project itself. Virtual organizations could emerge for the duration of the task, and dissolve once the mission is performed. That would also allow for templating of cross companies efforts, enforcing best practices and improving predictability (see Figure 6).

For this to happen, claims are a necessary, but not sufficient, condition: Emerging technologies such as virtual directories will play a key role as well (see Kim Cameron's blog, www.identityblog.com, for more details).

Call to Action

The claims-based approach is great for traditional and cloud scenarios alike. The difference is that while in traditional scenarios you can sometimes use a great deal of extra resources to get away with bad strategies, once the cloud is upon us, it won't be that easy. If you want to prepare yourself and your organization for the upcoming wave, here there are some things you can do:

- Experiment with web services and security, if you are not already doing it
- Experiment with claims-based programming, taking advantage of the Beta release of "Zermatt" Developer Identity Framework (http://go.microsoft.com/fwlink/?LinkId=122266)
- Consider running pilots within your organizations. For example: create an IP for your employees

Here I could offer the proverbial "we barely scratched the surface of the topic," and that would be true. Instead, I will say that claims are the cornerstone of exciting developments in identity management for traditional and cloud scenarios alike. I invite the geeks among you to enjoy this special transition moment in which much of those interactions are still evident, and I reassure all the others by foreseeing that most of the details will likely sink in the infrastructure and be made invisible by new breeds of tools.

Resources

Vittorio Bertocci's blog

- "R-STS": http://blogs.msdn.com/vbertocci/archive/2007/09/24/the-resource-sts-r-sts-rp-sts-a-sts-the-other-face-of-token-issuing.aspx
- "claims types": http://blogs.msdn.com/vbertocci/archive/2008/05/05/claim-types-a-coarse-taxonomy.aspx
- "The Tao of Authentication" I, II and III:
  http://blogs.msdn.com/vbertocci/archive/2008/02/20/the-tao-of-authentication-part-i.aspx
  http://blogs.msdn.com/vbertocci/archive/2008/03/10/the-tao-of-authentication-part-ii.aspx
  http://blogs.msdn.com/vbertocci/archive/2008/03/11/the-tao-of-authentication-part-iii-last.aspx

Kim Cameron's blog

- "virtual directories": http://www.identityblog.com/?p=983

"Understanding Windows CardSpace", Chapter II (Addison Wesley 2008), on MSDN
http://download.microsoft.com/download/d/3/9/d399b17c-0958-49b7-8559-bea1956c2c5b/0321496841_02.pdf

RSA2008 OSIS Interop event

- logos: http://web3id.org/files/RSA 2008 Handout/I3-Handout.jpg
- OSIS interop: http://osis.idcommons.net/wiki/Main_Page

WS-Security
http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

WS-Trust
http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf

About the Author

Vittorio Bertocci, senior architect evangelist, Cloud Services Evangelism, Microsoft Corp, helps large enterprises leverage new technologies. After a few years in the Italian Microsoft Services, he moved to the U.S. headquarters, where he has spent the past three years helping customers deploy solutions based on identity and access management, SOA, and services. He currently focuses on the identity and access aspects of cloud computing; *Understanding Windows CardSpace*, the book on user-centered identity he coauthored, was published last January. Vittorio maintains a popular blog at http://blogs.msdn.com/vbertocci.

Enterprise Identity Synchronization Architecture

by Mike Morley and Barry Lawrence

Summary

With an increasingly diverse mix of hosted, product-based, and custom solutions found in the modern corporate environment, reliably ascertaining the identity of employees across the enterprise can be difficult, and is often left to manual processes.

This article will discuss a system and approach that accomplishes identity synchronization, developed by BST Global, a Microsoft Gold Partner, and deployed at Matrix Solutions. The discussion will focus on the first phase of the project describing the architecture of the Employee Identity Synchronization system, which synchronizes employee identity from Ceridian's hosted HR Payroll, BST Global's Enterprise PSA, Microsoft's Active Directory, and Exchange Server.

Each individual system contains domain-specific data regarding the employee. These systems also use their own terminology to describe the data — terminology that may not be applicable to the overall corporate understanding of those data, or relevant to other system domains. HR/Payroll systems, for example, typically contain extensive information that only HR requires to produce payroll and manage the employee, whereas PSA (Professional Service Automation) systems contain employee data that relate to projects, employee charge rates as applied to projects, time, and expense data.

To effectively synchronize the relevant data across systems, a specification that defines the employee data in terminology relevant to the corporation, regardless of system domains, is required. To address the challenge, Matrix Solutions Incorporated (Matrix) has adopted the use of a SOA engine called the BST Freedom Exchange (BSTFX), a services-based workflow engine that facilitates the corporate definition and synchronization of federated keys and employee identity across systems. The workflow engine has the ability to translate data from system domains to the corporate domain and handles the process of populating the corporate definition from the individual systems, managing changes in those data, and synchronizing the resulting data back into the subscribing systems.

Scenario

Matrix Solutions Inc. is a rapidly growing Environmental Engineering consulting firm based in Calgary, Alberta Canada that primarily serves the upstream oil and gas industry. The company's rapid growth from 100 to 250 employees in 4 years makes managing employee identity a challenge.

As with many modern medium-sized firms, Matrix makes use of a 'best of breed' approach to primary business systems rather than attempting to use one monolithic ERP (Enterprise Resource Planning) system to manage core business information. While this approach increases efficacy of each system within a particular business domain, the fact that each system will have its own native database and associated identity management framework exacerbates the issue of identity management.

Matrix uses a mix of commercial hosted onsite, commercial hosted offsite and custom developed systems to run its business. Each has its own database and domain-specific method for storing and describing business data.

Identity is at the heart of every system, since it defines the user experience within each domain, and is the means through which security is governed. Security is dependent on proper knowledge of not only identity for authentication but also for authority and access to the correct information within the system in question. Keeping employee identity current in all core systems is therefore critical to the smooth and secure operation of the business.

Synchronization of identity between systems at Matrix traditionally has been addressed by manual methods using full-time staff responsible for each system. In some cases, batch data reconciliations between systems were conducted by temporary staff to bring systems back into sync, resulting in identity disconnects between systems and data inconsistency. One specific example of this involved an employee who had returned from an extended leave having two separate identity accounts created for them. Because of the manual methods used to synchronized identity, it was not until sometime later that the error was detected, creating a great deal of extraneous work in reconciling and resynchronizing the systems that used the information.

Matrix operates in the dynamic, fast moving, and cost-competitive consulting business sector. The consulting business model is particularly sensitive to overhead costs — each additional overhead resource required to support the business has an impact on profitability. Keeping the number of administrative resources at an optimal level to support the operation of the business while ensuring they are being used to maximum effect is crucial to the success of the company. Having administrators dedicated to keeping systems synchronized by doing data-entry is not in the best interest of the organization or the resource.

Goals

Based on the business challenges, the following key objectives were established for a key IT initiative:

- Reduce the friction of information flow throughout the organization
- Maximize use of knowledge and administrative assets wherever possible
- Replace repetitive manual business processes with automated processes
- Get the correct information to people when they need it in the form they need it

In addition, the following architectural guidelines were established to govern the technical solution:

- Create a set of standard definitions that describes core business objects in terminology Matrix understands (domain-specific language).
- Create an extensible integration framework with the ability to capture critical information from many sources and build a cohesive view of the current state of these Matrix Business Objects.
- Use the Matrix Business Objects to keep subscribing business systems synchronized and up to date using a rules-based workflow engine that can route and handle exceptions.
- Construct a standard set of services using the Matrix Business Object definitions that can be reused for other business purposes.

Solution Approach

Matrix engaged with BST for consulting expertise and best practice guidance on systems integration. The project team consisted of Mike Morley and Adrian Brudnicki from Matrix Solutions, and David Hebbler from BST, working over the course of a six-week Enablement Program that was established to provide Matrix with a hands-on, guided approach leveraging the BSTFX platform:

- Week 1: System design, architecture, and hands-on coaching with the Freedom Exchange integration platform offsite at BST.
- Week 2: Codevelopment and guided implementation onsite at Matrix to establish baseline interfaces and system classes.
- Week 3 – 5: Independent programming by the Matrix development team to flesh out the working details of the solution.
- Week 6: Onsite solution review, QA testing, and deployment readiness.

System Architecture

Rather than a single monolithic system to manage core business data, Matrix uses best-of-breed systems for each business domain. To limit scope, three key systems were selected for the first release:

- Ceridian's HR/Payroll web system
- BST Global's Professional Services Automation (PSA) System
- Active Directory/Exchange for network authentication and email

These systems were chosen because they deal with the business data at the root of the entire corporation and have the largest reach in terms of impact to the user community. The Employee Identity Synchronization system's architecture was designed to be extensible to allow for inclusion of other existing and future systems.

Ceridian

Ceridian's HR/Payroll is a human resources and payroll management system. It contains extensive amounts of information regarding the employee required by Human Resources to manage the employee's life with the organization. It is the primary system for managing employee identity, since an employee must first be defined in the HR/Payroll system before they can exist in any other corporate system. From an architectural perspective, it is the 'Master System' for a number of the key Matrix Employee properties such as the employee's ID, first/last name, company, team, address, and status assignments.

Ceridian also contains information regarding employee benefits, dependant information, health and welfare data, etc. Much of the data stored in Ceridian regarding Employee identity goes beyond the reach of the other systems, and therefore, from the perspective of the overall Matrix organization, is extraneous.

Because the system is hosted by Ceridian, and does not offer a web service API, extracting the data from the system requires traditional-style file-based integration.

The data extract from Ceridian contains highly sensitive information such as employee names, addresses, and payroll data, so security was a critical architectural consideration.

Ceridian's consulting services division constructed a CSV style file export of common properties derived from the data analysis exercise. The data file is encrypted and transmitted to a secure FTP server hosted at Matrix specifically setup for the single purpose of receiving the extract file.

BST Enterprise

BST Enterprise is a Professional Services Automation (PSA) system that provides financial and project management tools for firms based in the Architecture and Engineering business. BST makes extensive use of employee identity information for tracking employee time and expenses, responsibilities and assignments on projects, and authorization to features within the BST system.

BST Enterprise therefore derives core identity information from Ceridian, but contains a large volume of information about the employee from a project perspective.

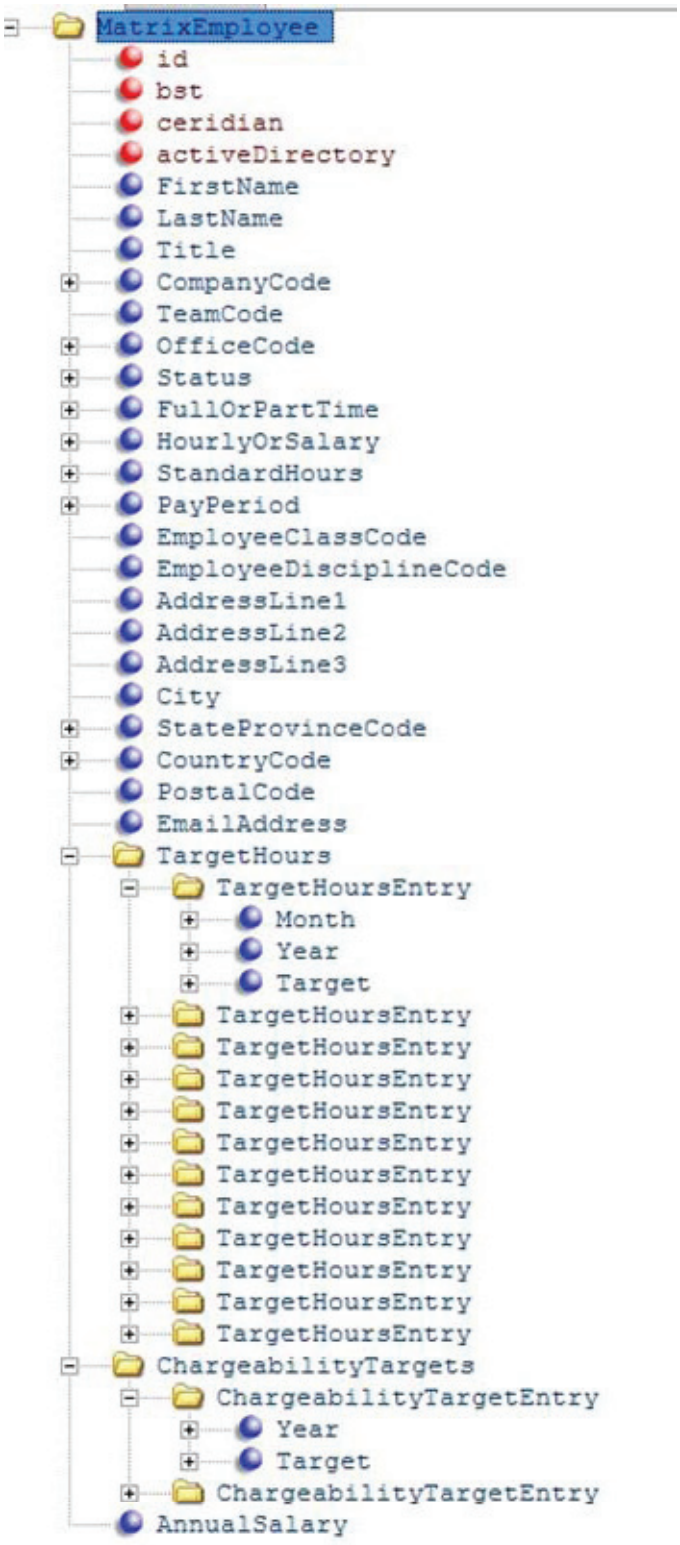
Ceridian views salary data from an annual context, and contains all sorts of information regarding reasons for a salary increase — whereas BST needs the salary information on an hourly basis in order to calculate and track project progress on a cost and budgetary basis. BST, therefore, is the owner of the properties associated with hourly project cost and charge rates. BST needs to know about salary increases in terms of the hourly cost rate, but does not need to know why these increases took place.

BST provides a .NET SDK (Software Development Kit) called the BST Freedom Framework™ that enables developers to programmatically interact with information in BST Enterprise in real-time. The SDK uses a document-centric model to define its business objects, and has strict validation rules about what is required to create a valid BST Employee.

"SYSTEMS TYPICALLY WILL TAKE ON THE CHARACTERISTICS OF THE BUSINESS DOMAIN THAT THEY ARE DESIGNED TO SERVE. THIS HAS SEVERAL ARCHITECTURAL IMPLICATIONS FROM THE STANDPOINT OF IDENTITY MANAGEMENT."

Therefore, key fields from BST are mapped into the Matrix Employee definition. In addition, to capture baseline information required by BST but not stored elsewhere, a 'default' baseline employee was established as a root mapping point.

Figure 1: Matrix employee business object XML



"THE EMPLOYEE IDENTITY SYNCHRONIZATION SYSTEM'S ARCHITECTURE WAS DESIGNED TO BE EXTENSIBLE TO ALLOW FOR INCLUSION OF OTHER EXISTING AND FUTURE SYSTEMS."

Active Directory/Exchange

Active Directory is used at Matrix as the primary authentication system, and Exchange Server is the email system.

Because Matrix is very dynamic, employees will move about in the organization frequently. It is important that an employee's status with the company is kept up to date in Active Directory and Exchange.

Since an employee's team as defined by Ceridian corresponds to an Active Directory security group, and email groups in Exchange, it was possible to create a set of rules that map the employee to the correct OU and Email group automatically.

Datamart

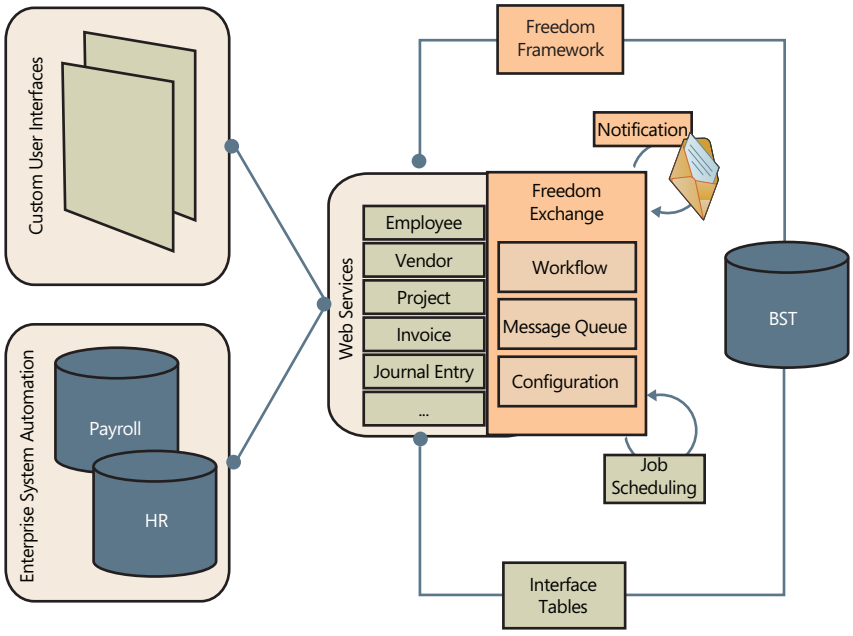
The Datamart is a custom database that acts as the hub for all Employee Identity Synchronization workflow and mapping, including:

- A staging area for Ceridian import data (the Datamart is used as a façade to represent Ceridian as a data source).

Figure 2: Matrix employee property ownership map

Source	Owner	Matrix.Employee.Xml
		<MatrixEmployees>
		<MatrixEmployee
MAP.GlobalID		id="{guid}"
MAP.Ceridian		ceridian="123"
MAP.BST		bst="00123"
MAP.ActiveDirectory		ad="{guid}">
Ceridian.FirstName	Ceridian	<FirstName/>
Ceridian.LastName	Ceridian	<LastName/>
Ceridian.Title	Ceridian	<Title/>
Ceridian.Company	Ceridian	<CompanyCode/>
Ceridian.Organization	Ceridian	<TeamCode/>
Ceridian.Location	Ceridian	<OfficeCode/>
Cerdian.Status	Ceridian	<Status/>
Ceridian.TimeStatus	Ceridian	<FullOrPartTime/>
Ceridian.PayPeriods	Ceridian	<HourlyOrSalary/>
Ceridian.StandardHours	Ceridian	<StandardHours/>
Ceridian.RateStatus	Ceridian	<RateStatus/>
EmployeeChargeability.Target	Datamart	<ChargeabilityTargets/>
Ceridian.EmpClassification	Ceridian	<EmployeeClassCode/>
Ceridian.EmpDiscipline	Ceridian	<EmployeeDisciplineCode/>
Ceridian.ProjectRates	BST	<ProjectRateAnnual/>
Ceridian.RegAmount	BST	<RegularAmountAnnual/>
Cerdian.PayrollRates	BST	<PayrollRatesAnnual/>
MappingRateCalculationByClass.RegularAmount	BST	<RegularAmount/>
MappingRateCalculationByClass.PeriodSalary	BST	<PeriodSalary/>
MappingRateCalculationByClass.OvertimeAmount	BST	<OvertimeAmount/>
Ceridian.Address1	Ceridian	<AddressLine1/>
Ceridian.Address2	Ceridian	<AddressLine2/>
Ceridian.Address3	Ceridian	<AddressLine3/>
Ceridan.City	Ceridian	<City/>
Ceridan.StateProvince	Ceridian	<StateProvinceCode/>
Cerdian.Country	Ceridian	<CountryCode/>
Ceridian.PostalCode	Ceridian	<PostalCode/>
Cerdian.Phone	Ceridian	<PhoneNumber/>
Ceridian.EmailAddress	ActiveDirectory	<EmailAddress/>
Ceridian.Salary	Ceridian	<AnnualSalary/>
	Datamart	<TargetHours>
	Datamart	<TargeHoursEntry>
	Datamart	<Month/>
	Datamart	<Year/>
	Datamart	<target/>

Figure 3: BSTFX overview



- Mechanism through which the Enterprise Identity Synchronization system tracks new Matrix Employee objects through a key table. The key table also defines which systems are subscribing to the synchronization.
- Central repository to store overall state of a Matrix Business object as derived through querying all subscribing business systems.
- Holds any information specific to Matrix that is not handled by other system. Examples of employee information not provisioned by other systems are the Employee Chargeability and Hours targets. These targets are key performance indicators (KPI) that allow employees and business managers to track how employees are doing with respect to goals set during performance reviews. These KPI are unique to Matrix, and as a result, none of the commercial offerings have facilities for managing them.

Matrix business object XML definition

Systems typically will take on the characteristics of the business domain that they are designed to serve. This has several architectural implications from the standpoint of identity management.

To leverage specialist knowledge associated with the business domain from a user experience perspective, systems will define identity objects using terminology associated with that domain. As a result, even identity information that is common across systems will be defined with different terminology and often on a different basis than is required by systems operating in a different domain.

Each system contains a great deal of information that is specific to the domain in which it resides, and has no purpose outside that domain. By going through each system in detail, we derived a definition of the union of identity that was outside each system domain. The resulting properties make up a business object that is called a 'Matrix Employee'. An XML-based standard was created to describe the Matrix Employee (Figure 1, see page 35).

A map defining the system owner of the properties (i.e. the 'master' property) and its name in the local system domain was established (Figure 2, see page 35). To limit the scope, the first iteration of the Employee Identity Synchronization system does not have a mechanism for reconciling differences between properties in different systems – therefore, a 'master system' is established for each property, which tells the mapping engine which system to use as the source for a given property when synchronization takes place. A more sophisticated handling/reconciling system is planned for a later release of the system.

Implementing identity synchronization

BST Freedom Exchange (BSTFX) is both an extensible service framework and a service host for implementing and managing enterprise services based on SOA (service oriented architecture) best practices and principles (Figure 3).

BSTFX exposes a set of integration interfaces and APIs that give developers the freedom to define and implement completely customized

Figure 4: Settings in BSTFX console

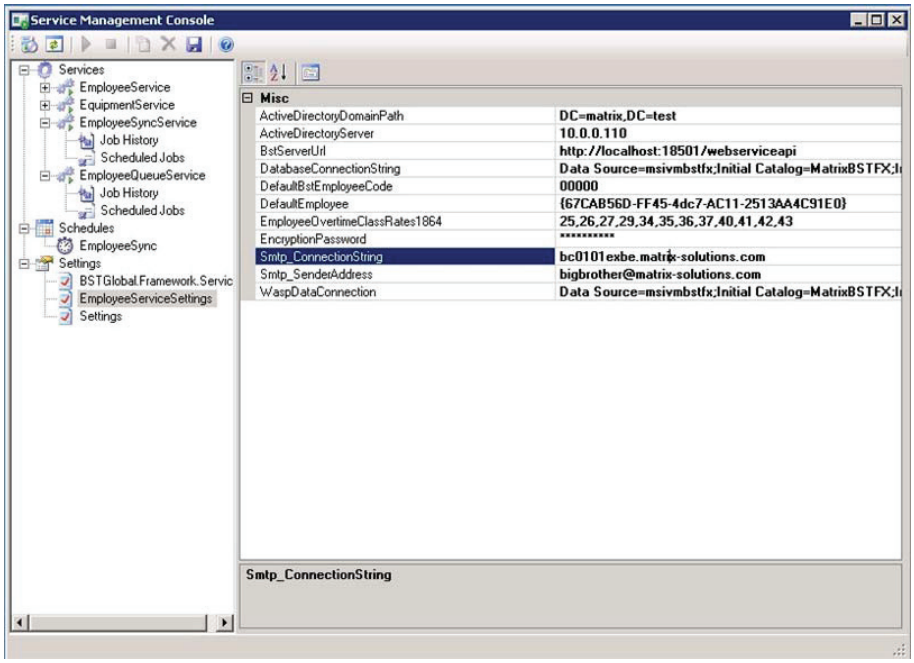
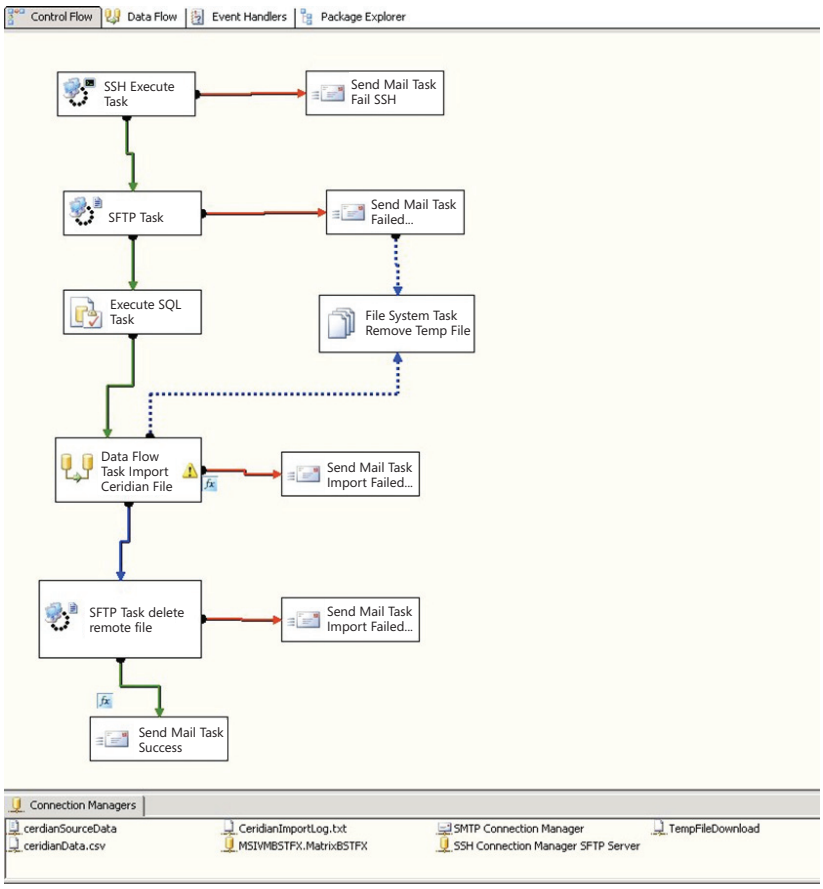


Figure 5: SSIS integration package diagram



services, service contracts, and workflows using WCF (Microsoft Windows Communication Foundation) and WF (Windows Workflow Foundation).

The Employee Identity Synchronization system is implemented as a set of BSTFX services, which are hosted by the BSTFX Runtime. The Runtime is responsible for registering and managing the services (Threading, Memory Management, Message Queuing, Activity Monitoring, etc.), while the Service Console provides administrators with the ability to manage and monitor registered services, including the ability to run and schedule jobs utilizing the custom service interfaces, inputs, and settings established by the developer (Figure 4).

The Enterprise Identity Synchronization process begins with Ceridian, which produces a nightly export of employee identity data to the Matrix SFTP site. Data is picked up by a scheduled SQL Server SSIS (SQL Server Integration Services) package, which is responsible for taking the data from the SFTP server, decrypting onto that server, and then downloading the file to a secure, temporary staging location on the SQL server, where it is imported into the Datamart (Figure 5). As an added security measure, the import process re-encrypts the salary information. Email alerts are sent out on success/fail of the import process.

Identity synchronization involves taking the Matrix Employee XML object and asking each of the subscribing systems to fill the properties they own, validate them against the subscribing systems, and finally commit the final data back to each system. Doing this effectively builds a full picture of the current state of the Employee's identity from all systems, and then commits the latest, composite version back to each subscribing system, thus bringing the systems into sync (Figure 6).

Interfaces

To isolate the specific requirements of each subscribing system from the overall synchronization pattern, the system is divided into two primary interfaces:

- IMatrixDataSource — describes the requirements for a valid synchronization data source and provides the basic methods for an object to fill its properties from the underlying data source, validate its properties against the Matrix

Figure 6: Enterprise Identity Synchronization structure

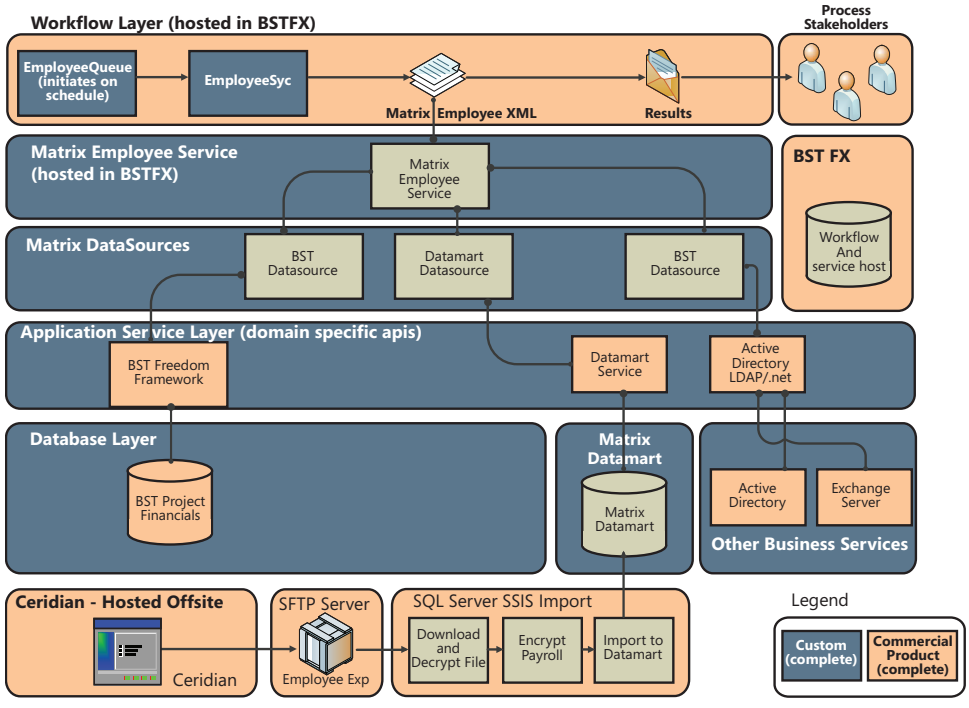


Figure 7: Employee Service Class diagram

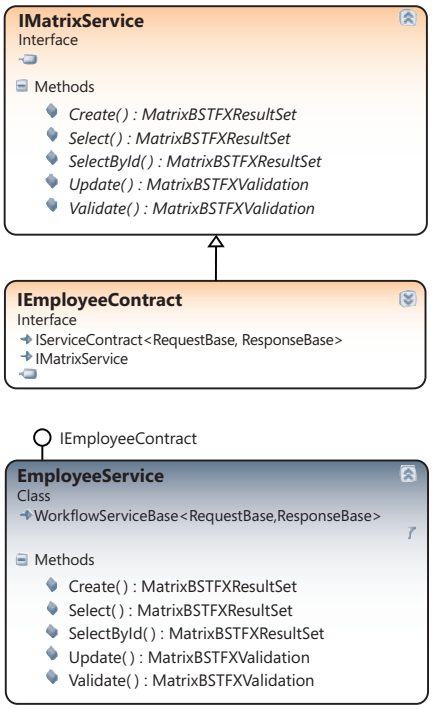
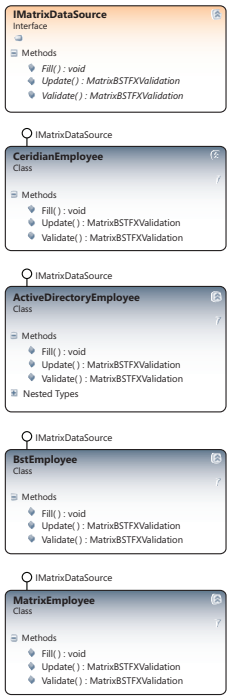


Figure 8: DataSource classes



process calls the EmployeeSync service for each employee, passing the unique identifier assigned to the employee's identity to the service.

Business services

EmployeeSync loads the Matrix Employee Business Object XML definition, and then calls the `SelectById()` method on the Employee service, which passes the XML to the DataSource layer for each of the systems registered in the Employee Service.

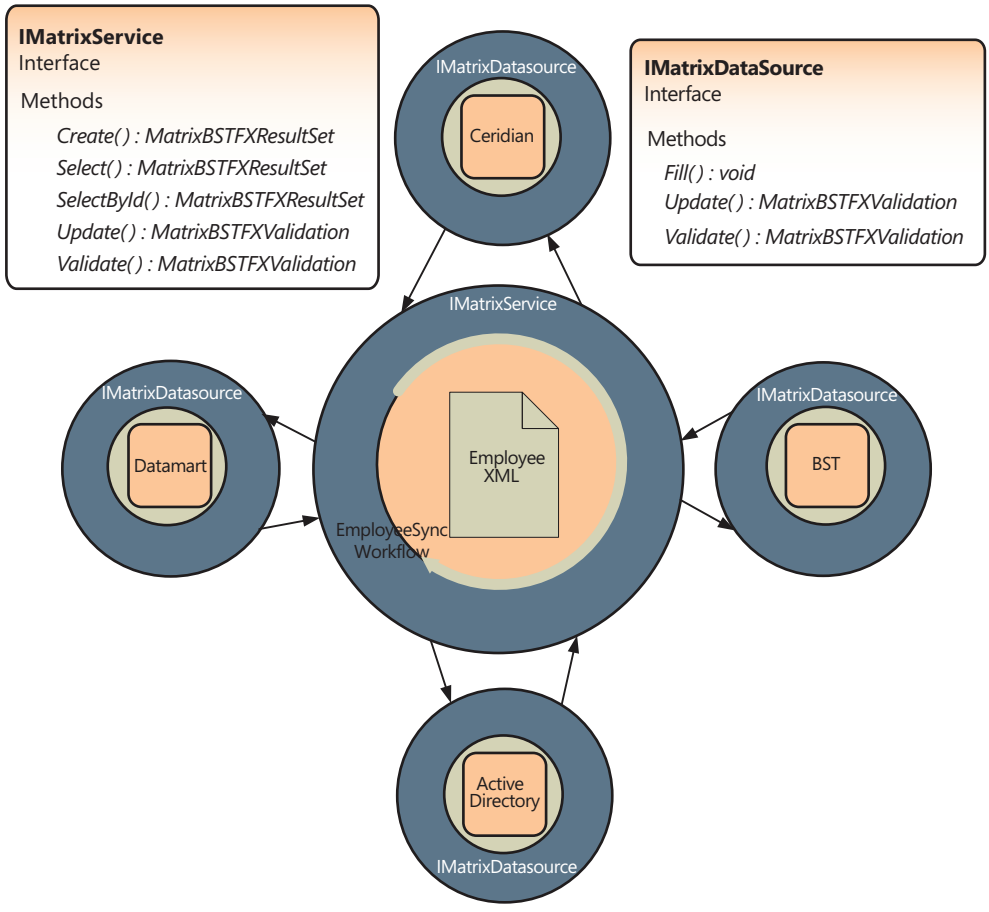
Once the `SelectById()` workflow step is complete, the EmployeeSync workflow moves to the next step in the process and invokes the Update() method of the Employee service using the Matrix Employee XML derived from the participating systems.

The `Update()` method first runs a `Validation()` pass on all the Matrix Employee XML against all registered systems, which returns a MatrixBSTFXValidation object. If the validation passes cleanly for the employee identity record, the `Update()` method of the Employee Service calls the corresponding `Update()` method on

Employee Business XML Object, and update any information that is mapped and synchronized from subscribing systems back into the local system domain.

- IMatrixService — describes the standard methods that must be implemented to define a valid Matrix Service that can be implemented to synchronize multiple data sources through workflow orchestration.

Figure 9: Synchronization interfaces



Workflow services

The EmployeeQueue is exposed as a top-level workflow service that is managed by the BSTFX scheduler. The EmployeeQueue workflow process is responsible for iterating through the Ceridian employee identity list in the Datamart, determining which of the employees are new, generating a unique identifier for each new employee, and registering the related employee identifiers from each of the other systems.

Once the new employee identities are registered into the Datamart, the EmployeeQueue

each registered data source, passing in the Matrix Employee XML definition (Figure 7).

Data services

Each data source provides a domain-specific implementation of `Fill()`, `Validate()`, and `Update()` methods of the IMatrixDatasource interface: (Figure 8)

- ActiveDirectoryEmployee
- CeridianEmployee
- BSTEmployee
- MatrixEmployee

The `Fill()` method maps the properties owned by the subscribing system into the Matrix Business Object XML. For example, the ActiveDirectoryEmployee data source will fill the Email property, but will not fill the FirstName and LastName properties, which are owned by the CeridianEmployee data source.

Architecting the Enterprise Identify Synchronization system in this manner ensures details of the mapping, domain-specific languages and queries are isolated from the top-level workflow and synchronization process. The top-level services and workflow services therefore only need to focus on the implementing requirements associated with handling the Matrix Business Object XML Each DataSource uses the Employee ID native to its data domain from the Matrix Employee XML. A blank value for a given ID means that the employee does not exist in the system in question – in which case default values are returned.

If a system has knowledge regarding an employee's identity, the DataSource layer maps the properties it owns from the local system domain into the Matrix Employee XML definition. This process builds the overall view of the employee's identity from all systems registered in the Employee Service.

A successful pass through this process results in the synchronization of the Employee's identity to all subscribing systems.

Conclusion

The Enterprise Identity Synchronization architecture provides Matrix with an automated framework for keeping identity up to date across multiple business systems and domains.

Defining an XML-based Matrix Business Object for data used by the synchronization architecture means that the services themselves become independent of the data they transport. This service/message pattern is the basis for creating a full Matrix Services Oriented Architecture. The use of a Matrix Business Object ensures that properties used across all systems are put into terminology that makes sense to Matrix Business, and is separate from specific system domains.

Because the BSTFX engine itself leverages Windows Communication Foundation, the actual service endpoints are separate from the communications protocol they use — so it is possible to take advantage of having multiple entry points to each of the services (Figure 11). Separating the architecture into workflow and pure service layers that follow common patterns and contracts gives additional flexibility of being able to either call the workflow layer from an external client, or call the service layer directly.

The enablement approach has also proven to be a powerful and effective blend of methodology, software architecture, and best-practice implementation, providing enormous business value and key benefits to Matrix, including:

- The system was constructed from architecture to initial deployment in six weeks.
- Leveraging BST's expertise, while doing the actual production development onsite at Matrix, ensured that the system would directly meet Matrix requirements.
- Because the engagement covered the entire software cycle from requirements through development, Matrix resources were able to participate in architectural decisions and more importantly build a full understanding of the system.
- The base classes and contracts created for the Enterprise Identity Synchronization are generic enough to allow for extension of the framework not only to other systems but to other classes of data — such as projects, clients, and vendors.

Figure 10: Synchronization pattern

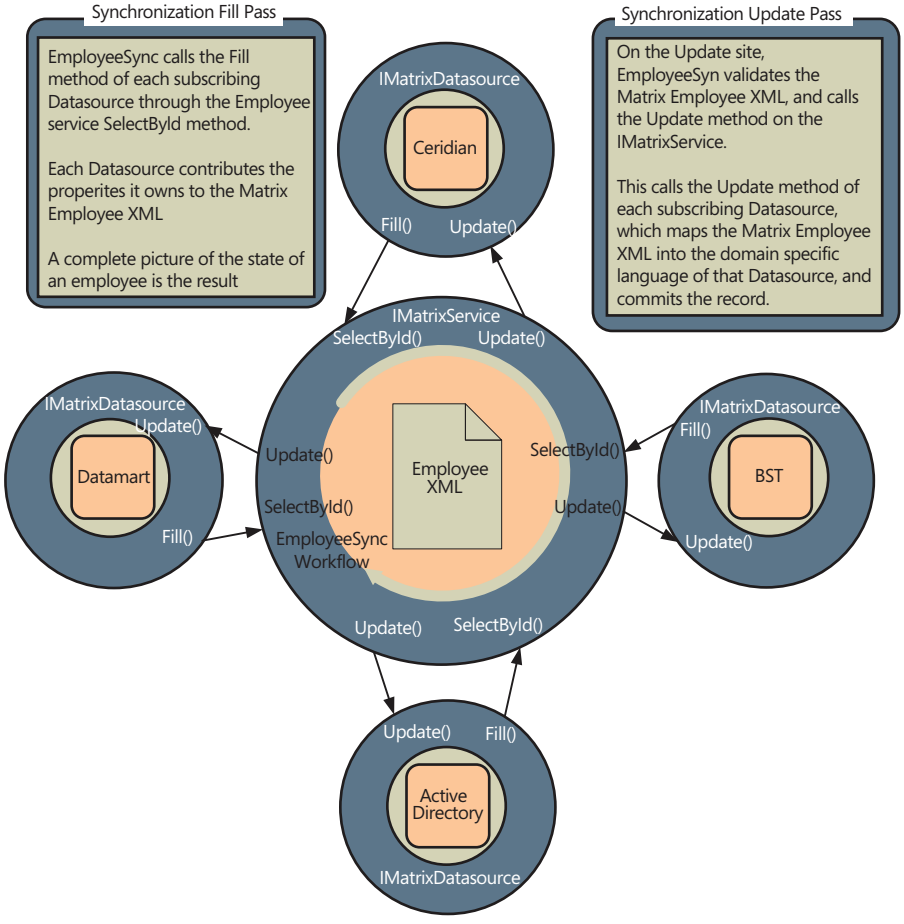
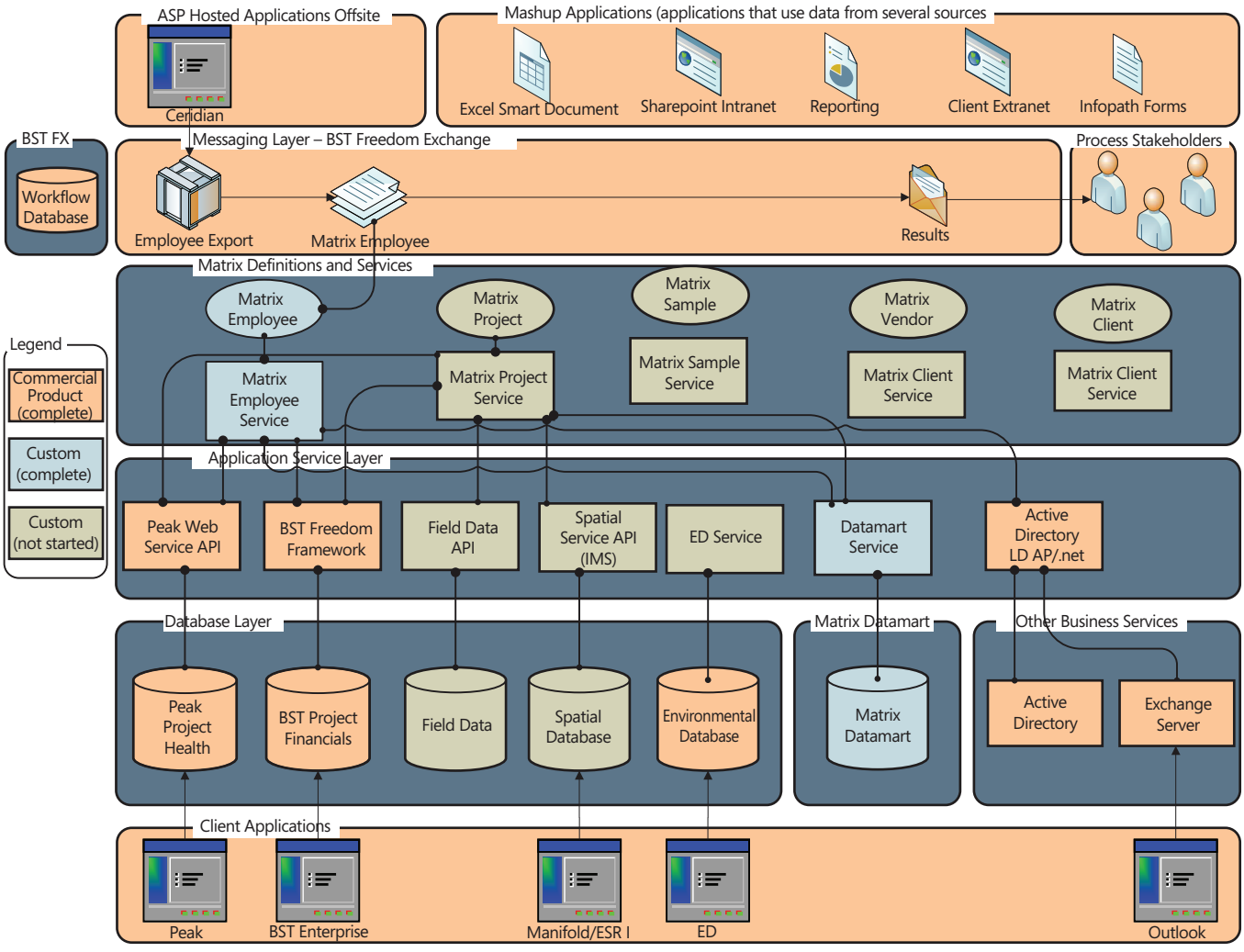


Figure 11: Future systems



- All materials, including source code, architectural drawings, and system design documents were turned over to Matrix — thus giving Matrix full access to all assets derived from the engagement and the ability to extend the system.

The flexibility of the architecture is already being leveraged by several new InfoPath systems that query the Employee service layer and use the XML definition to extract identity data for lookup lists in the forms.

About the Authors

Mike Morley is the Systems Architect at Matrix Solutions, a rapidly growing Environmental and Engineering Consulting firm based in Calgary, Alberta Canada. His responsibilities include designing and implementing corporate systems architecture, setting direction and vision for Matrix IT and IM based on Matrix business goals and system design, development, and deployment in support of and extending the business. Mike holds a Bachelor of Applied Sciences in Geological Engineering from the University of Waterloo. Mike has a diverse background in software development and design including developing three-dimensional blast design and drawing management applications for the mining industry, enterprise software development at Pandell Technology Corporation, and spatial/scientific software including 3D mine modeling systems at Golder Associates. Visit Mike's blog at http://www.menome.com/.

Barry Lawrence is the Director of Systems Integration Management for BST Global and is responsible for driving the vision and development of the BST integration platform. Barry's areas of expertise include Software Architecture, Quality Assurance, Project Management, and Software Development Methodology. Barry has over 15 of years of professional experience developing and leading technical teams in the implementation of enterprise-scale software systems, including seven years as an application development consultant for Software Architects (Sogeti, a subsidiary of Capgemini). Barry holds a Bachelor of Science in Marine Biology from the University of Tampa, and has certifications as a Microsoft Certified Solution Developer (MCSD) and Microsoft Certified Trainer (MCT).

ARC

Microsoft®

THE ARCHITECTURE JOURNAL™
Input for Better Outcomes