Claims-based Identity and FIM
A Whitepaper from Oxford Computer Group

This Whitepaper from Oxford Computer Group examines Claims-based Identity and FIM – are they friends or foes?

This paper considers two viable strategies for solving the "identity problem", and in asking whether they are somehow competitors, concludes that they are not merely co-operative, they are symbiotic.

Table of Contents

Claims-based Identity and FIM – Friends or Foes?

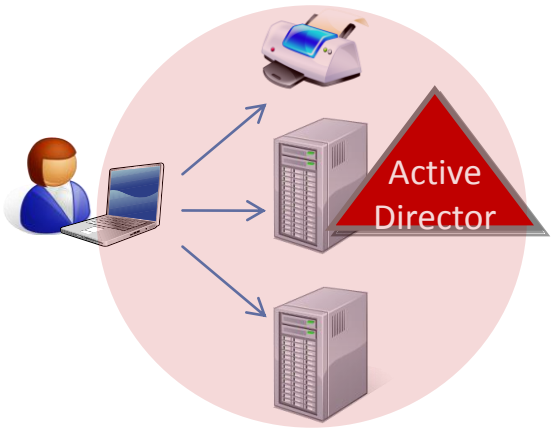
This paper considers two viable strategies for solving the "identity problem", and in asking whether they are somehow competitors, concludes that they are not merely co-operative, they are symbiotic.

What's the Problem?

First we'd better make sure we understand the problem we are trying to solve.

Simple Applications

When we sign on to Windows in a corporate network environment and start using straightforward Microsoft applications (accessing network shares, printers and so on) we expect a reasonably seamless experience in which we do not have to authenticate repeatedly, and in which authorization is taken care of through Active Directory groups and Access Control Lists. The small amount of personal data that is required (email address, initials etc.) might also be found in Active Directory, or might be looked after by the application concerned – but things normally roll along pretty smoothly. You would be annoyed if you had to log on separately to Excel, Word and Powerpoint. And it would be tedious if you had to log on each time you wanted to open or save a file, or print a document.



All part of the happy domain family: one log on, one set of permissions represented by an access token ("fistful of SIDs"),

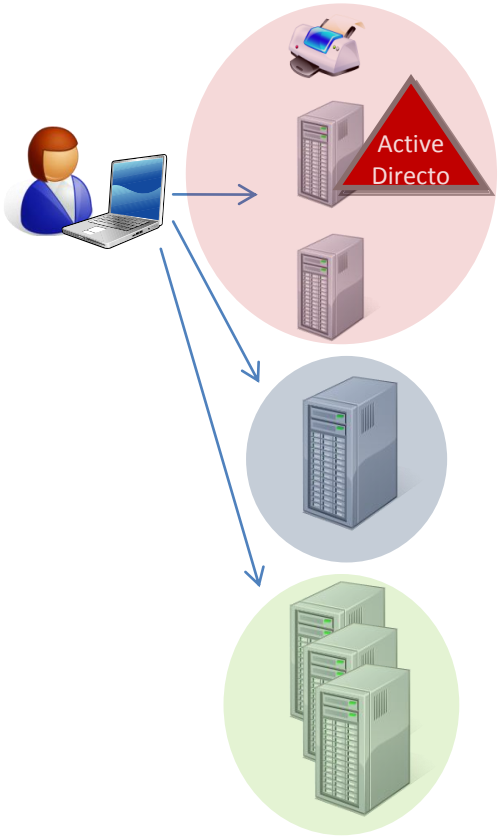
Complex Applications

As an organization rolls out more applications, like HR and ERP systems, the above typically no longer applies. First the application may require separate authentication – perhaps storing its own logon and password in its own database or directory. Second it may have its own authorization system, perhaps involving its own groups, or a more or less complex role hierarchy of some kind. Finally it may store the personal attributes that it needs, for example, email, telephone number, session information or variations on name (surname, display name etc.). All this stuff could perhaps be stored in Active Directory and retrieved as necessary, but for this to happen several things must be true:

- The organization must realize it is possible
- ... and be prepared to rely on a single technology and/or supplier
- ... and be prepared to pay the extra it will usually cost
- The relevant departments must be prepared to let it happen
- The software vendor or developer must make it possible
- There must be no significant technical hurdles

These things are rarely all true, a fact that is demonstrated by the fact that very few organizations have centralized in this way. The result is that:

- Organizations spend more time and money managing these systems than they need to
- Each new roll-out is harder to do than it needs to be
- Systems are not as secure as they could be
- Organizations find it hard to demonstrate compliance with legal



Islands of authentication, authorization and personal attributes; multiple logons, disconnected roles

and regulatory requirements about data
- Their users have a sub-optimal experience.

We could characterize the typical situation as "all silos", but few organizations have no centralization – most have several centres. AD forms one central point; then there may be SAP systems clustered around a Common User Access (CUA) data distribution point; there may be Lotus Notes apps; some non-AD LDAP directory providing services to XYZ clients; and maybe an Oracle "island". So there can be these "Islands of Identity", which in themselves make sense to the silo managers – but at the enterprise (or company) level it still looks like a bunch of silos.

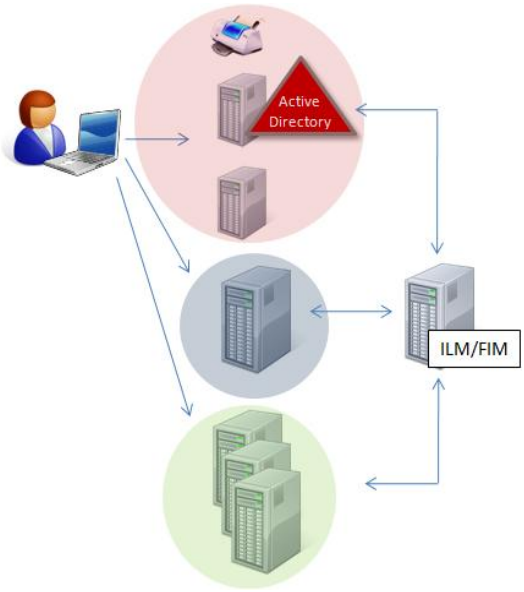
How Do We Fix the Problem?

Somehow we need to stop building and re-building custom plumbing and/or user account databases into every new application that comes along. But as we have seen, centralization hasn't typically happened, and even when it does happen, along comes a merger or acquisition to mess it up.

Identity Lifecycle Management

A popular, current solution is to implement some kind of Identity Lifecycle Management. This can:

- Synchronize existing identity objects and attributes
- Provision new objects based on an authoritative source of objects
- Deprovision unwanted objects
- Manage authorization information across systems (groups, roles etc.)
- Synchronize authentication information across systems (logons and passwords), leading to Reduced Sign-On (RSO)



Synchronization of logons, passwords, groups and roles, personal attributes (could combine with SSO solution to further improve user

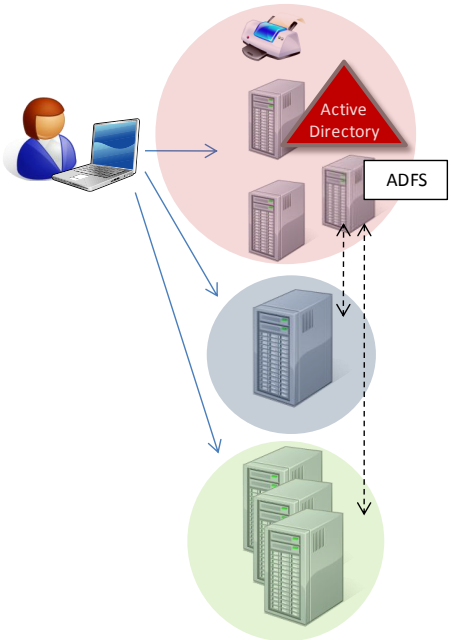
Microsoft's FIM 2010 can do all this pretty well (though authorization information generally requires some additional development work). If you add to this an SSO solution of some kind, then you have achieved Nirvana.

Except that it isn't really Nirvana – it's a bit of a kludge; it's a darn good kludge – but it's a kludge! And it does little to halt the proliferation of that custom plumbing, those user account databases, and those sign-ons.

Claims-Based Approach

But wait! Surely, we are told, a claims-based approach allows developers of applications to rely on other parties to authenticate users, to provide authorization information and perhaps even personal attributes – and to do so without the relying applications needing to know too much about the issuing services – as long as they are based on the same standards.

A user might log on with an Active Directory (AD) account as usual. When they want access to another (non-AD) application, either via a web browser or an intelligent client, that application could (having been suitable redeveloped) authenticate them based on a token provided by a suitable (trusted) provider (ADFS, for example). It could also accept claims from this (or another) provider which can be translated into permissions (via roles, attributes or whatever) – these claims perhaps



Applications redeveloped to accept authentication and claims from trusted providers (arrows only show conceptual connections and trust relationships)

being based on a database of enterprise role information. Personal attributes can also be represented as claims. The application does not now need to store authentication or authorization information, or even personal attributes locally in a database, and nor does it need to read them from a network directory service.

So provided that relying applications and issuing services can communicate effectively using standards, much of that custom plumbing and maybe even those user account databases could be taken out of the applications – or at least not further proliferated.

Federation, Claims, Trust, ADFS and WIF

When we talk of Federation, we are (at least in the simplest cases) referring to two or more security realms – usually different organizations – agreeing to trust each other to the extent that users from realm A can get access to resources (like data) in realm B more or less seamlessly. Rather than the user needing an account (and permissions etc.) in both realms, they might present a set of Claims provided by the (trusted) realm A – and are thus granted access in realm B. Whether these realms are within a single organization, or are simply two organizations, the benefit of only having one set of accounts and associated permission data is obvious enough.

A Claim is an assertion which is in doubt. Claims are usually derived from attributes, but the difference is that you have to trust that the claims made by the issuing party are true. For example, I can claim to you that my eyes are brown (brown eyes are an attribute of me), but if you can't see me, or a picture of me (and even that could be doctored), you have to treat that claim as in doubt. If I send you a token that confirms my eyes are brown, and that this has been validated by an authority you trust (or are prepared to trust for this transaction), then you can decide to accept that claim as being, provisionally at least, true. Trust, then, is vital to the claims paradigm – and in this case it means some of our "Islands of Identity" trusting others. That shouldn't be too difficult to arrange!

Microsoft's federation technology, which was codenamed "Geneva" during its incubation, has resolved into Active Directory Federation Services version 2 (ADFS v2) and Windows Identity Foundation (WIF). ADFS V2 is an update of ADFS, and is a server based security token service that issues and transforms claims and other tokens, manages user access, and enables federation and access management for simplified single sign-on. The Windows Identity Foundation is a set of tools to enable developers to develop applications so that they can interoperate based on industry standard protocols, allowing applications and identity infrastructure services to communicate via claims. So WIF can allow developers to redevelop applications to function in a claims-based manner, probably facilitated by ADFSv2.

So federation is a key solution when two organizations need to share data. It can also apply to cooperating realms within an organization (our "Islands of Identity"). We can argue about whether this is still "federation" – importantly, it is a claims-based approach. Any company that has, or plans to have in the future, more than one web application or web service, can benefit by starting to use a claims-based model for identity (because they can cut down on user account databases and the associated permissions data – all eth custom plumbing that is otherwise needed) – and this is especially true if they have plans to federate with other organizations at some point (and federation includes putting their applications out in the Cloud).

Switching to a claims-based paradigm in an organization of significant complexity is not going to be simple. It will take time, effort, co-ordination, will – and a critical mass of vendors making their wares compliant with the appropriate technologies (WIF, in our example). It follows that a big question for us is whether Identity Lifecycle Management is some sort of dead end to be eventually ripped out and replaced, or whether it can it somehow co-exist with a claims-aware paradigm, or even support it. Let's see!

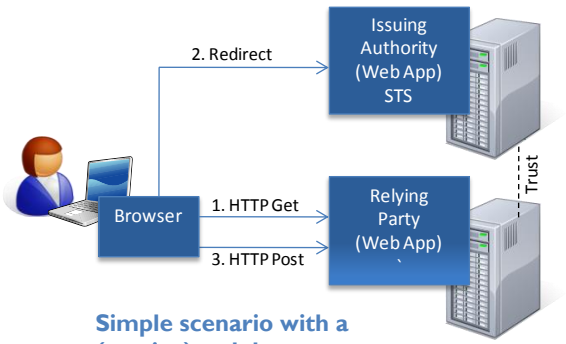
Some more terminology

Identity? Attribute? Claim?

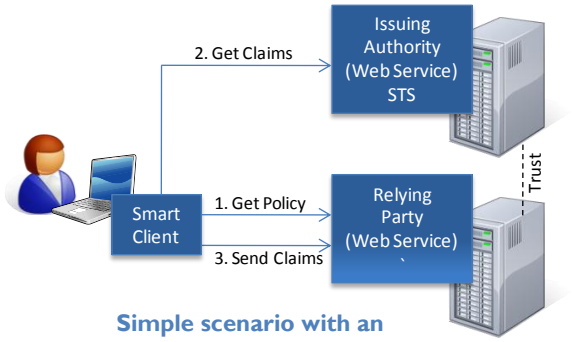
From the point of view of an Identity Lifecycle Management system, an Identity is an Object with a set of Attributes; from a Claims point of view an Identity is an Object with a set of Claims. Idetity Management concerns itself with more or less atomic attributes – the majority of which are simple strings: email address, phone number, display name and so on and so forth. Claims could be simple strings – for example a unique identifier like email address or userPrincipalName – but a claim could be an assertion like "I am over 18", or it could be something a little richer in content, like a list of roles.

Issuing Authorities, Relying Parties and Security Tokens

An Issuing Authority authenticates users, and issues security tokens that can contain serialized lists of claims. If a Relying party trusts the issuer, it can use those claims to decide how to handle requests from users. If the token is digitally signed it can also have confidence that the claims are genuine.



Simple scenario with a (passive) web browser



Simple scenario with an active client and web

In these two examples, the client finds out what claims are needed, goes to an issuing authority and – having been authenticated – gets a token and presents it to the relying party along with its request. In a web service, these claims are carried in the security header of the SOAP envelope. In a browser-based web application, the claims arrive via an HTTP POST from the user's browser, and may later be cached in a cookie if a session is desired.

How to get there from here

Where is "here"? Well as already mentioned, we could characterize the typical situation as "all silos" – but in fact it is more usual to find "Islands of Identity".

The evolution in the organization typically takes the form of increasing clumping of systems into islands, and a non-silo project which implements co-operation between the islands. We are talking about the founding of the "Federated Islands of Identity" – which, as with all such projects in the political world, involves cooperation, pooling of powers and responsibilities, a level of trust, and an over-arching body which coordinates the activities among the otherwise autonomous states.

Now suppose you are about to implement your very first claims-aware application, in an organization that is more or less "siloed". Where will it get its claims? By using methods provided by WIF it can to go to an ADFS server and get some claims. The fact that a valid security token is returned means the user has been authenticated. WIF can also return claims based on AD attributes and group memberships. Cool!

But wait a minute! Are we sure that AD has all the information we need to define fine-grained access control in this new system (and every other system that comes along)? And what about all the personal attributes we need to store – do we want them clogging up our corporate AD?

Let's rewind a little. Suppose instead that we are implementing the same claims-aware application, but in an environment that already has a mature Identity Lifecycle Management implementation. The role information

that you can use to control fine-grained access is already being managed, and is available – all we need to do is make sure we can get at it. If that information is in a SQL Server database, then to query this database for data to turn into claims is a trivial matter when using WIF – and it just happens that FIM could provide just such a database (or rather a view into a database). Also, personal attributes are already being managed, so the chances are we can get these from somewhere too. At the very least, we know where everything is, and we can decide whether to use the existing sources, or whether in this case application might as well be store some locally needed information itself.

Let's try to summarize the story so far:

	Authentication	Authorization	Personal Attributes
1) Starting point – more or less in silos	Several different ones	Little standardization	Each system stores and manages what it needs
2) Centralized	One authentication – SSO built in	One set of authorization data (e.g. role definitions)	One set of personal attributes
3) Mature Identity Lifecycle Management	RSO (logins and password synchronized) – or an additional SSO solution	Role information defined in one place, and mapped and propagated to others to provide authorization	Attributes synchronized
4) Claims-based Paradigm	User must still be authenticated by multiple systems, but the user might be largely unaware (there is no need for multiple accounts)	Any number of issuing parties, but claims-aware client goes and gets what each relying party needs, without the user being aware	Again, any number of issuing parties – but note that many relying parties may as well keep data that only they care about (which implies an account)

Of course, once in position 4, we are also well placed to federate (and remember, federation might include cloud-based providers). Any new apps can make use of existing providers, and it is also possible to gradually combine issuing parties behind the scenes and simplify matters.

1 to 2?

As discussed earlier, it has generally turned out to be almost impossible to get to 2 at all. If you have managed this you can stop reading (and you probably stopped earlier anyway!)

1 to 3?

Yes – we have done this literally hundreds of times.

1 to 4?

Getting direct from 1 to 4 in one go is not easy either – and it may not be possible at all until a lot of systems become WIF-capable (in other words that vendors make changes to bought-in solutions, and/or you redevelop your home-grown systems).

1-3-4

Quite simply, we know how to get from 1 to 3, and we can gradually get from 3 to 4. If you plan in future to go to a claims-aware model, it makes sense to start now using an Identity Lifecycle Management approach to identity data – including roles. This gives you a solution right now, and you might even facilitate a better solution later, including the possibility of federation with other organizations. In fact, even in the long term, it will probably never make sense to remove the Identity Lifecycle Management piece (though its job may become simplified).

If you implement FIM 2010 now, including role management, you get attribute synchronization (and provisioning and deprovisioning), RSO, and access control co-ordinated across the enterprise. When a claims-

aware application comes along, FIM 2010 is the obvious system to use as *the* enterprise authorization claims issuer (WIF can get authorization claims from easily-configured SQL Server queries). Thus your first claims-aware application can be provided with its authentication, authorization and personal attribute needs.

Conclusion

Not only is it easier to implement claims-aware applications once an Identity Lifecycle Management system (like FIM 2010) has been implemented – it is hard to see that there is any other sensible route. If you haven't sorted out attributes and roles in the enterprise, you really can't expect to implement claims-aware apps effectively (and anyway, you can't expect a potential federator to trust you!)

For Further Information

For more information about how Oxford Computer Group can help you evaluate, upgrade, implement or enhance your identity management system, please contact us as follows:

Oxford Computer Group (OCG) is a Microsoft Gold Partner specializing in Identity and Access (IAM) management consulting and training. With operations in the USA, UK and mainland Europe, OCG has an enviable repository of expertise, solution components and training courses. OCG has deployed over 500 IAM solutions and trained over 5000 people on Microsoft IAM technologies. We understand IAM – benefit from our expertise and capability.

Email: info@oxfordcomputergroup.com
Web: www.oxfordcomputergroup.com

UK: Bignell Park Barns, Chesterton, Oxfordshire, OX26 1TD, UK
Tel: +44 (0)8456 584425 Fax: +44 (0)8456 584426

US: One Microsoft Way, Building 25, Room 1492, Redmond WA 98052, USA
Tel: + 1 877 862 1617

DE: Winterlestr.10b D-85435 Erding, Deutschland
Tel: +49 8122 892089-0 Fax: +49 8122 892089-99

BeNeLux: Prins Willemstraat 21, 2584 HS, Den Haag, The Netherlands
Tel: +31 (0)70 36 21 045 Fax: +31 (0)70 36 21 677

