Developers and Identity Services

- Tackling Identity Data with Identity Hub

An Oracle White Paper
Sept 2008

Developers and Identity Services
- Tackling Identity Data with Identity Hub

Developers and Identity Services
- Tackling Identity Data with Identity Hub

EXECUTIVE OVERVIEW

Service-Oriented Security (SOS) aligns with the overall Application-Centric approach of the entire Oracle Fusion Middeware platform - with the goal of providing a comprehensive, standards-based, developer-friendly platform. By leveraging and sharing many of the common Identity Services, SOS allows developers to spend the effort on where it counts the most – the application logic itself. However, to truly move away from the traditional silo-based approach in dealing with identity management, application developers building identity-enabled applications must look to the benefits of these services and understand when and how to use these services in their application design.

"However, to truly move away from the traditional silo-based approach in dealing with identity management, application developers building identity-enabled applications must look to the benefits of [Identity Services] and understand when and how to use these services in their application design."

This is the first in a series of "Developers and Identity Services" whitepapers, each intended to focus on a specific area of Identity Services from a developer standpoint. It builds on the concepts of Identity Services and Identity Externalization defined previously in the paper entitled - *Service-Oriented Security – An Application–Centric Look at Identity Management* - available at:

http://www.oracle.com/technology/products/id_mgmt/pdf/serv_oriented_sec.pdf

In this paper, we will take a closer look at the issues surrounding the current patterns for accessing identity data and what an ideal Identity Hub in SOS should provide for developers.

INTRODUCTION

Understanding each type of identity service and the use cases surrounding it is as crucial as the implementation itself. Of all the identity related tasks, the ability to access identity data is perhaps the most fundamental part of any identity-enabled application from a developer standpoint. Yet it is no trivial task. The many dimensions and the nature of identity data pose many challenges for developers. In general, the way identity data is stored, distributed and used is deployment-specific. Developers must try to take these variations into considerations without imposing unnecessary application-specific constraints on the customer environment. Identity data is also sensitive. Security becomes an issue both in the application's ability to access the identity sources and in the application's ability to handle and manage the identity data. The sensitivity of identity data has also brought along a list of compliance and privacy regulations that companies must now comply with.

"Of all the identity related tasks, the ability to access identity data is perhaps the most fundamental part of any identity-enabled application from a developer standpoint."

DEVELOPING AROUND IDENTITY DATA – A DAUNTING CHALLENGE

How an application consumes identity data is one of the most challenging aspects of developing an identity-enabled application. Identity data is different from regular application data in that there are many policy, privacy, security and deployment considerations that must be accounted for.

Identity Stores

Traditional applications have often relied on the application's own storage as the main identity data provider at runtime. With sources of identity data generally being distributed, synchronization becomes a key factor in this approach to keep the local store up-to-date. Externalizing the identity data to a centralized identity repository that can be shared across applications only simplifies the problem by reducing synchronization need for each individual application. The need for synchronization still persists at the centralized store. In reality, not everything can be centralized, and developers will often need to engage with multiple identity repositories. For example, a human resource management system (HRMS) should not be synchronizing an employee's salary or social security number into any external source even though the information may be needed by other applications outside of the HRMS.

"In reality, not everything can be centralized, and developers will often need to engage with multiple identity repositories."

"Authoritativeness" is often the reason for this separation of application data versus centralized data. If a particular application is seen as the sole source for a particular identity attribute, then the application should be considered as "authoritative" for that attribute. In this way, an HRMS is authoritative over employee salary or social security number. In the end, in order to create the complete image of a single identity needed for applications to function, developers are required to go through some convoluted process to correlate the identity record from one store to the other as different attributes that make up an identity are distributed across multiple identity stores.

Identity Protocols

Identity repositories in the form of an RDBMS or LDAP have played a central role in providing identity data. With different types of repositories come the different types of protocol. Current developers must equip themselves to handle all aspects of the different types of protocol from storing the different types of connect information (such as database connect string or LDAP URL) to understanding how to use the various protocols themselves such as JDBC and JNDI to retrieve information. With the advent of federated and web-based protocols such as SAML, WS-*, SOAP, etc, the nature and usages of these protocols are only becoming more and more diverse and complex.

Identity Schema

With the right set of protocols accessing the backend data, the developer must now navigate through the different kinds of identity schema to request and extract the data. A user's email may be retrieved from a particular column of a particular table of a corporate RDBMS. In an enterprise where a centralized LDAP is used for identity data, the developer can use a JNDI query will retrieve the "mail" attribute from the user's LDAP Distinguished Name. LDAP provides well known "objectclasses" which define a set of standardized LDAP schema for common objects such as users, groups and organizations. However, there are cases where a custom LDAP schema is used and client applications may have to react accordingly. In the case where SAML is used, a set of user profile attributes including email may be made available through the Attribute Assertion within the SAML assertion of the user. Again, the actual attribute carrying the email value and how the value is represented may vary depending on the types of SAML profile being used which would depend on how the handshake between the SAML identity provider and the SAML service provider is configured. The importance of understanding the identity schema becomes even more crucial if the application needs to write back information to update any identity information.

"Ultimately, how the data is stored and how it is represented varies depending on the customer deployment environment – even with the benefit of strong interoperable standards."

Looking closely, the issues identified in the area of repositories, protocols and schemas are mostly related to deployment time concerns. Ultimately, how the data is stored and how it is represented varies depending on the customer deployment environment – even with the benefit of strong interoperable standards. And when faced with the various options in identity repositories, protocols and identity schemas, a developer is often forced to cut corners for simpler, more manageable solutions – perhaps by limiting to a single private store that is accessible using a single API through a single protocol where the identity data is represented in a well known, maybe even proprietary schema. With that, we have gone back a full circle to the application-specific silo approach that we are so desperately trying to avoid.

Identity Data Security

Access Control

Once the developer solves the problem of accessing the identity data, the next hurdle is to control the access to the data. There will always be some form of native security at the repository level – security that is implemented within the storage mechanism itself to control access of the data. However, in order to define

“However, in order to define native access policies, the native store must recognize the entity that is accessing it - be it a user or an application, and sometimes a combination of both.”

native access policies, the native store must recognize the entity that is accessing it - be it a user or an application, and sometimes a combination of both. Using LDAP as an example, in order to define access control for an application through LDAP access control, the application must be a recognizable entity in the form of a Distinguished Name that can bind and query against the LDAP. In other words, a footprint of the application must exist. The same applies to RDBMS where a database user is defined before access control can be defined natively within the database.

With applications coming and going, this creates a hassle and potential security holes for the identity store administrators who typically do not welcome these types of application specific security requirements. In some cases, where security is needed based on the application and user context together, protocol such as LDAP simply is not equipped to handle it. For example, a payroll application and an HR application will have different access to the login user data in terms of which user attributes are allowed to be fetched by the different applications. Without the necessary backend support, this becomes an extremely difficult problem to solve from a developer standpoint.

Delegation

“The difficulty in defining the right level of access is due largely to the lack of ability to define the right context.”

Delegation is another tricky use case encountered by developers. For example, a manager may perform certain tasks on behalf of a direct report who is on vacation. Precisely capturing the right delegation and access control required for these tasks is often difficult. From a read-only perspective, the access to identity data granted to the manager should be the same or at least sufficient for him to carry out the operation on behalf of his direct report – but no more. Furthermore, any operations or changes to identity data or records done should be recorded as that of the manager’s. The access must be carefully controlled and audited. Once again, to implement these access controls on the application side is difficult for developers. Many of the criteria driving these access controls and policies are likely residing in the identity layer – such as manager/direct-report relationships, etc.

The difficulty in defining the right level of access is due largely to the lack of ability to define the right context. Today’s access control can handle a one-dimensional context like the access for a particular user or an application. However, more complex security requirements are forcing access control to handle multi-dimensional context – to understand the full context that involves a combination of user, application, target, protocol, purpose, etc in order to assess the right level of access.

Identity Privacy and Protection

With new regulations and compliance requirements, data protection goes beyond just protecting the data source. Developers are now faced with many emerging requirements dealing with how the application itself uses and manages the identity data.

Identity Data Life Cycle

Regulations such as the European Union Directive on Data Protection mandate that information to be disposed and/or archived when no longer needed or after some period of inactivity. From a developer standpoint, it may affect how long certain information can be retained within the application.

Minimal Information Approach

"With new regulations and compliance requirements, data protection goes beyond just protecting the data source."

Consider the use cases where an application needs to know whether the end user is an adult or not. One option is to fetch the birth date of the user and based on current date, the developer can write code to figure out whether or not this person is an adult. This violates the *Principal of Least Knowledge* as knowing the birth date of a person is much more sensitive than knowing whether he or she is an adult. Another example is to verify the last 4 digits of someone's social security number. Again, the idea behind it all (even from the end user perspective) is that the verifier (be it an actual person or a software application) need not know the complete social security number to verify the end user's identity. For a developer, the easiest way to implement this would be to fetch the entire social security number and check that it ends with the correct 4 digits. Again, this exposes more information to the application than needed.

The minimal information approach is to address these types of scenarios to minimize the amount of information needed by a developer and an application to obtain what is necessary. It is sufficient to simply tell the application that a user is an adult or not (an example of a *Derived Attribute* or *Claim*), or have the provider accept the last 4 digits of the social security number and return a true or false (an example of a *Verification Service*).

Solving these types of privacy problems today proves to be extremely difficult as the identity providers themselves, which in most cases are the native identity stores, do not have the capabilities necessary to support these use cases.

IDENTITY HUB – THE MISSING LINKS

An Identity Hub serves as the broker between the application and the various authoritative sources of identity attributes in both enterprise and federated scenarios – providing identity data to the application. In looking at these use cases above, we have identified some key requirements in order for an Identity Hub to be successful.

"A well-defined Identity Hub should provide the necessary abstraction for the developer to be agnostic to the actual deployment environment."

Deployment Abstraction

A well-defined Identity Hub should provide the necessary abstraction for the developer to be agnostic to the actual deployment environment.

- Backend agnostic - The integration and handshake with one or more identity stores are handled within the Identity Hub itself and shielded away from the developer. Furthermore, the Identity Hub should be able to negotiate across the multiple identity stores that exist in the environment and provide a single cohesive view of the identity. This includes the ability to aggregate identities whose parts are distributed across multiple stores, and accumulating different identities that reside in different repositories.
- Protocol agnostic - The developer now deals with a single virtual Identity Hub through a single programming interface – removing the need for a developer to juggle multiple protocols and improving extensibility and customization with respect to the customer deployment environment.
- Schema agnostic – The developer now deals with a single schema supported by the Identity Hub. With this added Identity Hub layer, it should be able to define different schema profiles for different applications and industry verticals, tailored to their specific needs. Appropriate mappings can then be defined at the attribute level, making life much simpler for a developer.

Security Features

The added layer in the Identity Hub should also accommodate additional security features to better protect the identity data.

- Additional Access Control – The Identity Hub can provide additional filtering and access control that is otherwise not suitable or not possible to implement in the native identity store. The key requirement is to align security with the context of the party requesting the information. This is needed to support the application-user-context and other delegation use cases.

"The Identity Hub can provide additional filtering and access control that is otherwise not suitable or not possible to implement in the native identity store."

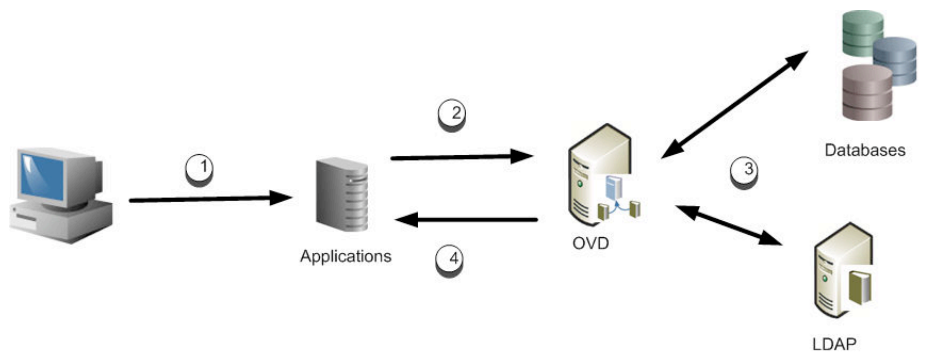
- Identity Life Cycle – Any life cycle requirement on identity data can be defined within the Identity Hub. Appropriate treatment of local identity data can now be enforced against a centralized set of policies defined by the Identity Hub.
- Minimal Information – The Identity Hub can extend its schema to provide more complex attribute mapping to support these use cases. For example, Boolean responses (predicates) can be constructed for the is-Adult use case without returning the birth date of the end user. In addition, is-Adult may mean >18 years of age in some countries and >21 in others. These policies now reside in the Identity Hub rather than in the application logic itself – allowing for easier customization, uniform enforcement and compliance.

The Identity Hub allows the application developers to push some of the complexity in dealing with identity data down the stack – away from the application logic itself. Furthermore, it provides the ability for an application to declare its requirements up front - leaving it up to the Identity Hub to satisfy these requirements. This frees up the developer to concentrate on the application logic by decoupling the work for the provider. But most importantly, the Identity Hub provides the vehicle to handle much more complex identity data related requirements where customizations, configurations and extensions are easier to achieve.

IMPLEMENTING AN IDENTITY HUB WITH ORACLE VIRTUAL DIRECTORY

"A simple way to implement an identity hub for LDAP-enabled applications is through Oracle Virtual Directory.**"**

The ideal identity hub should provide a single authoritative view of user data in what is generally a decentralized environment where user data is scattered among multiple repositories including various directories and databases. A simple way to implement an identity hub for LDAP-enabled applications is through **Oracle Virtual Directory** (OVD). OVD's virtualization technology provides the ability to unify multiple directories and to allow LDAP access to databases or other proprietary identity data stores. OVD supports adapters connecting to majority of the directory servers including Oracle Internet Directory, Microsoft Active Directory, Novell eDirectory, IBM Tivoli Directory Service and Sun Java System Directory Server. It also supports adapters connecting to relational databases including databases from Oracle, IBM and Microsoft.



1 Client connects to an application (e.g. Portal)

2 Application accesses OVD to locate, authenticate, and authorize the user as if the OVD was a standard LDAP Server

3 OVD transforms the request into one or more native requests to the authoritative identity sources (e.g. via LDAP, SQL, or Web Services)

4 OVD normalizes responses from the native identity sources and transmits the results in a way that the application can use to grant or deny access to the client or create a personalized response

Figure 1. Typical Deployment with OVD

With OVD, the effort and the logistics to unify the identity data are decoupled from the developer's hands. The developer for the application can now interact through an LDAPv3 interface (for example, using JNDI) against this virtual identity hub and obtain a real-time consolidated view of a person's identity record.

DECLARING IDENTITY REQUIREMENTS WITH IDENTITY GOVERNANCE FRAMEWORK

Identity Governance Framework (IGF) provides many key components to the Identity Hub solution by defining the appropriate open standards for the "Application-Identity Hub" handshake. From the application end, IGF's **Client Attribute Requirements Markup Language (CARML)** provides a declarative way for designers and developers to communicate their "identity requirements" without building this logic into the application itself. An important goal with CARML is to support expression of privacy constraints for "identity data" (in the form of WS-Policy). This provides the language to define the security requirements that can be tailored towards the application specific requirements.

"The Attribute Services API **(aka** Aris ID API**) enables externalized identity services because it allows the developer not to have to make decisions that are handled best by deployment managers and identity infrastructure managers."**

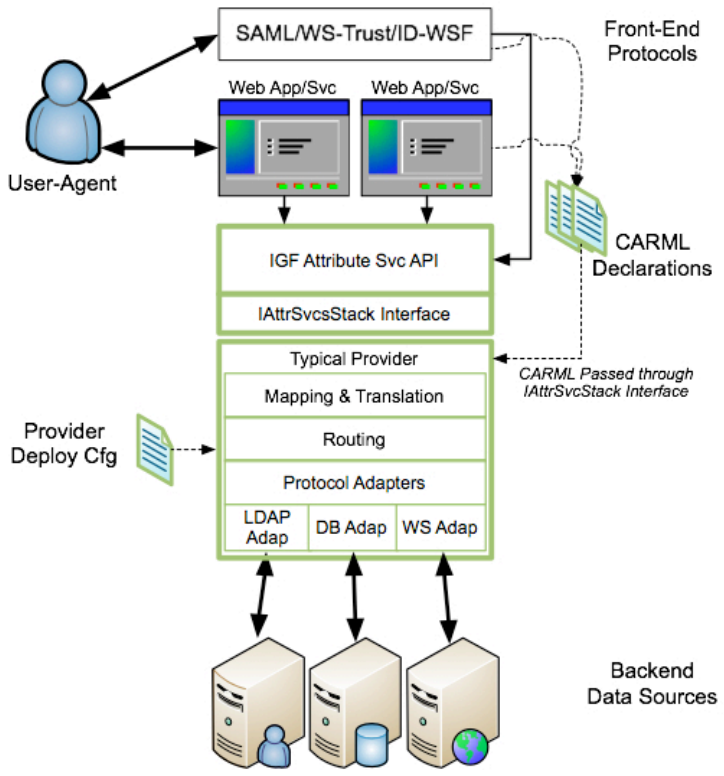


Figure 2. High-level IGF Architecture

With the OVD approach, a solution is available for today's LDAP enabled applications. But OVD alone does not provide a completely protocol-agnostic solution. Oracle has been leading an open source effort at www.openLiberty.org known as the Aris Project whose goal is to develop an API that allows developers to use externalized identity data. The **Attribute Services API** (aka **Aris ID API**) enables externalized identity services because it allows the developer not to have to make decisions that are handled best by deployment managers and identity infrastructure managers.

The API implements IGF (Identity Governance Framework) standards and enables decoupled development where developers have the capability to fully test the use of identity services within the IDE environment without having to be encumbered by having to deploy complex infrastructure at development time. Furthermore, when applications are developed with the Aris API, the applications will be fully ready to use externalized data whether in a federated environment using SAML, WS-*, or in internal enterprise services using directories or databases.

Aris is also the first developer API to implement IGF's CARML and Privacy Constraints standards. With security defined in the declaration requirements of CARML, the decoupling allows Attribute Services API to provide a simpler interface and usage model. Developer no longer has to account for differences in implementations – avoiding complex protocol handling code and configuration requirements (such as JNDI). Furthermore, with CARML in the picture, additional access control and privacy constraints can now be defined in a backend-agnostic manner to further protect identity data based on application-specific requirements.

"With CARML in the picture, additional access control and privacy constraints can now be defined in a backend-agnostic manner to further protect identity data based on application-specific requirements."

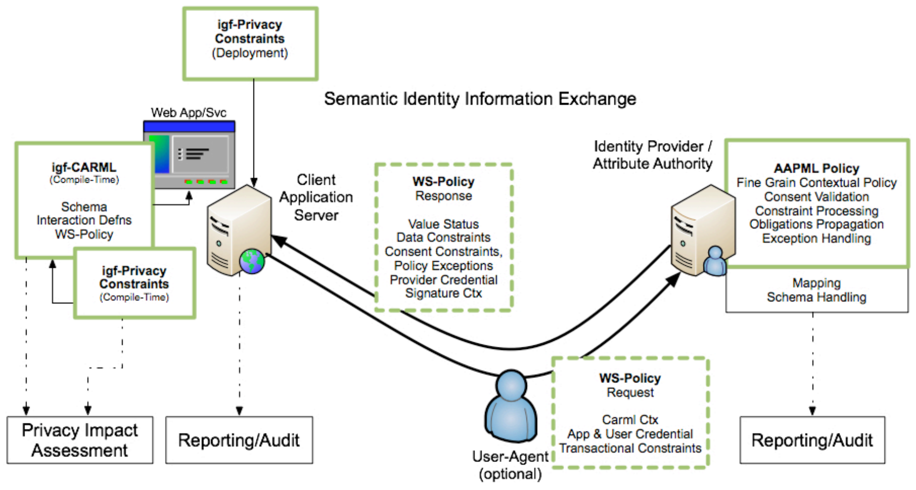


Figure 3. Data Exchange in IGF

Applications enabled with the Aris API will have the opportunity to select from a number of Attribute Service "provider" technologies based on the open source standard interfaces. For example, Oracle customers can deploy using either embedded Oracle Virtual Directory technology, or using other implementations by Oracle competitors or open source communities. (Note: At this time, Oracle is working with the Higgins community to adapt Higgins IdAS for this purpose).

For more information on Identity Governance Framework, please visit:

http://www.projectliberty.org/liberty/resource_center/specifications/igf_1_0_specs

CONCLUSION

Identifying the source of identity data is an important part of any identity-enabled application design. With the appropriate Identity Hub, a developer is decoupled from many of the hassles encountered in traditional integration between applications and backend identity stores. Because the issues surrounding identity data providers are mostly deployment time issues, the decoupling puts Identity Hub in the right position to address these issues that are otherwise out of the hands of the developers.

A new paradigm emerges from the Identity Hub by allowing applications to declaratively state the requirements to the Identity Hub – leaving it up to the provider to satisfy the requirements. In doing so, the Identity Hub is able to provide features in the area of schema mapping, access control and privacy support that are difficult if not impossible to achieve at the application level with traditional methodologies. The end result is applications developed with a more efficient, elegant and secure way to handle and operate with identity data.

For more information on Oracle's security technology,

*Go to **http://www.oracle.com/security***

ORACLE

Developers and Identity Services – Tacking Identity Data with Identity Hub
September 2008
Author: Stephen Lee
Contributing Author: Phil Hunt, Nishant Kaushik

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com