2008

Microsoft

Kim Cameron

[IDENTITY SOFTWARE + SERVICES ROADMAP]

This paper is based on a presentation made to the Microsoft Professional Developers Conference (PDC) in Los Angeles in early November 2008. It speaks primarily to a developer and architect audience, but I'm sharing it more widely in the hope that it might shed light on how Microsoft sees identity, providing insight into the products and services we have been building to deliver on the industry-wide vision of an interoperable Identity Metasystem. This paper is written as a narrative that mirrors how I delivered the talk on stage.

Contents

The First Lines of Every Application

Whether you are creating serious Internet banking systems, hip new social applications, multi-party games or business applications for enterprise and government, you need to know something about the person using your application. We call this identity.

There is no other way to shape your app to the needs of your users, or to know how to hook people up to the resources they own or can share.

I want to be clear. Knowing "something" *doesn't* mean knowing "everything". We want to keep from spreading our own and our customers' personal information all over the Internet. It means knowing only what's needed to provide the experience you are trying to create.

This *might* sometimes include someone's name. Or knowing they are really the person with a certain bank account.

Or it might just involve knowing someone is in a certain role. Or in a certain age bracket.

It also might simply be a question of remembering that some user prefers Britney Spears to New Kids on the Block or Eminem.

Identity Turbulence

But getting identity information into the app is one of the messiest aspects of application development. People are up to their necks in passwords. They use many different mechanisms to establish identity, and it is getting worse.

One of my friends at Microsoft is in charge of the identity aspects of one of our flagship apps. Let's call him Joe. His experience with identity reflects what many other developers have told me.

He started years ago by building in support for usernames and passwords. He was supposed to finish up within a few weeks – but ended up in a second project to build in "better" password reset and account management.

He thought this would be the end of it, but when large enterprise customers saw the application, they wouldn't deploy it unless he added a "help desk" component - so he did…

Active Directory started gaining critical mass. He had to set up a new project to integrate with it. No sooner was this finished than large sites tried to use the app with multiple "forests", and getting this right took even longer than the initial AD support.

Next, people wanted to use his app "outside the firewall", so he needed to add One Time Password (OTP) support. But some customers preferred Smart Cards. They were adamant that if OTP was supported, Smart Cards should work too!

Joe was all set to return to his "core work" when he was asked to add support for SAML. He hadn't even finished his investigation of what this might imply when customers started requesting OpenID support too. Just days later, he was asked to integrate the app with another software product from a different vendor - and it used a completely different and proprietary approach to identity.

I won't even discuss the phishing attacks…. They brought about a whole new round of security reviews - one for every one of the projects just discussed.

And today, Joe needs to make sure his application is easily "hostable" in the cloud as well as on-premise. Plus it should support delegation so it can access other services on behalf of his users…

You'll understand that Joe really wants and needs a better way. *He wants to focus on the core value of his app, not ever-changing identity requirements.*

Claims-based Access

As we understood more about these problems, and the hardships they were causing developers, we started work on a way to insulate the application from all these issues.

Our goal? You, the developer, would write the application once and be done, yet it would support all the identity requirements customers have as they host it in different environments and use it in changing scenarios.

This was the same kind of problem we had in the early days of computing. Back then, you needed to write separate code for each type of disk drive you wanted to support. There was no end to it. If you didn't support the new drives you'd lose part of your market. So we needed the idea of a logical disk that was always the same -- a single model for handling all disk access.

We can now do the same thing around identity. We use what we call the Claims Based Access model.

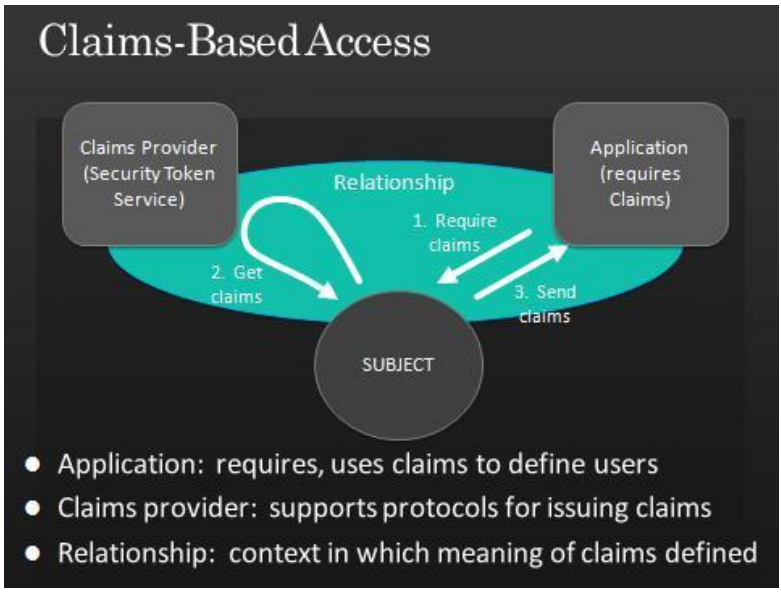
A claim is a statement by one subject about another subject. Examples: someone's email address, age, employer, roles, customer number, permission to access something. *There are no constraints on the semantics of a claim.*

The model starts from the needs of the application: you write your application on the assumption you can get whatever claims you need.

Then there is a standards-based architecture for getting you those claims. We call it the Identity Metasystem – meaning a system of identity systems. This is a shared architecture with broad support across the industry.

Here's how the architecture works. As I said, the application is in control. It specifies the kinds of claims it requires. The claims providers support protocols for issuing claims. You can also pop in claims providers that translate from one claim to another – we call this claims transformers. That makes the system very flexible and open.

The technical name for a claims provider is a "Security Token Service". You'll see the abbreviation STS used in many of my illustrations.



The important thing here is that all existing identity mechanisms can be represented through claims and participate in the Identity Metasystem. As an app developer, you just deal with claims. But you get support for all permutations and combinations without getting your hands dirty or even thinking about it.

I say "all" to emphasize something – the open nature of this system. It accepts and produces identities from and for every type of platform and technology – There are no walled gardens.

You get to choose how to participate in the Identity Metasystem: You can choose to get your identity from anywhere you wish. You can choose any framework to build your app. You can choose to use any client or browser. In all parts of the architecture, you can choose a Microsoft component or someone else's…or build your own.

What's Involved for the Developer?

Let me give you an example. I'll peek ahead and show you how the claims-based model is used in the "Geneva" Framework - new capabilities within .NET. Other frameworks would have similar

capabilities, though we think our approach is especially programmer-friendly.

Basically, to answer the "Who are you?" question, you write your app as you normally do, and simply add this extra configuration to your **app.config** file:

```
<federatedAuthentication enabled="true">
   <wsFederation
      issuer="https://sts1.contoso.com/FederationPassive/"
      realm = "http://web1.contoso.com/MyApp"
      passiveRedirectEnabled = "true"/>
</federatedAuthentication>
```

(There are a few more cut-and-paste lines needed, to make sure some modules are included, but otherwise that's it.)

Now, when a user hits your app, if they haven't been authenticated yet, they will get automatically redirected to the claims provider at **https://sts1.contoso.com/FederationPassive** to pick up their claims. The claims provider will get your user to authenticate, and if all goes well, will redirect her back to your application with her identity set, and any necessary claims available to your program. In other words, with zero effort on your part, no unauthenticated user will ever hit your app. Yet you are completely free to point your app at any claims provider on any platform made by any vendor and located anywhere on the Internet.

To drill into the actual claim values, you use the

```
IClaimsIdentity caller = Thread.CurrentPrincipal.Identity
                         as IClaimsIdentity;
string Role = (from c in caller.Claims
               where c.ClaimType == MyClaimTypes.Role
               select c.Value).Single
```

technique shown in this code snippet. You'll see the Thread has a Current Principal, and the Principal has an Identity, so you get a Claims Identity interface as shown, then cycle through the claims or pull out the one you need. In this case, it is the claim with the type of "role" - in the the enum MyClaimTypes.Role…

If you are an application developer, we've already come to the big takeaway of this presentation. You can get up and go home now. Everything else I'm going to show you is just to give you a deeper understanding of all the many use cases and scenarios that can be supported through these mechanisms.

Again, the claims shown in this example are implemented through well accepted industry standards. The same code works with claims that come from anywhere, any platform, any vendor, any operating system, any cloud provider.

Solving Problems with Claims

I don't want to overwhelm you with a shopping list of all the scenarios in which the Claims-based architecture solves problems that used to be insurmountable. Suffice it to say that claims can be used in a very wide range of scenarios from intra-enterprise to federation to consumer Web, and both on-premises and in the cloud.

But I'll start from the enterprise point of view, and look at how this system helps with the big new trend of federation between partners. Then we'll look at cloud computing, and see that the same architecture dramatically simplifies developing applications that can take advantage of it. Finally, we'll see how the approach applies to consumer-oriented web applications.

Enterprise Federation

The rigid enterprise perimeter is dissolving as a result of the need for digital relationships between an enterprise and its suppliers and customers, as well as the outsourcing of functions and services, the use of temporary workers, and having employees who sometimes work from home. The firewall is still a useful element in a concentric set of defenses, but must at the same time now be permeable.

Most of us are even learning to collaborate on a per-project basis with partners who in other contexts might be our competitors. So the relationships between business entities must be defined with more and more granularity.

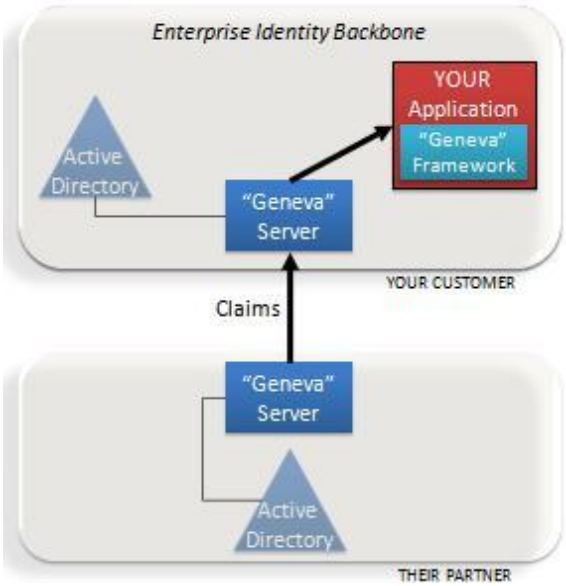
In looking at this, I'm going to start with a very simple scenario - a story of two companies, where one has built an app in-house or has installed an ISV app for their own employees, and now wants to extend access to employees from a partner.

In the past, even this simple requirement has been really hard and expensive to fulfill. How can Microsoft help you solve this problem using the claims model?

Microsoft Code Name "Geneva"

Well, I'm happy to announce today, the first beta of "Geneva" software for building the claims-aware applications I've been talking about. It has three parts:

1. The "Geneva" Framework: A framework you use in your .Net application for handling claims. This was formerly called "Zermatt" Framework.



2. "Geneva" Server: A claims provider and transformer (STS) integrated with Active Directory. It comes with Windows, and makes managing trusts and policies easy. Importantly, it supports Information Cards, making it easier for people to understand what identities they are using where, and to *avoid phishing of their enterprise credentials*. You may in the past heard this server being referred to as AD FS "2". We re-named it because it does so much more than just federation now.
3. Windows CardSpace "Geneva": The second generation Information Card client for federation that is dramatically faster and smaller than the first version of CardSpace, and incorporates the feedback and ideas that have emerged from our customers and collaborators. It is also integrated with "Geneva" Server to enable managed cards.

In the use case we've been considering, our solution works this way: each enterprise puts up a single Geneva Server – harnessing the power of their Active Directory.

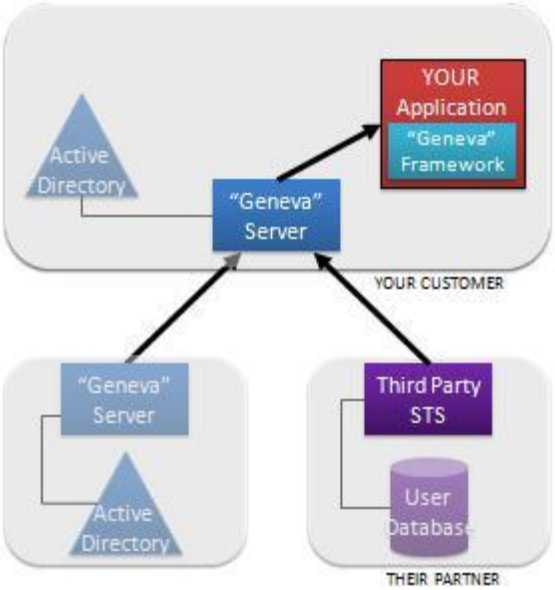
Then the administrators of the application alter the .NET configuration to point to their enterprise's "Geneva" Server (with the configuration change I demonstrated here). At this point, your customer's application has become part of what we call an enterprise identity backbone, and can accept claims.

So the software framework and components provide a single identity model that users configure in any way they want. If you have written to this model, your app now works for both "employees" and "partner users" without a code change. All that is required is to set up the "Geneva" STS's .

The Fatal Flaw

Anyone who has been around the block a few times knows there is one fatal flaw in the solution I've just described: Your customer may have partners who don't use Active Directory or don't use "Geneva" or have settled on a non-Microsoft product.

No problem. All aspects of "Geneva" are based on



standards accepted across the industry – WS-Trust and WS-Federation.

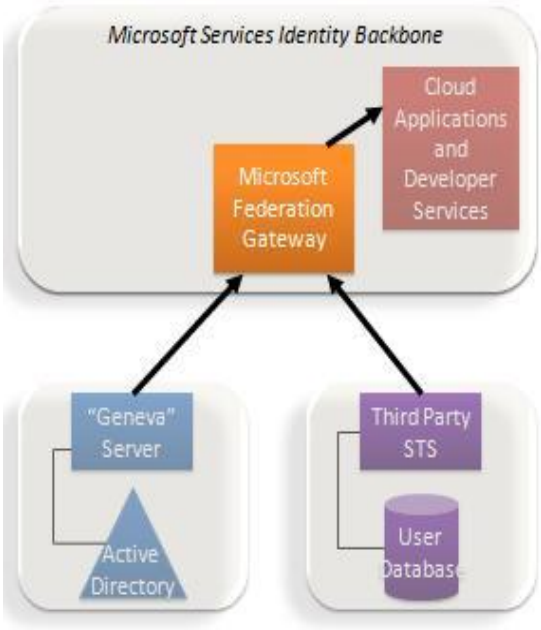
I'm also very happy to announce that "Geneva" supports the SAML 2.0 protocol. Basically, *no system that supports federation should be out of reach.*

All this means your partners aren't forced to use "Geneva" if they want to get access to your applications. They can use any third party STS and that is part of the great power of the solution.

Does Microsoft practice what it preaches?

Microsoft is an enterprise too. So if this architecture is supposed to be good for our enterprise customers, what about for Microsoft itself? Are we following our own advice?

I'm here today to tell you Microsoft has fully stepped up to the plate around federation. And it is already providing a lot of benefits and solving



problems.

You've heard a lot at the PDC about Azure. Microsoft offers cloud applications like hosted SharePoint and Exchange, and cloud developer services like the .Net Services and SQL Data Services, as well as a whole range of applications. We want other enterprises to be able to access these services and sites, much like other enterprises want their own customers and partners to access the systems pertaining to their businesses.

So we make our offerings available to customers via the Microsoft Federation Gateway (MFG), which anchors our "services identity backbone", and is based on the same industry standards and architecture delivered through "Geneva" Server. It is all part of one wave, the "Geneva" wave of identity software + services.

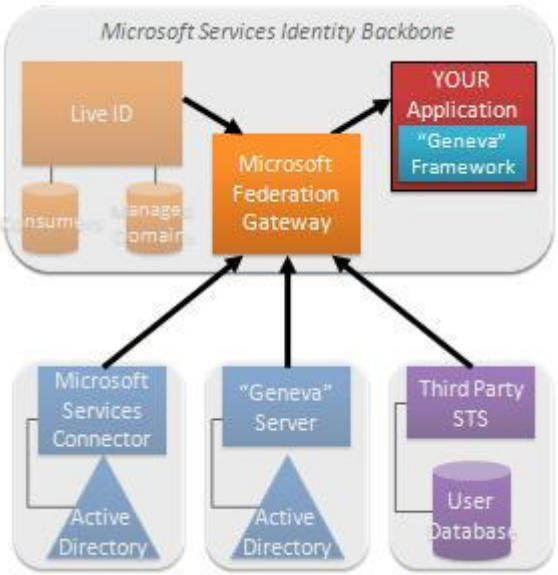
The result is pretty stunning, in terms of simplifying our own lives and allowing us to move forward very quickly - as it will be for enterprises that follow the same route. Through a single trust relationship to our gateway, our customers can get access to our full range of services.

Again, we're not telling our customers what federation software to use. They can federate with the MFG using "Geneva" or other third party servers that support standard protocols. And they can use the same protocols to federate with other gateways run by other organizations.

Live ID Becomes Interoperable

Microsoft also operates one of the largest claims providers in the world - our cloud identity provider service called Windows Live ID.

It plays host to more than four hundred million consumer identities.



In the "Geneva" wave, Live ID will add "managed domain" services for sites and customers wanting to outsource their identity management. With this option, Live would take care of identity operations but the sign in/sign up UX can be customized to fit the look of your site.

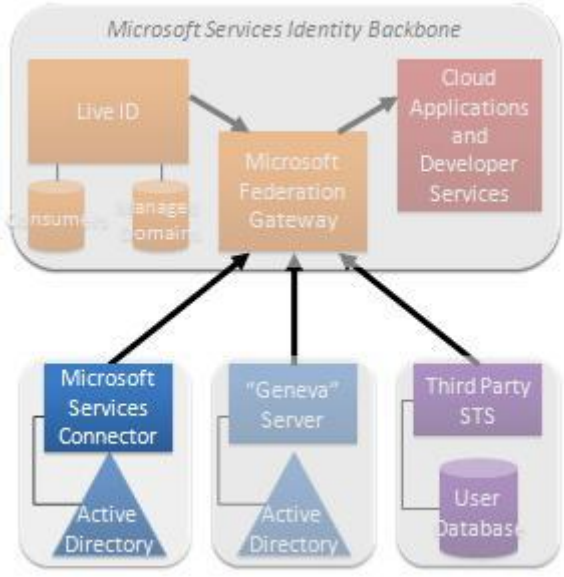
But in what I think is an especially exciting evolution, Live IDs also get access to our cloud applications and developer services via the gateway, and are now part of the *same open, standards-based architecture that underlies the rest of the "Geneva" wave.*

Microsoft Services Connector

Some customers may want to take advantage of Microsoft's cloud applications, hosting, and developer services - and have Active Directory - but not be ready to start federating with others.

We want to make it very easy for people to use our cloud applications and developer services without having to make any architectural decisions. So for that audience, we have built a fixed function server to federate Active Directory directly to the Microsoft Federation Gateway.

This server is called the Microsoft Services Connector (MSC). It was built on "Geneva"



technology.

Since it's optimized for accessing Microsoft cloud applications it manages a single trust relationship with the Federation Gateway. Thus most of the configuration is fully automated. We think the Microsoft Services Connector will allow many enterprises to start working with federation in order to get access to our cloud, and that once they see the benefits, they'll want to *upgrade their functionality to embrace full federation through "Geneva" Server and multilateral federation.*

Through the combination of "Geneva" Framework and Server, Microsoft Services Connector, Live ID, the Microsoft Federation Gateway - and the ability to use CardSpace to protect credentials on the Internet - millions of Live and AD users will have easy, secure, SSO access to our cloud applications and developer services.

What About YOUR Applications?

OK. This is all very nice for Microsoft's apps, but how do other application developers benefit?

Well, since the Federation Gateway uses standard protocols and follows the claims-based model, if you write your application using a framework like "Geneva", you can just plug it into the architecture and benefit from secure, SSO access by vast numbers of users - ALL the same users we do. The options open to us are open to you.

This underlines my conviction that Microsoft has really stepped up to the plate in terms of federation. We haven't simply made it easier for you to federate with Microsoft in order to consume Microsoft's services. We are also trying to make you as successful as we can in this amazing new era of identity. The walled garden is down. We want to move forward with developers in a world unconstrained by zero sum thinking.

Configure your application to accept claims from the Microsoft Federation Gateway (MFG) and you can receive claims from Live ID and any of the enterprise and government Federation Gateway partners who want to subscribe to your service. Or ignore the MFG and connect directly to other enterprises and other gateways that might emerge. Or connect to all of us.

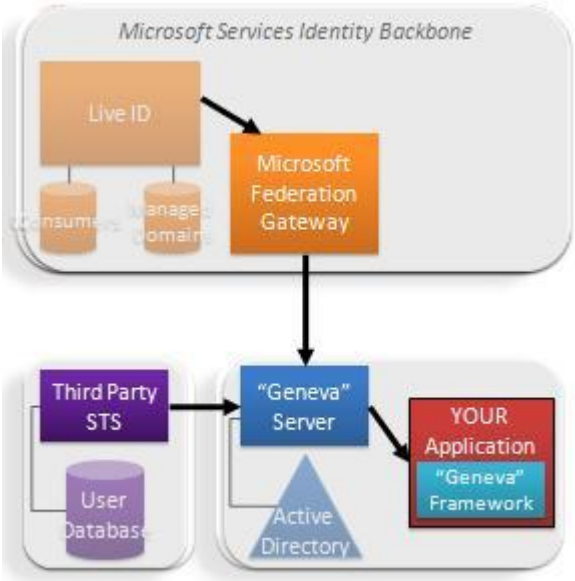
Crossing Organizational Boundaries

If this approach sounds too good to be true, some of you may wonder whether, to benefit from Microsoft's identity infrastructure, you need to jump onto our cloud and be trapped there even if you don't like it!

But the claims-based model moves completely beyond any kind of identity lock-in. You can run your application wherever you want - on your customer's premise, in some other hosting environment, even in your garage. You just configure it to point to the Microsoft Federation Gateway or any other STS as a source of claims. It's your choice.

These benefits are a great demonstration of how well the claims model spans organizational boundaries. We really do move into a "write once and run anywhere" paradigm.

Configure Any Scenario



For even more flexibility, you can use an enterprise-installed "Geneva" Server as your application's claim source, and configure that server to accept claims from a number of gateways and direct partners.

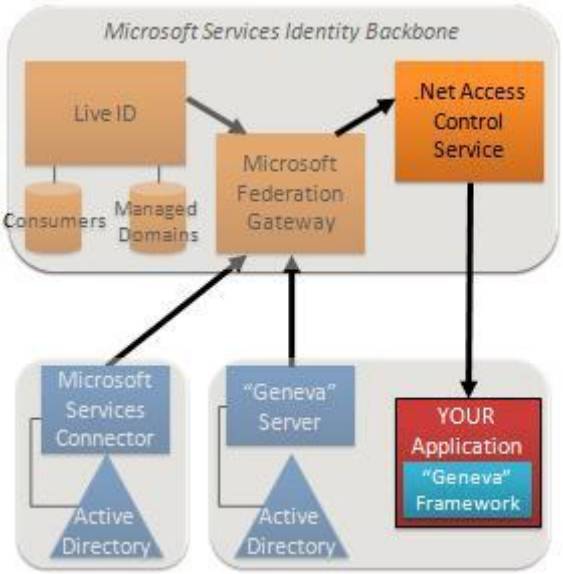
In the configuration shown here, the Geneva Server can accept claims both hundreds of millions of Live ID users and from a partner who federates directly.

Claims-based access really does mean applications are written once, hosted anywhere. Identity source is a choice, not a limitation.

You get the ability to move in and out of the cloud at any time and for any reason.

Even more combinations are possible and are just a function of application configuration. It's a case of "Where do you want to get claims today?". And the answer is that you are in control.

.NET Access Control Service



We have another announcement that really drives home the flexibility of claims.

Today we are announcing a Community Technical Preview (CTP) of the .Net Access Control Service, an STS that issues claims for access control. I think this is especially cool work since it moves clearly into the next generation of claims, going way beyond authentication. In fact it is a **claims transformer** STS, where one kind of claim is turned into another.

An application that uses “Geneva” can use ACS to externalize access control logic, and manage access control rules at the access control service. You just configure it to employ ACS as a claims provider, and configure ACS to generate authorization claims derived from the claims that are presented to it.

The application can federate directly to ACS to do this, or it can federate with a “Geneva” Server which is federated with ACS.

ACS federates with the Microsoft Federation Gateway, so it can also be used with any customer who is already federated with the Gateway.

The .Net Access Control Service was built using the “Geneva” Framework. Besides being useful as a service within Azure, it is a great example of the kind of service any other application developer could create using the “Geneva” Framework.

You might wonder – is there a version of ACS I can run on-premises? Not today, but these capabilities will be delivered in the future through “Geneva”.

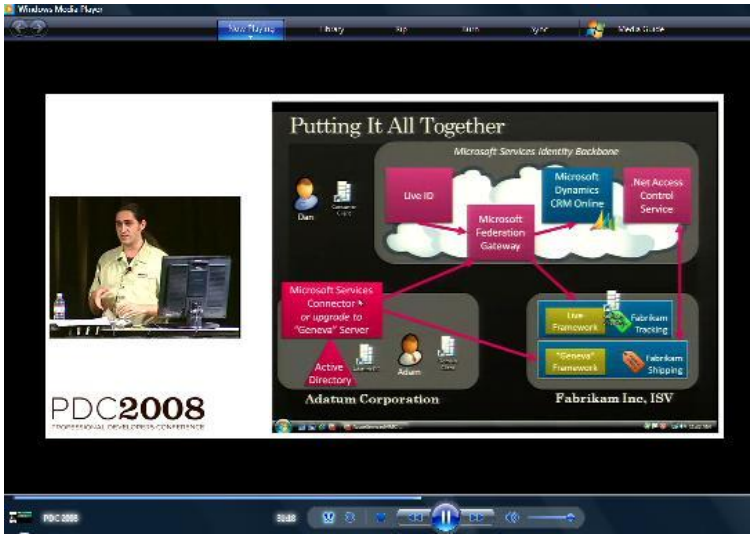
Putting It All Together

Let me summarize our discussion so far, and then conjure up Vittorio Bertocci, who will present a demo of many of these components working together.

- The claims-based model is a unified model for identity that puts users firmly in control of their identities.
- The model consists of a few basic building blocks can be put together to handle virtually any identity scenario.
- Best of all, the whole approach is based on standards and works across platforms and vendors.

Let’s return to why this is useful, and to my friend Joe. Developers no longer have to spend resources trying to handle all the demands their customers will make of them with respect to identity in the face of evolving technology. They no longer have to worry about where things are running. They will get colossal reach involving both hundreds of millions of consumers and corporate customers, and have complete control over what they want to use and what they don’t.

Click on this link[1] - then skip ahead about 31 Minutes - and my friend Vittorio will take you on a whirlwind tour showing all the flexibility you get by giving up complexity and programming to a simple, unified identity model putting control in the hands of its users. Vitorrio will also be blogging [2]in depth about the demo over the next little while. [If your media player doesn't accept WMV but understands MP4, try this link[3].]



[1] http://channel9.msdn.com/pdc2008/BB11/
[2] http://blogs.msdn.com/vbertocci
[3] ipod:%20http://mschnlnine.vo.llnwd.net/d1/pdc08/MP4/BB11.mp4

Live IDs Become OpenIDs

I've made a number of announcements today that I think will have broad industry-wide support not only because they are cool, but because they indelibly mark Microsoft's practical and profound commitment to an interoperable Identity Metasystem that reaches across devices, platforms, vendors, applications, and administrative boundaries.

I'm very happy, in this context, to announce that from now on, **all Live ID's will also work as OpenIDs**.

That means the users of 400 million Live ID accounts will be able to log in to a large number of sites across the internet without a further proliferation of passwords – an important step forward for binging reduced password fatigue to the long tail of small sites engaged in social networking, blogging and consumer services.

As the beta progresses, CardSpace will be integrated into the same offering (there is already a separate CardSpace beta for Live ID).

Again, we are stressing choice of protocol and framework.

Beyond this support for a super lightweight open standard, we have a framework specifically tailored for those who want a very lightweight way to integrate tightly with a wider range of Live capabilities.

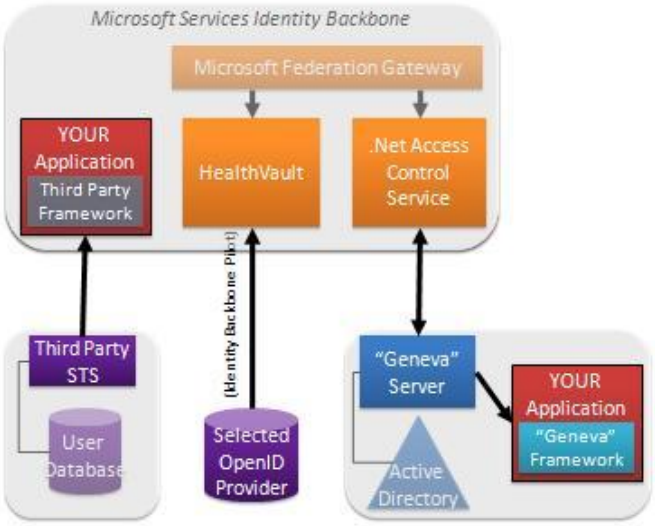
The Live Framework gives you access to an efficient, lightweight protocol that we use to optimize exchanges within the Live cloud.

It too integrates with our Gateway. Developers can download sample code (available in seven languages), insert it directly into their application, and get access to all the identities that use the gateway including Live IDs and federated business users connecting via "Geneva", the Microsoft Services Connector, and third party Apps.

Flexible and Granular Trust Policy

Decisions about access control and personalization need to be made by the people responsible for resources and information – including personal information. That includes deciding who to trust - and for what.

At Microsoft, our Live Services all use and trust the Microsoft Federation Gateway, and this is helpful in



terms of establishing common management, quality control, and a security bar that all services must meet.

But the claims-based model also fully supports the flexible and granular trust policies needed in very specialized contexts. We already see some examples of this within our own backbone.

For example, we've been careful to make sure you can use Azure to build a cloud application – and yet get claims directly from a third party STS using a different third party's identity framework, or directly from OpenID providers. Developers who take this approach never come into contact with our backbone.

Our Windows Azure Access Control Service provides another interesting example. It is, in fact, a security component that can be used to provide claims about authorization decisions. Someone who wants to use the service might want their application, or its STS, to consume ACS directly,

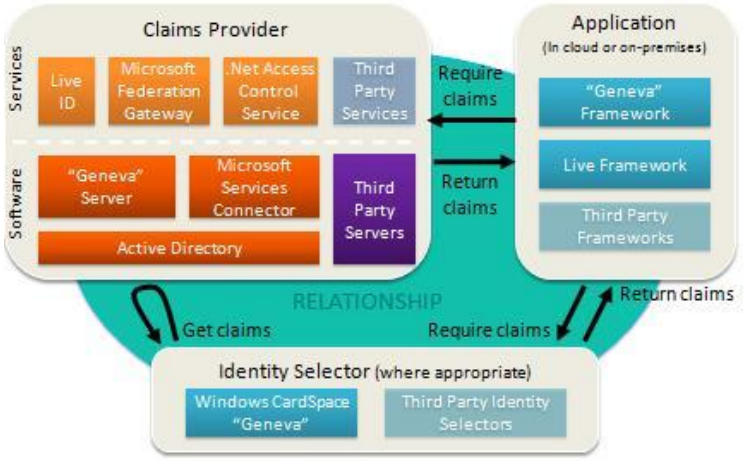
and not get involved with the rest of our backbone. We understand that. Trust starts with the application and we respect that.

Still another interesting case is HealthVault. HealthVault decided from day one to accept OpenIDs from a set of OpenID providers who operate the kind of robust claims provider needed by a service handling sensitive information. Their requirement has given us concrete experience, and let us learn about what it means in practice to accept claims via OpenID. We think of it as pilot, really, from which we can decide how to evolve the rest of our backbone.

So in general we see our Identity Backbone and our federation gateway as a great simplifying and synergizing factor for our Cloud services. But we always put the needs of trustworthy computing first and foremost, and are able to be flexible because we have a single identity model that is immune to deployment details.

Identity Software + Services

To transition to the services world, the identity platform must consist of both software components and services components. We believe Microsoft is well positioned to help developers in this area.

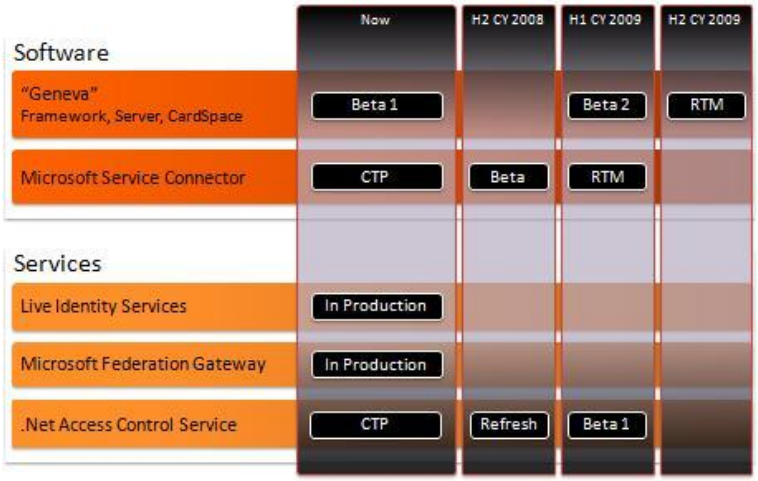


Above all, to benefit from the claims-based model, *none of these components is mandatory*. You select what is appropriate.

The needs of the application drive everything. The application specifies the claims required, and the Metasystem needs to be flexible enough to supply them.

Roadmap

Our roadmap looks like this:



Identity @ PDC

You can learn more about every component I mentioned today by drilling into the 7 other presentations presented at PDC (watch the videos…):

Presentations on Software

(BB42) "Geneva" Server and Framework Overview[4]

(BB43) "Geneva" Deep Dive[5]

(BB44) Windows CardSpace "Geneva" Under the Hood[6]

[4] http://channel9.msdn.com/pdc2008/BB42/
[5] http://channel9.msdn.com/pdc2008/BB43/
[6] http://channel9.msdn.com/pdc2008/BB44/

Conclusion

I once went to a hypnotist to help me give up smoking. Unfortunately, his cure wasn't very immediate. I was able to stop – but it was a decade after my session.

Regardless, he had one trick I quite liked. I'm going to try it out on you to see if I can help focus your take-aways from this session. Here goes:

"I'm going to stop speaking, and you are going to forget about all the permutations and combinations of technology I took you through today.

"You will remember how to use the claims based model. You'll remember that we've announced a bunch of very cool components and services. And above all, you will remember just how easy it now is to write applications that benefit from identity, through a single model that handles every identity use case, is based on standards for open interoperability and choice, and puts users in control."

Presentations on Services

(BB22) Live Identity Services Drilldown[7]

(BB29) Connecting Active Directory to Microsoft Services[8]

(BB28) .NET Services: Access Control Service Drilldown[9]

(BB55) .NET Services: Access Control In the Cloud Services[10]

[7] http://channel9.msdn.com/pdc2008/BB22/
[8] http://channel9.msdn.com/pdc2008/BB29/
[9] http://channel9.msdn.com/pdc2008/BB28/
[10] http://channel9.msdn.com/pdc2008/BB55/