



# **Id Governance - Identity Privacy and Access Policy Marketing Requirements Document Use Cases**

**Version:** 1.0

## **Abstract:**

This document provides the use cases and requirements for Id-Governance Identity Privacy and Access Policy provided by the participants of the Liberty Alliance Business and Marketing Requirements Expert Group.

**Filename:** liberty-use-cases-for-Id Governance-v1.0.pdf

**Editors**

Phil Hunt, Oracle Corporation

**Contributors**

Shin Adachi, NTT

Conor Cahill, Intel

Marco Casassa Mont, Hewlett-Packard Labs

Philippe Clement, France Telecom/Orange

Hidehito Gomi, NEC Corporation

Makoto Hatakeyama, NEC Corporation

Lena Kannappan, Fugen Solutions

Kurt Kolok, Liberty Staff

Rob Lockhart, Liberty Staff

Paul Madsen, NTT

Prateek Mishra, Oracle Corporation

Andy Rappaport, Computer Associates

Colin Wallis, New Zealand

This Market Requirements Document (MRD) has been developed by the Business and Marketing Expert Group of Liberty Alliance to capture the business requirements for an identity governance framework. Liberty Alliance is making this MRD publicly available to the industry at large for review and consideration. In addition, this MRD is being provided to the appropriate technical standards development group within Liberty Alliance for consideration of new technical work to address the requirements identified herein. This publication does not constitute a commitment by Liberty Alliance, explicit or implied, to develop technical specifications in full compliance with the requirements herein, now or in the future.

**Notice:**

This document has been prepared by Sponsors of the Liberty Alliance. Permission is hereby granted to use the document solely for the purpose of implementing the Specification. No rights are granted to prepare derivative works of this Specification. Entities seeking permission to reproduce portions of this document for other uses must contact the Liberty Alliance to determine whether an appropriate license for such use is available.

Implementation of certain elements of this document may require licenses under third party intellectual property rights, including without limitation, patent rights. The Sponsors of and any other contributors to the Specification are not and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights. **This Specification is provided "AS IS," and no participant in the Liberty Alliance makes any warranty of any kind, express or implied, including any implied warranties of merchantability, non-infringement of third party intellectual property rights, and fitness for a particular purpose.** Implementers of this Specification are advised to review the Liberty Alliance Project's website (<http://www.projectliberty.org/>) for information concerning any Necessary Claims Disclosure Notices that have been received by the Liberty Alliance Management Board.

Copyright © 2007 2FA Technology; Adobe Systems; Agencia Catalana De Certificacio; America Online, Inc.; American Express Company; Amsoft Systems Pvt Ltd.; Avatier Corporation; BIPAC; BMC Software, Inc.; Bank of America Corporation; Beta Systems Software AG; British Telecommunications plc; Computer Associates International, Inc.; Credentica; DataPower Technology, Inc.; Deutsche Telekom AG, T-Com; Diamelle Technologies, Inc.; Diversinet Corp.; Drummond Group Inc.; Enosis Group LLC; Entrust, Inc.; Epok, Inc.; Ericsson; Falkin Systems LLC; Fidelity Investments; Forum Systems, Inc.; France Télécom; French Government Agence pour le développement de l'administration électronique (ADAE); Fugen Solutions, Inc; Fulvens Ltd.; GSA Office of Governmentwide Policy; Gamefederation; Gemalto; General Motors; GeoFederation; Giesecke & Devrient GmbH; Hewlett-Packard Company; Hochhauser & Co., LLC; IBM Corporation; Intel Corporation; Intuit Inc.; Kantega; Kayak Interactive; Livo Technologies; Luminance Consulting Services; Mark Wahl; MasterCard International; MedCommons Inc.; Mobile Telephone Networks (Pty) Ltd; NEC Corporation; NTT DoCoMo, Inc.; Netegrity, Inc.; Neustar, Inc.; New Zealand Government State Services Commission; Nippon Telegraph and Telephone Corporation; Nokia Corporation; Novell, Inc.; OpenNetwork; Oracle Corporation; Ping Identity Corporation; RSA Security Inc.; Reactivity Inc.; Royal Mail Group plc; SAP AG; Senforce; Sharp Laboratories of America; Sigaba; SmartTrust; Sony Corporation; Sun Microsystems, Inc.; Supremacy Financial Corporation; Symlabs, Inc.; Telecom Italia S.p.A.; Telefónica Móviles, S.A.; Telenor R&D; Thales e-Security; Trusted Network Technologies; UNINETT AS; UTI; VeriSign, Inc.; Vodafone Group Plc.; Wave Systems Corp. All rights reserved.

**Contents**

<b>1</b>	<b>Introduction .....</b>	<b>6</b>
1.1	Revision History.....	6
1.2	Existing Materials .....	6
1.3	Introduction to the Id-Governance Initiative.....	6
1.4	Use Cases: Privacy-Enabled Exchange of Identity Data .....	7
1.4.1	Simple Attribute Exchange .....	7
1.4.2	Identity-Related Data Exchange with Policy .....	8
1.4.3	Declarative Applications.....	9
1.4.4	Federated Exchange .....	10
1.4.5	Flexible Applications .....	12
1.5	Terminology .....	12
1.6	Scope .....	16
<b>2</b>	<b>Use Cases.....</b>	<b>17</b>
2.1	Developer Declarative Requirements.....	17
2.1.1	Using a Client Attributes Requirements-aware API.....	18
2.1.2	Code-Scan Generated Client Attributes Requirements.....	19
2.1.3	Hand Generated Client Attributes Requirements .....	20
2.2	Application Deployment Manager .....	21
2.2.1	Client Attributes Requirements-aware Client API Enabled .....	21
2.2.2	Legacy LDAP App With Client Attributes Declaration .....	23
2.3	Attribute Authority Manager.....	24
2.4	Dynamic Privacy Policy and Attribute Exchange.....	26
2.4.1	Benefit.....	27
2.4.2	Dependencies with Other Use Cases .....	27
2.4.3	Details .....	27
2.5	Privacy Auditor .....	29
2.6	Deployment Cases.....	30
2.6.1	Mapping and Minimization Scenario.....	30
2.6.2	Break-Glass Scenario.....	30
2.6.3	Multiple-Claims Status .....	31
2.6.4	Propagation Between Attribute Authorities.....	31
2.7	Access Control and Delegation Cases.....	31
2.7.1	Identity Life-Cycle.....	31
2.7.2	Manager Scenario .....	32
2.7.3	Customer Care Representative.....	32
2.7.4	User Revises Consent Rules .....	32
2.7.5	Power of Attorney / Authorized Agent.....	32
2.7.6	Anonymous Delegated Representative.....	32
2.7.7	Transfer of Authority / Professional Delegation.....	33
<b>3</b>	<b>Requirements.....</b>	<b>34</b>

3.1	Compile-Time Declaration.....	34
3.2	Abstracted/De-coupled API .....	34
3.3	Support for Existing Applications.....	34
3.4	Flexible Deployment .....	34
3.5	Attribute Authority Schema .....	34
3.6	Mapping / Obfuscation / Filtering / Minimization .....	35
3.7	Attribute Access Policy .....	35
3.8	Delegation .....	36
3.9	Trusted Policy Enforcement.....	36
3.10	Policy Management.....	37
3.11	Multi-source Arbitration .....	37
3.12	Multi-Protocol Support .....	37
3.13	Multi-Schema Support .....	37
3.14	Multiple Trust Domains .....	37
3.15	Secure Storage.....	37
3.16	Ability to Audit .....	37
3.17	Consent.....	38
3.18	Propagation Between Attribute Authorities .....	39
3.19	Expired Information .....	39
<b>4</b>	<b>Appendix .....</b>	<b>40</b>
4.1	End-to-End Scenarios.....	40
4.1.1	HealthCare Service Scenario .....	40
4.1.2	JAAS Authentication / Split Profiles .....	41
4.1.3	Partner Users Access Corporate Resources via SAML .....	43
4.1.4	Corporate Travel Booking Site (multi-mode).....	44

## 1 Introduction

This document provides the key business actors, use cases, and requirements for identity privacy and access policy. Actors define the key individuals that play a role or have a responsibility for the use of identity-related data in enterprise web applications. Following the actor definitions are use case scenarios that highlight scenarios that drive the requirements for a future specification.

This document is based on the contributions from Oracle's Identity Governance Framework submission to the Liberty Alliance (now known as Id-Governance), the Liberty Alliance Technical Expert Group use cases, and individual Liberty member case submissions.

### 1.1 Existing Materials

This document builds on information and concepts published originally by Oracle in November 2006 (<http://www.oracle.com/goto/igf>). For the purposes of this document, the Oracle materials form some of the foundational concepts and lexicon used in this document as well as the original use cases described.

The BMEG-MRD committee reviewed a requirements document submitted by France Telecom from the Open Mobile Alliance on Global Permissions Management. ([http://www.openmobilealliance.org/release\\_program/docs/RD/OMA-RD-GPM-V1\\_0-20060928-C.pdf](http://www.openmobilealliance.org/release_program/docs/RD/OMA-RD-GPM-V1_0-20060928-C.pdf)). The goals of this effort are, in many respects, specific to the mobile market but there are some overlapping requirements in the area of a policy language for attribute access control. Details of the OMA proposal have not been published and are not currently available to the BMEG.

We propose to continue to track this area as more information becomes available from OMA while Liberty Alliance continues development of the Identity Governance framework. We would also recommend that any proposal from OMA be based on existing policy language standards such as XACML 2.0. This avoids duplication of effort and will make future harmonization easier.

### 1.2 Introduction to the Id-Governance Initiative

Many governments are enacting legal legislation that will address issues of identity theft, privacy, and the appropriate use of information by service providers on the Internet. As identity information is exchanged across departmental, organizational, and jurisdictional boundaries, contracts, machine policy, and audit trails, between consumers and producers of identity-related data are critical to documenting the use of identity information and its secure exchange.

User-centric identity is changing the way identity information is distributed on the Internet. Instead of users entering their personal information into each web application they use, user-centric protocol applications allow users to provide their data via their desktop or via a third party identity provider. This greatly reduces the need for applications to store identity-related data, but changes the user-application relationship by introducing a new third party. That new third party is part of a new emerging service known as an Identity Provider.

While user-centrism is an important evolutionary change for the Internet, we must acknowledge the important role that authoritative providers of personal information have in applications architecture. They are responsible for maintaining high-quality information as well as protecting the privacy of the subjects for whom they hold information.

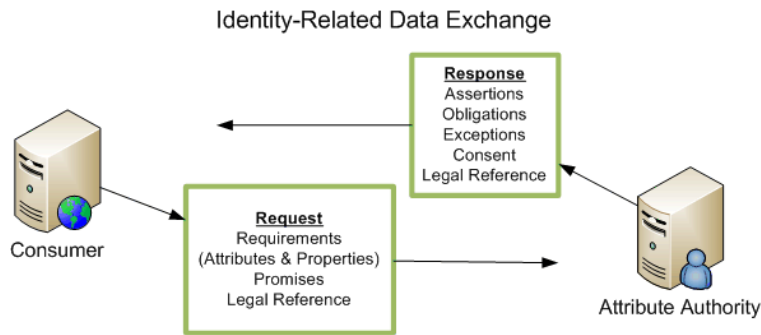
The intent of the Id-Governance initiative is to add policy enforcement to systems that produce and consume identity data in order to help all parties manage the risks and to provide a level of assurance to users that their privacy is being maintained by the parties to whom they are entrusting their information.

### **1.3 Use Cases: Privacy-Enabled Exchange of Identity Data**

The following is a simple overview of how policy is tied to the exchange of identity-related data. Items outlined in **green** are subjects of standardization or are related to how the standard is applied to a specific protocol through a profile.

#### **1.3.1 Simple Attribute Exchange**

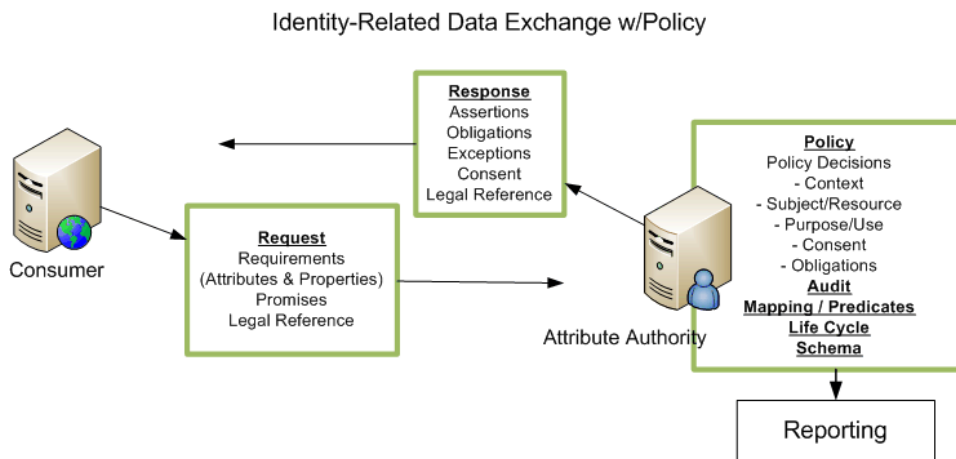
To exchange information between parties, whether directly, or via a user-agent (browser), a client application or a consumer issues a request for attributes and properties of an identity to an attribute authority. The request may include promises or privileges that are requested regarding the use of data received. For example, the consumer may wish to indicate a request to propagate information to certain parties, or to store or cache information for a period of time. The request may also include a reference to a legal document that defines the terms and conditions for sharing information between the parties.



**Figure 1**

On receiving a request, the attribute authority responds with a set of assertions that may include metadata such as restrictions, consent, or other legal documentation. The metadata is used to inform the client application about any transaction-level restrictions that may apply. If the client application requested an attribute or property that was not available, allowed, or filtered due to consent, an exception may also be included documenting why the information or operation was not performed.

### 1.3.2 Identity-Related Data Exchange with Policy



**Figure 2**

The decision to provide identity attributes and/or properties, filter, or modify a response to a client request is determined based upon policy enforced by the attribute authority. The attribute authority policy is able to say that in a specific context, a particular user may perform a specific action (read/write) against a subject record. In addition to using define policy, the attribute authority may also need to define mappings and predicates that map client attribute requests to the schema available within the attribute authority. Finally, the attribute authority is also responsible for the life cycle of identity-related data that is collected for a specific purpose and retained only for the period of time required. Policies, audit, mapping, life cycle, and schema characteristics are functions of a governance-enabled attribute authority.

### 1.3.3 Declarative Applications

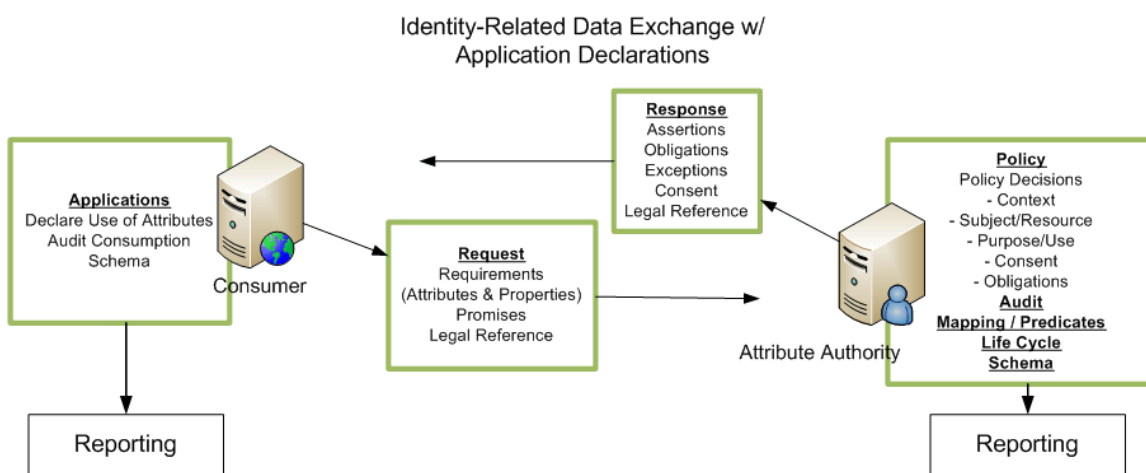
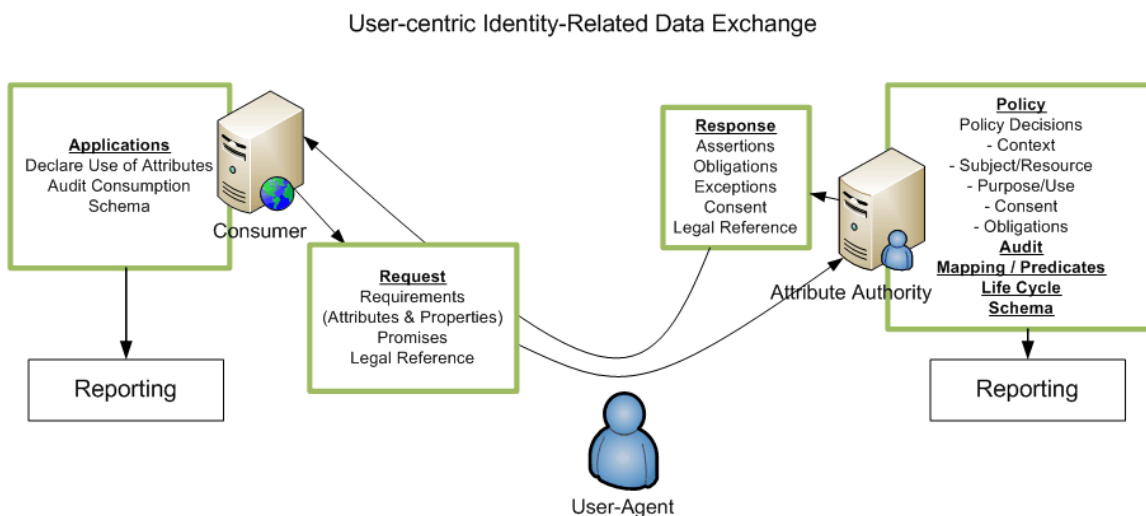


Figure 3

In order to define an attribute authority policy that enables a client application (consumer) to request and receive identity-related attributes successfully, the attribute authority should be able to understand what the client application requirements for identity-related data are (Figure 3). This information can be passed to the attribute authority in advance or as part of the exchange protocol. The combination of client attributes requirements and attribute authority policy gives auditors and identity managers an excellent understanding of where information is published and where it is consumed. In federated systems, it should also document the legal terms under which information was exchanged and provide attestation evidence that the rules were followed.



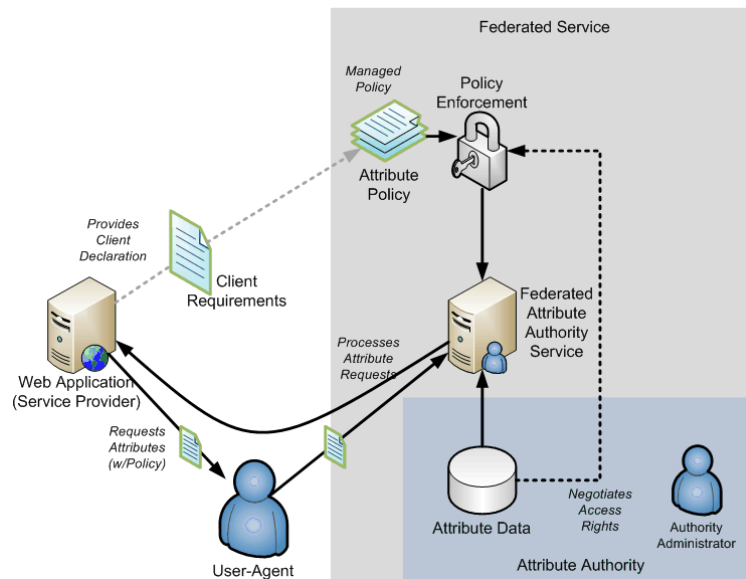
**Figure 4**

The governance of identity information extends not only to simple client-server relationships but also to three way relationships (Figure 4) where a user-agent (browser) is responsible for acting as the vehicle for exchange of information and may also be responsible for influencing the flow of information and aggregation. The advantage of user-agent-based interactions is that the ability to give users control over the flow of their information, adding much to the enablement of user privacy and control of disclosure of personal information.

### 1.3.4 Federated Exchange

Before web applications may consume information from a federated *Attribute Authority* service, it is assumed, but not always guaranteed, that the *Web Application* has some business relationship with the *attribute authority* service. Before information can be exchanged, the *Web Application* and federated attribute authority service must agree on protocol (how to access the information), the quality of information (whether it may be trusted), the schema (that may be mapped), and when it can be used.

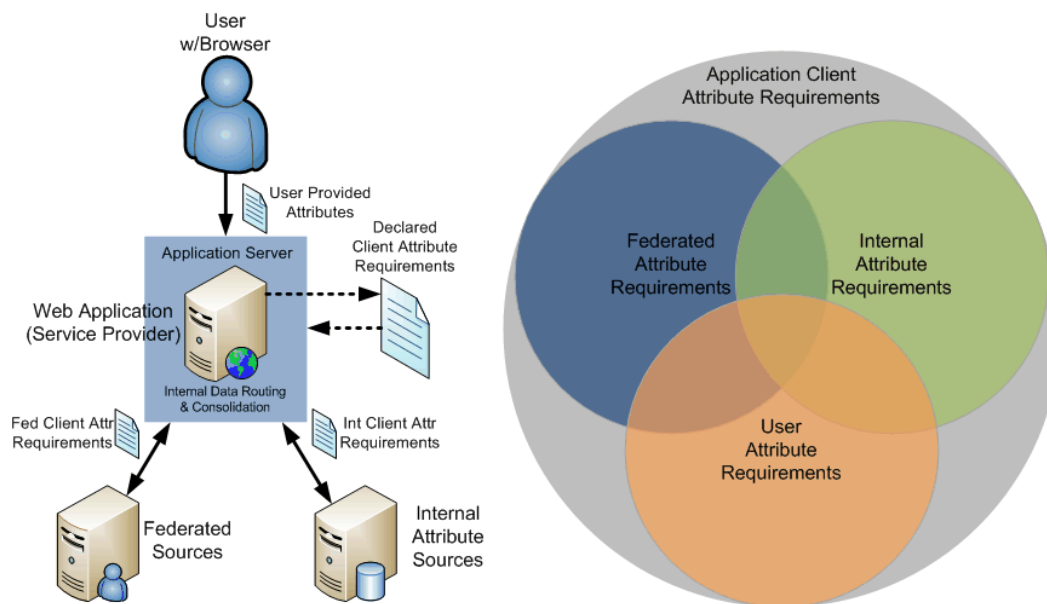
An identity governance framework suggests that a client *Web Application* will provide a declaration of *Client Attribute Requirements*, which specifies the identity-related data, required and the obligations and usage requirements the application has for using information (e.g., may propagate with specific business partners). On receipt of the *Client Requirements*, the *Identity Services Manager* works to identify appropriate authoritative sources of attribute data that meets the requirements of the client application. Once identified, the *Identity Services Manager* negotiates with the *Attribute Authority Administrator* to determine the appropriate *Attribute Policy*. Once defined, the *Web Application* may then request and/or update attributes from the *Identity Service*.



**Figure 5**

In the case of an interaction where a user-agent or browser is present, the user-agent (Figure 5) is responsible for propagating the specific attribute requests (e.g., Web Application Policy such as WS-SecurityPolicy). Depending on protocol, the user may be aware of this transfer, or it may be part of a referral request given to the user's browser. The Federated Attribute Authority Service must then use Attribute Policy to determine if the request is appropriate and then process the request.

### 1.3.5 Flexible Applications



**Figure 6**

A web application (see Figure 6) must be able to handle identity-related information that may be flowing to it from the end-user, from federated sources, and from internal attribute sources (e.g., databases and directories). When deploying an application, consider that while an application may have declared *Client Attribute Requirements*, those requirements will need to be distributed between the *User*, the *Internal*, and *Federated Sources*. This suggests the need for some kind of API, application server provider, or internal service that handles routing and consolidation between the various sources in a configurable and/or manageable way.

## 1.4 Terminology

Terminology for the identity governance framework includes definitions from the following sources:

**Identity Gang Lexicon** – <http://identitygang.org/moin.cgi/Lexicon>

**SAML 2.0 glossary** – <http://docs.oasis-open.org/security/saml/v2.0/saml-glossary-2.0-os.pdf>

**Internet Security Glossary, Version 2** – <http://www.ietf.org/internet-drafts/draft-shirey-secgloss-v2-08.txt>

**Liberty Glossary** –

[http://www.projectliberty.org/resource\\_center/specifications/liberty\\_alliance\\_specifications\\_support\\_documents\\_and\\_utility\\_schema\\_files/liberty\\_glossary\\_v2\\_0](http://www.projectliberty.org/resource_center/specifications/liberty_alliance_specifications_support_documents_and_utility_schema_files/liberty_glossary_v2_0)

Note: many of the definitions marked with an "\*" below are taken directly from the SAML Core specification [SAML-CORE]. They may have been edited for clarity in the context of identity services.

**Application Deployment Manager** – The application deployment manager (ADM) is the person responsible for deploying a given *web application* in a specific enterprise environment. This person is responsible for configuring the application server to host the application.

**Attribute** – A property of a *Subject* that may have zero or more *attribute value assertions*. Example: GivenName or TelephoneNumber are attributes that designate a person's first name or telephone number.

**Attribute Value Assertion** – a value or values associated with an *attribute*. Example: "GivenName: Phil" or "TelephoneNumber: +1 604 555 1212."

**Attribute Authority** – An attribute authority is typically a business entity that holds information about *subjects* that is considered to be authoritative. Depending on context, the attribute authority may refer to the organization that owns, controls, or is otherwise responsible for the information held, or attribute authority may refer to a service acting as an Identity Service Provider (e.g., generated tokens or assertions). An application that either stores *attributes* or propagates *attributes* and/or *properties* may also be considered an attribute authority when the application switches roles from consuming to publishing identity information. Example: an HR system collects information from a user and other sources (e.g., database) and then stores the information for later use by itself or other applications. Examples of attribute authorities include:

"**User Managed Attribute Service**" - the user directly controls the *attributes* and/or *properties* being handed out from the service, irrespective of whether the *attributes* and/or *properties* are self-asserted, on the desktop vs. at a server, or whether some third party validation has been performed.

"**Third Party Managed Attribute Service**" - the attribute service is managed by some autonomous entity distinct from the user, this entity controls who can access *attributes* and/or *properties*, the user has some legal/business relationship with the entity but does not directly control to whom the *attributes* and/or *properties* may be handed out.

"**Autonomous Attribute Service**" - as above but the user has no direct relationship with the service. However, legislation and corporate practice constrain to whom the user data may be provided.

"**Enterprise Attribute Service**" – an attribute service managed by an enterprise. This service does not typically interact with the end-user and is, instead, accessed by management applications and other client applications. Examples of such services are directory services and databases.

**Attribute Authority Manager** - The businessperson who is responsible for a repository containing *attributes and/or properties* about *subjects*. This person negotiates the *attribute authority policy* under which *client applications* may use *subject attributes* under the attribute manager's responsibility.

**Attribute Authority Policy** – The rules under which *users* using particular *web applications* may access and update *attributes* and/or *properties* about *subjects* under the control of a particular *attribute authority*. Example: a principal using the self-service application may update his or her own password. A telephone book application may provide telephone number *attributes* and/or *properties* for *subjects* providing the *user* is authenticated.

**Client Attribute Requirements** – The list of *attributes, properties*, and their usage requirements as declared by a *client application* requesting *attributes* from an *identity service*.

**Client Application** – A business application often accessed by a *user* entity that collects, uses, and potentially stores *attribute value assertions* about *subjects*. Example: a web application such as a shopping site user purchasing or profile management service. Note, if an Identity Service Provider consumes information from another *Identity Service* provider, then it too becomes a Client Application to the other *Identity Service* provider.

**Consent** – The agreement of a *subject* to one or more business conditions or proposals. Consent may be recorded as a set of one or more *attribute value assertions*. An example would be a user consenting to sharing information with marketing partners, or an agreement to receive e-mails and notifications of new products and services, e.g., "shareMarketingPartners: true."

**Developer** – A developer is any person writing code for a *web application*. The developer can be an enterprise developer writing custom code or the developer could be writing code for use by many organizations in many scenarios.

**Identity Service** – A service that provides access to identity-related information about *subjects*. The service may be a web service provider (e.g., offering services via WSF, WS-Trust), RESTful provider (e.g., OpenID), or may be based on a specific protocol (e.g., LDAP).

**Identity Services Manager** – The person responsible for coordinating the flow of identity-related data in an enterprise and coordinating application requirements (specified in *client attribute requirement* documents) with business units that hold information about individuals (*attribute authorities*).

**Obligation** - A restriction or a requirement on the use of identity information that is provided by an attribute authority to a client application that the client is responsible for enforcing. Examples: "Do Not Cache," "Delete after 30 days," "Do Not Propagate," and "NotifyUserOnUse." It could also include some other secondary operation.

**Purpose** – a client application accessing an attribute authority may have several different types of interactions within the same application. Examples of this might be:

"PatientSelfService," "RecordsUpdateByPhysician," and

"EmergencyPhysicianPatientRecordAccess." A *client attributes requirements* specification may contain multiple purpose declarations, each with separate attribute and property requirements. An attribute authority policy may also include named "purposes" as part of the rule predicate conditions. Purpose is also related to the "legal" reason for collection, storage, and use of *attributes* and *properties*.

**Predicate** – a statement about *attributes* and/or *properties* whose truth can be evaluated. This could be a simple policy or could be a condition of *consent*. Example: attribute "shareMktgPartner" is "true," client application is "www.myapp.org.com" and subject is self.

**Privacy Auditor** – The person is responsible for verifying that *client applications* are only accessing allowed *subject properties* and/or *attributes*.

**Property** – A special type of *attribute* whose returned value is true or false. A property can be specified as a simple name, or might provide some matching property value. Examples include: "IsOverEighteen," "Last4DigitsSSN" is "1234" (a partial value match), or "NorthAmerica" is "CA," "US," or "MX" (matches one of a set of values).

**Principal** – An *entity* whose identity can be authenticated. [X.811]

**Resource** – XACML defines resource as a data, service, or system component. For the purposes of this document, this term refers to any addressable *subject*, *attribute*, *attribute value assertion*, and/or *property* in an *attribute authority*. An *attribute authority policy* is typically defined to protect a *resource*.

**Subject** – A *principal* in the context of a security domain. An *attribute value assertion* makes a declaration about a *subject*.

**User** – a natural person who makes use of a system and its resources for any purpose. Associated with the *user* is a set of *attributes* referenced collectively as a *subject*.

A *user* is the initiating *principal*, typically using a browser that interacts with a service provider and potentially one or more Identity Providers. The user initiates actions and is often required to provide personally identifiable information (a.k.a. claims) either directly (e.g., as in a web form), or indirectly through assertions from Identity Providers.

A user may be acting on their own behalf or may be acting on behalf of others. A user may therefore be subject to terms and conditions of a service provider and obligations stipulated by an Identity Provider.

In some cases, it is possible that a user is also a web service provider. Most often, this scenario occurs when a web service provider is performing a back-channel service request on behalf of another user.

**Web Application** – an application (usually based on HTTP) that provides services to a *user* using the Internet. For the purposes of this document, the web application is assumed to have some form of dynamic capability and to have some notion of *users*. The application might be RESTful, be web-services-based, or support some other dynamic content technology.

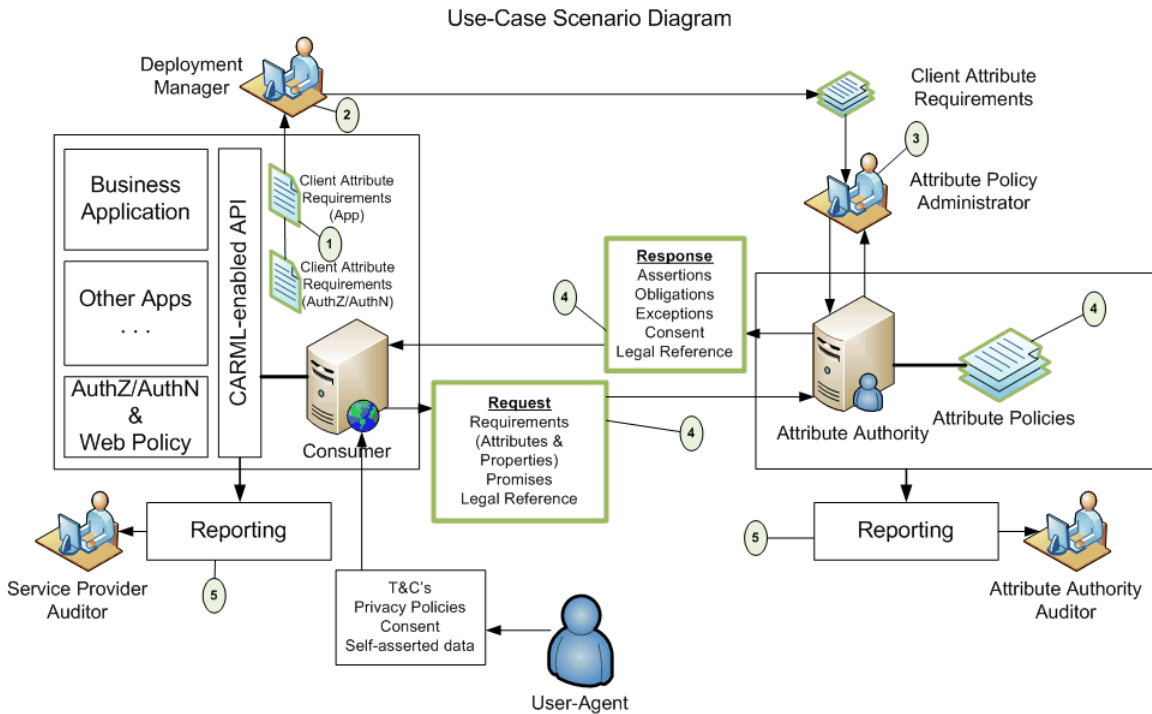
## 1.5 Scope

This document describes requirements for the client attributes declaration and attribute authority policy as a broad foundation to a policy-enabled identity governance system. Implementations of services and protocols are out-of-scope except to the extent that policy should be adaptable to multiple access protocols.

At the time of writing of this document, there are a wide variety of methodologies (mainly protocol-dependent) for collecting web site terms and conditions. The collection of terms and conditions is out-of-scope, but the ability to interpret different assertion types representing terms and conditions stored within a subject profile is within scope.

Similarly, the collection of consent information is out-of-scope. The interpretation of different consent assertion types (particularly by policy rules) is within scope.

## 2 Use Cases



**Figure 7**

Sections 2.1 through 2.5 use cases are based on the steps indicated in Figure 7.

1. Developer
2. Application Deployment Manager
3. Identity Services Manager/Attribute Authority Manager
4. Client application access to identity information
5. Audit Reporting

### 2.1 Developer Declarative Requirements

Developers create a declaration that documents an application's use of identity information.

Developers might create a client attributes declaration by:

- Using a client attribute requirements supporting API which builds the declaration and handles protocol level communication;
- Using a code scanner to infer client attribute requirements declaration; and,

- Constructing a client attribute requirements declaration by hand for use with existing compiled code.

## 2.1.1 Using a Client Attributes Requirements-aware API

A developer of a web site or service business application needs to consume and use identity-related data.

### 2.1.1.1 Benefit

Once all interaction functions are declared, the developer can then retrieve user records and/or set values of user records based on specific purpose declaration.

The developer is isolated from varying deployment conditions and configurations.

### 2.1.1.2 Details

<b>Title/ID</b>	Client Attribute Requirements Declaration Created using API
<b>Pre-Conditions</b>	A developer using an IDE writes application code accessing and setting identity-related information through a "client attribute requirements"-enabled API.
<b>Constituents</b>	Application Developer
<b>Use Case</b>	Using a "client attributes requirements" declaration created by an API, framework, or tool, the developer declares the attributes and properties (schema) to be used and the rights desired such as read or write – producing a client attributes requirements document. The developer may define multiple interactions with the attribute authorities. Each interaction defined documents a specific function or <i>purpose</i> for obtaining information from an <i>attribute authority</i> . Examples of purpose might be "UserProfileDisplay," "PublicProfileAccess," or "UserSelfService."
<b>Post Conditions</b>	A client attribute requirements declaration is produced that can be used by the application deployment manager to configure the application in its host environment.
<b>Alternate Courses of Action</b>	Depending on how the API is developed, the client declaration may be defined explicitly by the developer, or implicitly (e.g., through the use of a java object model).

## 2.1.2 Code-Scan Generated Client Attributes Requirements

In this scenario, a developer uses a code-scanning tool to look at an application's use of identity-related protocols (e.g., LDAP) to create a client attributes requirements declaration.

### 2.1.2.1 Benefits

By using a scanner tool, declarations for existing applications can easily be generated, assuming the application is using a protocol that is supported by the identity services and attribute authority servers.

### 2.1.2.2 Details

<b>Title/ID</b>	Developer Uses Tools to Generate Client Attributes Declaration
<b>Pre-Conditions</b>	<p>A tool like Java2WSDL is used to scan compiled java or source code.</p> <p>The scanned application is using a protocol that can be scanned and from which a declaration can be produced.</p> <p>The available attribute authorities can support the protocols required by the application.</p>
<b>Constituents</b>	Application Developer, existing application library, and/or source code.
<b>Use Case</b>	<p>A code scanner is used to infer a client attributes requirements document specification. Below is a possible scenario for a Java developer.</p> <p>Steps:</p> <ol style="list-style-type: none"> <li>1. Developer prepares her application for promotion from testing to a pre-production environment. She begins to package JARs, deployment descriptors, and client attributes requirements (CARML) files.</li> <li>2. She runs a tool that auto-generates the client attributes requirements file by crawling and analyzing the source. Developer is responsible for adding any client metadata (e.g., app's classification, etc) that could help the tool. For each interaction with the attribute service, the developer may declare a separate purpose name. Multiple purpose interactions can be declared in a single client attributes requirements specification.</li> <li>3. As the tool runs, it can raise warnings such as user searches for all attributes that might return too much data (i.e.,</li> </ol>

	search '*' in an HR user store).
<b>Post Conditions</b>	A client attribute requirements declaration is produced that can be used by the application deployment manager to configure the application in its host environment.
<b>Alternate Courses of Action</b>	If the code can not be scanned, the declaration may need to be hand generated.

### 2.1.3 Hand-Generated Client Attributes Requirements

In this situation, access to source code may be limited. In this scenario, a consultant or developer may use forensic documentation or protocol analysis to determine what the application is doing with regards to identity-related information. Once specified, the application deployment administrator can process a deployment request as described in the use case: [Application Deployment Manager \(Legacy LDAP App\)](#).

Notes: This scenario is supplied because it reflects how to handle legacy applications. Software vendors could then choose to document identity information usage by providing a client attributes requirements declaration with existing products eliminating the need for customers to use forensic analysis in their deployments.

#### 2.1.3.1 Benefits

Existing applications can be included in identity governance framework by having either the software manufacturer or an IT software analyst create a client attributes requirements declaration by hand.

#### 2.1.3.2 Details

<b>Title/ID</b>	Hand-Generation of Client Attribute Requirements for Existing App
<b>Pre-Conditions</b>	The application is fully documented or can be understood through protocol analysis, and/or log file analysis of data repositories used by the application. An assumption that the protocol used by the application is supported by the attribute authority policy server.
<b>Constituents</b>	Analyst who reviews the requirements of an application.
<b>Use Case</b>	An analyst reviews application documentation and, if necessary, uses protocol analysis to determine actual usage of attribute information by an application. Example: For an LDAP application, the analyst uses protocol

	analysis to observe the type of LDAP operations used and attributes affected.
<b>Post Conditions</b>	Once the client requirements declaration is written, it is handed to a deployment manager. Example: For an LDAP application, a virtual-directory acting as an attribute authority is configured to provide the correct schema and operations required by the application.
<b>Alternate Courses of Action</b>	N/A.

## 2.2 Application Deployment Manager

### 2.2.1 Client Attributes Requirements-Aware Client API Enabled

The application deployment manager of an application locates the client attributes requirements declaration file provided with the application and determines how identity information should flow into and out of the application.

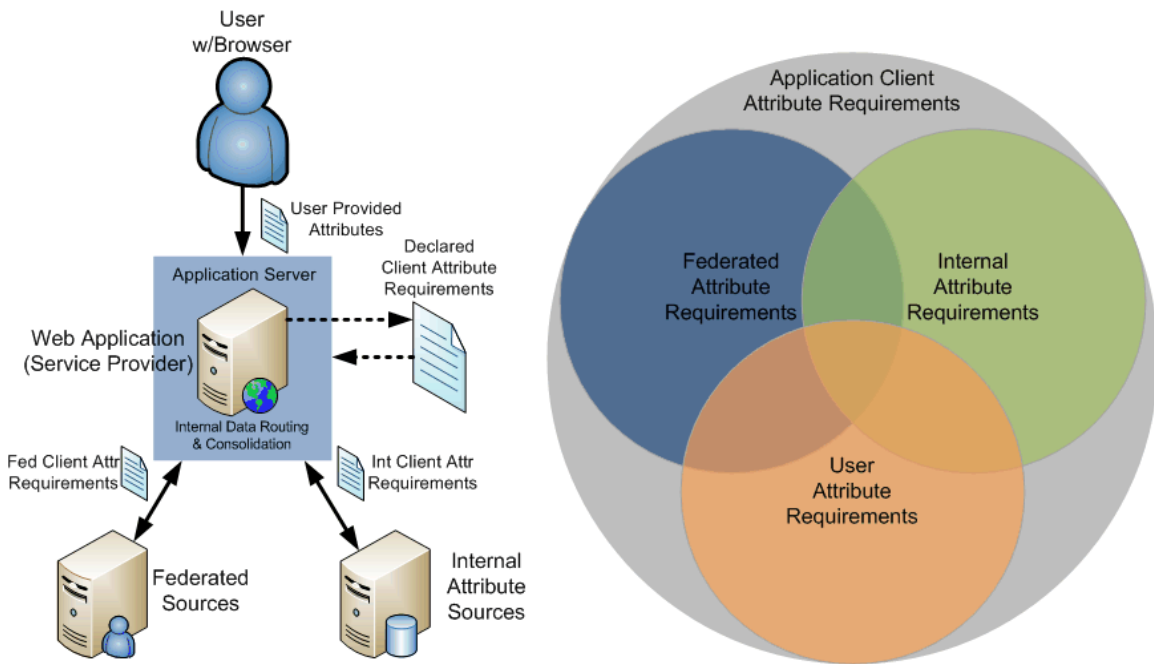


Figure 8

### 2.2.1.1 Benefits

The application deployment manager is able to deploy, flexibly, the application against a wide variety of deployment scenarios varying in both front-end and back-end protocols. Depending on implementation, the deployment manager may be able to configure support for multiple protocols at the same time. This might, for example, enable old forms-based users to transition to newer user-centric protocols.

### 2.2.1.2 Details

<b>Title/ID</b>	Deploying an Application using Client Attributes Requirements-Enabled API
<b>Pre-Conditions</b>	<p>The application has been written using a client attributes requirements-enabled API that enables the application to adapt, more flexibly, in a multi-protocol environment.</p> <p>As part of the application deployment files (e.g., war file), the client attributes requirements declaration file is available.</p>
<b>Constituents</b>	<p>The application to be deployed is installed by an application deployment manager.</p> <p>The deployment manager works with various attribute authorities to configure available sources of identity attributes.</p>
<b>Use Case</b>	<ol style="list-style-type: none"> <li>1. The deployment manager reviews the client attribute requirements and determines the appropriate information sources and flows. The manager must determine what will come directly from users, what will come from federated sources, and what will come from back-end sources.</li> <li>2. The deployment manager creates a new client attribute requirements file for each source and provides it to the source owner (attribute authority). The attribute authority reviews the declaration and determines the appropriate mapping and policy to fulfill the request. Note: It is assumed that the appropriate legal documentation has been developed and can be documented as a URI in the configuration files for audit purposes.</li> <li>3. The deployment manager determines what information will be supplied by the end-user (e.g., via InfoCards) and what information will come from the identity service. The manager adjusts the server security configuration (e.g., such as WS-Policy) to trigger the appropriate user-centric mechanism to provide data. Depending on configuration, the application</li> </ol>

	<p>may be configured to request information directly from a user using different methods such as forms and/or protocols such as ID-WSF, WS-*, and OpenID.</p> <p>4. The deployment manager configures the application host environment to connect the appropriate back-end or federated data sources based on the declarations previously provided to the attribute authorities.</p>
<b>Post Conditions</b>	Appropriate auditing and logging can now be configured.
<b>Alternate Courses of Action</b>	<p>Missing Attributes or Refused Requests</p> <p>In the case that an attribute authority refuses a request, the deployer must either locate another appropriate source or determine if a mapping or default value can replace refused or missing schema values. If none are available, the deployer will have to consider alternate courses of action such as application configuration options, or obtaining the information directly from the user.</p>

## 2.2.2 Legacy LDAP App With Client Attributes Declaration

The difference between this scenario and the previous scenario is that the client application is using LDAP rather than a client attributes requirements-aware client API that uses some simpler or trivial identity protocol. Because of this, the identity service required to support this requires an LDAP-based identity service. Further, because of the custom schema requirements of each client application, it is likely that some level of virtualization would be required in the LDAP service.

### 2.2.2.1 Benefits

This case is intended to discuss how protocols such as LDAP can be supported with pre-existing applications.

### 2.2.2.2 Details

<b>Title/ID</b>	Deploying an LDAP Application with Client Attributes Declaration
<b>Pre-Conditions</b>	An LDAP-enable application is available to be deployed. Along with the application, a client attributes declaration is provided.
<b>Constituents</b>	The application to be deployed is installed by an application deployment manager.

	<p>The deployment manager works with various attribute authorities to configure available sources of identity attributes.</p> <p>The deployment manager may use a product such as a virtual directory to provide LDAP access to attribute sources. The virtual directory may or may not act as an attribute authority itself.</p>
<b>Use Case</b>	<ol style="list-style-type: none"> <li>1. The deployment manager reviews the client attribute requirements and determines the appropriate information sources and flows. The deployment manager uses the client attributes requirements declaration provided with the application and presents it to the attribute authority. The attribute authority reviews the declaration and determines the appropriate mapping and policy to fulfill the request. In the case of an LDAP virtual directory, the attribute authority configures view and schema optimized for the client LDAP application. Note: It is assumed that the appropriate legal documentation has been developed and can be documented as a URI in the configuration files for audit purposes.</li> <li>2. The deployment manager configures the application host environment to connect the appropriate virtual or plain old directory source based on the declarations previously provided to the attribute authorities.</li> </ol>
<b>Post Conditions</b>	Appropriate auditing and logging can now be configured.
<b>Alternate Courses of Action</b>	<p>Missing Attributes or Refused Requests</p> <p>In the case that an attribute authority refuses a request, the deployer must either locate another appropriate source or determine if a mapping or default value can replace refused or missing schema values. If none are available, the deployer will have to consider alternate courses of action such as application configuration options, or obtaining the information directly from the user.</p>

### 2.3 Attribute Authority Manager

The attribute authority manager is the person who is responsible for a repository containing personally-identifiable information about individuals. This person negotiates which applications may use identity-related data for which they are responsible and under what conditions.

Because attribute authorities hold confidential data, they are very concerned about the security of the information and the conditions under which it may be used. This motivates the need for administrative policies over identity-related data. These include rules for application access, constraints on users on whose behalf data is being sought, company policies about use of personal or corporate data, and legal and regulatory requirements. In addition, availability of user or provider consent may also be the subject of administrative policy.

Some Sample Rules:

1. Any authenticated employee can read employee name, titles, corporate office, and phone numbers.
2. Any customer can read and update their home address and phone numbers in the customer repository. A customer service representative can also read and update these attributes on behalf of the user.
3. User name and e-mail addresses may be provided to marketing applications provided explicit user consent is available. Marketing applications should not propagate or cache this information.
4. A help-desk operator can only see a portion of the credit card details of a subject with whom he or she is currently interacting on the phone.

### 2.3.1 Benefits

Based on a client attributes requirements declaration, the attribute authority is able to configure an access policy and mappings that meet the needs of the client application. Note that, in some cases where information cannot be provided, this might result in outright rejection, or providing a dummy value that meets the client application's needs and is acceptable to the business application owner.

### 2.3.2 Details

<b>Title/ID</b>	Attribute Authority Manager Defines Policy
<b>Pre-Conditions</b>	An application deployment manager provides the Attribute Authority a client attribute request declaration.
<b>Constituents</b>	The application deployment manager and attribute authority manager.
<b>Use Case</b>	<ol style="list-style-type: none"> <li>1. When contacted by the Identity Services Manager, the manager negotiates the necessary legal terms and defines policy acceptable to both parties.</li> <li>2. If necessary, in order to create the schema and values required</li> </ol>

	<p>by the client, the attribute authority may have to define mapped or virtual values upon which the AAPML policy can be applied.</p> <ol style="list-style-type: none"> <li>3. The attribute authority administrator configures the attribute authority policy to be enforced. Each function defined by the client is defined as an interaction in the attribute authority policy.</li> <li>4. Once the policy has been defined, the attribute authority may return a policy confirmation identifier to the deployment manager. This identifier can be used by the CARML client to identify requests to the attribute authority server.</li> </ol>
<b>Post Conditions</b>	The attribute authority server policy has been defined and is ready for operations from the client.
<b>Alternate Courses of Action</b>	In some scenarios, there may be no formal interaction between the application deployment manager and the attribute authority manager. In these cases, attribute policy is based on some generic or default access rules that clients may use. This might be the case in fully dynamic federation scenarios such as those covered by OpenID.

## 2.4 Dynamic Privacy Policy and Attribute Exchange

This use case describes attribute exchange between a web services client (WSC, business application) and web services provider (WSP, attribute provider) in different security domains. For example, a travel booking site provides, besides the ordinary travel booking, car rental services or event and tour booking services. The service provider composing the services out of other services needs to exchange the user's attributes with the service providers based on the user's consent. When providers exchange attributes, they need to manage the exchanged attributes appropriately.

In this scenario, the WSC and WSP have previously exchanged their policies (see Section 2.3) specifying what kinds of attributes to be exchanged and how to deal with them, and confirm usage of them. This exchange has been consummated in a legal agreement that documents and defines the exchange of information.

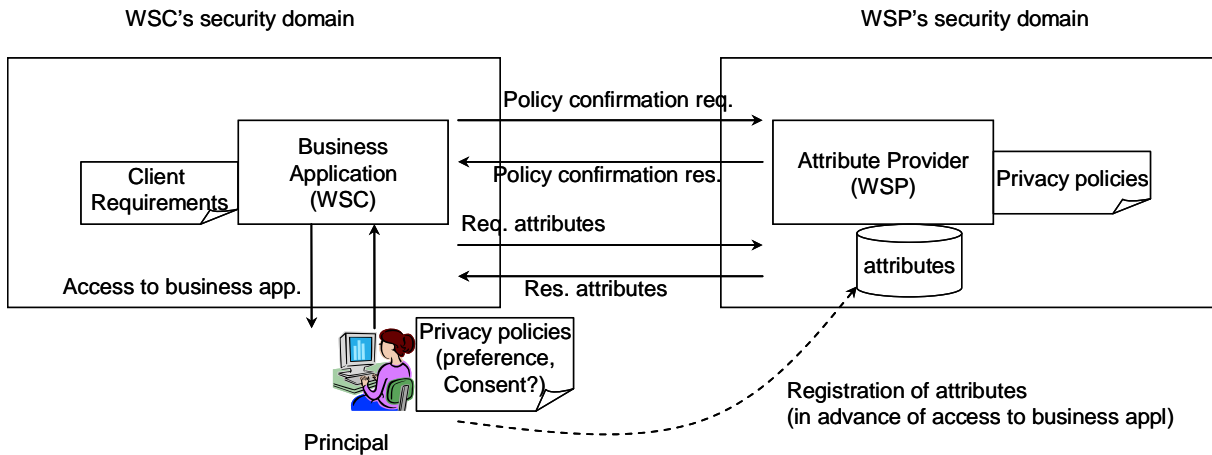


Figure 9

### 2.4.1 Benefit

WSC and WSP can reduce cost for maintenance of attributes and the risk of their leakage because they exchange limited attributes.

WSC and WSP can clarify their own responsibilities by means of a mutual confirmation about handling a user's attributes.

WSC does not obtain unnecessary attributes that it does not require while WSP can enforce a confirmed privacy policy on WSC.

### 2.4.2 Dependencies with Other Use Cases

The client application (WSC) and attribute authority (WSP) have already exchanged client attribute requirements and configured attribute authority policy.

### 2.4.3 Details

<b>Title/ID</b>	WSC and WSP Exchange Principal's Attributes after Client Attribute Requirements Exchange.
<b>Pre-Conditions</b>	WSP and WSC are in the same trust circle but in different security domains. WSP maintains Principal's consent policy for attribute exchange (e.g., the use of my address information must be limited for the purpose of booking hotel only once).
<b>Constituents</b>	Principal, WSC, WSP
<b>Use Case</b>	Dynamic Policy Exchange (Optional): 1. The principal requests WSC's service (such as car rental site).

	<ol style="list-style-type: none"> <li>2. WSC requests the principal's attributes from WSP with WSC's client attributes requirements (e.g., the attributes and properties requested and the promises made such as "No Caching").</li> <li>3. WSP checks the WSC's client requirements with its own attribute policy (e.g., the attributes may be shared in WSC's group companies) and the principal's consent policy and verifies whether it can provide the attributes with WSC.</li> <li>4. If the request is invalid or consent does not allow it, an exception message is returned. Otherwise, process continues...</li> <li>5. WSP sends a confirmation message about the policy verification with confirmation identifier.</li> </ol> <p>Attribute Exchange:</p> <ol style="list-style-type: none"> <li>6. WSC requests the attributes to WSP with the confirmation identifier and the client attribute "purpose" name.</li> <li>7. WSP sends the attributes to WSC. The response may include additional information such as individual attribute exceptions or obligations.</li> <li>8. WSC receives returned results and provides the service with Principal.</li> </ol>
<p><b>Post Conditions</b></p>	<p>WSC uses the received attributes while it follows the regulation(s) specified in the confirmation.</p>
<p><b>Alternate Courses of Action</b></p>	<p>At step 3, if privacy policies of WSP and WSC do not match, WSP returns an error message to WSC.</p> <p>At step 4, if the request is invalid, WSP can also send an acceptable policy (e.g., only Principal's age information is available) to WSC. When WSC receives it, WSC may check that the WSP's policy can satisfy another client attributes requirements.</p> <p>Once the confirmation of handling the attributes is shared between WSP and WSC, they can skip the process of policy exchange, and proceed to exchange of the attributes based on the confirmation.</p> <p>At step 6, the client does not have to specify the confirmation identifier. However when not specified, the attribute authority may have to perform extra work to match client through authentication of the client application and the specified "purpose."</p>

## 2.5 Privacy Auditor

Example: Health Information Scenario

A Privacy Auditor verifies that non-Health-Related (HR) apps can not access protected health information data.

- Mandated corporate policy: Only HR client apps can access Personal Health Information (PHI).
- Auditor is tasked with determining if any non-HR apps are out of compliance.

### 2.5.1 Benefits

The use case explains how an Auditor could use client attributes requirements and attribute authority policy files to see, quickly, if there is any misuse. This could be done manually or via automated tools. The manual steps are described here.

### 2.5.2 Details

<b>Title/ID</b>	Privacy Audit in Healthcare
<b>Pre-Conditions</b>	Client attribute requirements and attribute authority policies, including all historical variants, are available along with client and policy server logs.
<b>Constituents</b>	The privacy auditor and relevant client application and attribute authority owners.
<b>Use Case</b>	<ol style="list-style-type: none"> <li>1. Auditor gathers attribute authority policy files for the Identity Store(s) for which she is responsible. She makes note of each user property marked as containing PHI (Private Health Information).</li> <li>2. Auditor gathers the client attribute requirements files for each non-health-related application.</li> <li>3. Auditor is able to cross-reference the PHI properties (from the attribute authority policy) with the attributes consumed as declared in the client attributes requirements files.</li> <li>4. Auditor can cross-reference the client attribute requirement functions and attribute authority interaction files and see if any of the PHI attributes (as declared/tagged) in the attribute authority policy are consumed by the non-HR client.</li> <li>5. The auditor can review usage logs to verify the correct functioning of policy.</li> </ol>

<b>Post Conditions</b>	Auditor should be able to produce an automated report confirming correct policy configuration and usage.
<b>Alternate Courses of Action</b>	In some federated cases, the client logs may be separated from the attribute authority logs. Each security domain must be able to audit usage within their domain.

## 2.6 Deployment Cases

The following cases represent example modifiers to the above scenarios (particularly Section 2.4) where the deployment environment may impact requirements.

### 2.6.1 Mapping and Minimization Scenario

A client application declares its requirements for the attributes and properties that it needs in the form of a client attributes requirements document. An attribute authority accepts this declaration and determines what schema it has that potentially maps to the clients requirements. If a simple attribute mapping is not available, a mapping may need to be defined that meets the needs of the client application.

Example types of mappings include:

- **IsOver65** – A property calculated by subtracting a birth date from today's date and determining if the value is > 65 and returning the value as a true/false property.
- **Last4DigitsSSN matches 1234** – This is another property test in which the attribute authority policy accepts a test value "1234" and compares it with a similar value or the actual value. If matched, true is returned.
- **Location** – The city name is mapped to a region name.
- **MatchesOneOf "a," "b," or "c"** – This property test checks to see if a value matches one of a set of values provided by the client.
- **International Telephone number** – a North American value of 604 637 9135 might be mapped to +1 604 637 9135.

### 2.6.2 Break-Glass Scenario

In the break-glass scenario, a special condition might be triggered that allows specified individuals to access private information. When triggered, this special condition might cause additional logging to occur. An example of this case would be an emergency room doctor accessing a patient's medical history. Under normal circumstances, that doctor could not access the information. However, in the context of the ER, the doctor may access the information provided either some other extended test is passed (verification from another identity) or some other process like extended logging is triggered.

### **2.6.3 Multiple-Claims Status**

A lawyer named Joe is visiting the Justice Department. When he attempts to sign in from his firm's Identity Provider, the Justice Department indicates that in order for Joe to be granted access to its systems, the department requires that Joe's status as a lawyer in good standing, in addition to proof of authentication, be provided.

The combined authentication and standing claims could be provided by having Joe, serially, go to both his authentication provider and then to his bar association to accumulate the necessary tokens. Alternatively, Joe could merely be required to provide an authentication and the Justice Department can verify his status directly with the bar association.

Example: Joe uses his IdP to authenticate and the Justice Department independently verifies Joe's status as a lawyer.

This case highlights the need to confirm multiple attributes (or claims) about an individual. These claims could be submitted by an Identity Provider, or may be gathered from a back-channel source.

### **2.6.4 Propagation Between Attribute Authorities**

Due to conflicts in service level agreements between a web service provider and an attribute authority, the attribute authority agrees to allow the web service provider to set up a new attribute authority which will hold a copy of source attribute authority's information. The new attribute authority will provide the data on behalf of the source authority with the appropriate service levels.

The second attribute authority agrees to maintain logs and provide audit/usage information back to the primary authority.

## **2.7 Access Control and Delegation Cases**

These cases define relationships where a user is allowed to access and/or update another subject's attributes.

### **2.7.1 Identity Life-Cycle**

A user has provided information under specific terms and conditions that require the information to be disposed and/or archived when no longer needed or after some period of inactivity. Such a requirement may be due to legal restrictions (e.g., European Union) or simple consent terms specified by the user.

These obligations might include reminder notification, archival, and disposal of information.

Such a requirement might be affected by legal legislation, pending litigation (which might temporarily delay deletion), enterprise policy, or requirements for non-repudiation.

## 2.7.2 Manager Scenario

In this scenario, an enterprise's HR systems indicate the relationships between employees and their management. A manager using an application such as a calendar system wishes to access identity-related information about one of their employees. The calendar application makes the request on the manager's behalf to the Attribute Authority. The Attribute Authority evaluates its rules and confirms that the manager is able to access another employee's record for a specific set of attributes.

In this case, the attribute authority policy rule specified must be able to define a relationship between two different subjects. In this case, the relationship is expressed by retrieving attribute data from HR about the target subject (the employee) that allows the authority to confirm that the manager is the employee's manager.

## 2.7.3 Customer Care Representative

A customer care representative is able to update a user's account when the user is not online. The records at the customer profile system, an attribute authority, capture that the updates were created by the customer care rep "on behalf of" the customer. Audit captures this information via policy.

## 2.7.4 User Revises Consent Rules

Jane communicates with her identity data provider (which acts as an attribute authority) and reviews her registered data access consent records and chooses to revoke consent to access her phone number for Joe's Pizza Shop. While there, she also notices that Paul has not been granted access yet, so she decides to proactively grant him access (hoping he will call her one day).

In this case, the attribute authority has several application-defined rules that allow the subject to modify their consent settings which allow attribute authority policy to enable access to different parts of a person's attribute information depending on their relationship to the subject.

## 2.7.5 Power of Attorney / Authorized Agent

Bob delegates access to his Government of Canada tax page to any accountant from KeepYourBooks, Inc. Bob doesn't specify delegation in terms of individual accountants but rather authorizes any member of the accounting firm, trusting that KeepYourBooks will allow only appropriate representatives to access the information and/or manage access.

## 2.7.6 Anonymous Delegated Representative

A social worker registers a disabled citizen at a university for an upcoming course. The university is able to recognize that the registration was performed by someone authorized to act on the citizen's behalf, but NOT the identity the particular social worker.

### **2.7.7 Transfer of Authority / Professional Delegation**

In normal terms, "delegation" refers to enabling one person acting on another person's behalf. In professional situations, this may not be appropriate.

Frank is a doctor with a private practice. He hires another doctor, Rosemary, to act on his behalf as his "Locum" while he is away on holidays. Rosemary is acting under her own professional authority. Rosemary is granted access to Frank's patient records while he is on holidays. However, legal responsibility for access is held by Rosemary instead of Frank.

## 3 Requirements

### 3.1 Compile-Time Declaration

The developer shall be able to provide a requirements declaration (use in place of CARML) for attributes and properties used by the application along with requested rights (cache, propagate) and permissions (e.g., read or write).

### 3.2 Abstracted/De-coupled API

The developer requires an API that de-couples client application requirements from the run-time environment. The API should be able to handle, automatically, information flow variability between end-users and various back-end sources of information. The API should also be able to insulate the developer from variability in protocol and data source type.

### 3.3 Support for Existing Applications

Existing applications written prior to the specification (or even user-centric protocols) may be adapted by either manually writing client attribute requirements declarations, or by code scanning (if available). This assumes that the Identity Service is capable of providing support for protocols used by the application to access data (e.g., LDAPv3, WS-Trust, ID-WSF).

### 3.4 Flexible Deployment

The deployer may need to edit the client requirements declaration to include application intent documentation such as a description of use and legal agreements regarding the applications basis for using personally identifiable information. Typically, intent information would be additional documentation to a "purpose" as defined by a developer.

Additionally, the deployer may need to work with an attribute authority manager to determine appropriate web application policy. It may also be possible that the web policy server may also provide header information important to the application.

The deployer determines how client attribute requirements will be fulfilled whether from the end-user, from federated sources, or from internal or locally provided sources.

### 3.5 Attribute Authority Schema

The attribute authority should be able to provide a schema catalogue, and an expression of the types of functions it can support. This will assist the identity services manager to reconcile client requirements declarations with available attribute authorities.

### 3.6 Mapping / Obfuscation / Filtering / Minimization

The attribute authority must be able to establish mappings between various information sources and defined client requirements. The mappings should be able to define:

- Name Translation – converting differences between attribute names.
- Property Translation – the ability to support properties that return true or false based on a Boolean filter or mathematical calculation. There are several types of properties:
  - Condition – determine if a particular condition is true; e.g., IsOver65 is a property calculated by subtracting a birth date from today's date and seeing if the difference is > 65 years.
  - Match – determine if a supplied value matches some condition, e.g., Last4DigitsSSN of 1234 matches.
  - Set – determine if the matched condition is contained within a set of supplied values, e.g., NAMember is defined as one of "CA," "US," or "MX," which implies a mapping where the attribute country is checked to see if it is one of the three supplied values to return a true or false.
- Masking – the ability to obfuscate or mask part of a value that is returned, e.g., the last four digits of the SSN is 1234.
- Format Conversion – the ability to perform some kind of translation to handle differences in value formats or syntax between providers and clients, e.g., a North American telephone number of "604 637 9135" is mapped to "+1 604 637 9135."
- Default or Override – Another requirement for mapping may occur from attributes that are used for consent processing; e.g., within a specific domain, an attribute "OkToShareWithMktgPartners" might have to be filtered or set to false when propagated to a marketing partner system.

It is not necessary that a standard for the implementation of mappings be defined. It is only necessary that mappings must be supported that include, but are not limited to, the functions described above.

### 3.7 Attribute Access Policy

The attribute authority manager must be able to set policies on what applications may access and/or modify data under specific contexts. The policy applies to both attributes stored in a repository and mapped data as described in Section 3.5.

Policy requirements include but are not limited to:

- Multiple policies or interactions can be defined for each application.
- Policies can contain usage and privacy restrictions.

- Policy rules can be triggered based on attribute conditions within a subject record (e.g., consent).
- Ability to define what actions may be performed against specific attributes or properties including set and get (including support for single or multi-value attribute assertions).
- Ability to define the context under which a rule applies. This includes defining the application, the user, the subject being retrieved, as well as the action being performed.
- Context can include requirements for one or more claims from a user or application (use case 2.6.3).
- Context must be able to support relationships between users such as defined by delegation requirements.
- A policy can be declared as requiring extended event logging (break-glass scenario).

### 3.8 Delegation

Delegation settings may allow one user to access another subject's record to perform certain operations. This forms part of the context determination for an attribute policy rule. Note: There may be similar web policy rules that are also enforcing delegation relationships within the application.

Types of relationships include (but are not limited to):

- Managerial – an attribute that defines a subject's manager can be used to indicate that the manager subject has certain rights.
- Role – the presence of an attribute assertion that defines a role may be used to allow a user access to a subjects record. Examples of roles include job functions (e.g., "CustomerCare") or may express organizational relationships expressed through attribute assertions (e.g., organization is "KeepYourBooks Accounting").
- Attribute List – an attribute value may list specific subjects or may allow access to specific functions. This value might be maintained by a user-profile system to allow the user to name other parties to have access to the user's record (see use case 2.7.4).

### 3.9 Trusted Policy Enforcement

Attribute authority policy can be enforced directly by an attribute authority, or by a trusted identity provider or gateway.

### **3.10 Policy Management**

The identity services manager or attribute authority should be able to define new sources in connection with attribute authorities. The attribute authority manager must be able to negotiate and manage new attribute authority policy rules based on specific client attribute requirements declaration(s).

### **3.11 Multi-source Arbitration**

The attribute authority should be able to determine which attribute source or sources are best for each application.

### **3.12 Multi-Protocol Support**

The identity services manager should be able to connect multiple types of attribute authorities, including non-web protocols such as SQL or LDAP.

### **3.13 Multi-Schema Support**

It is important to understand that the Identity Service has no one single schema (standard or not). The Identity Service should be capable of supporting many different client attribute requirements and thus multiple client schemas. This allows for de-coupling and separation of client application schema from the service or attribute authority repositories. This de-coupling strategy dramatically reduces the need to customize, re-write, or specially provision new enterprise applications.

### **3.14 Multiple Trust Domains**

Attribute authorities can choose to enforce local attribute authority policy or allow trusted Identity Providers to do so on their behalf. It is quite possible that there may be multiple layers of policy enforcement between the web service provider client application and the source attribute authority.

### **3.15 Secure Storage**

A client application receiving identity information is obligated to store information in a secure form. For example, the information must be encrypted so that data administrators may not access the data directly from the data store.

### **3.16 Ability to Audit**

The privacy auditor should be able to verify the functioning and compliance of attribute authority rules and general integrity of the policy system. This includes having access to declarations, policy, and logs.

There should be multiple log levels with a configurable amount of detail. Each log level may reveal different amounts of information regarding the requestor, the calling application, the operation requested, the attributes impacted, and, potentially, the attribute values. In some cases, if attribute values are logged, the values may require masking to maintain privacy of information when published to the audit log.

In the event of an extended logging policy (see requirement 3.7 and use case 2.6.2), a more detailed log level may be specified.

### **3.17 Consent**

Consent is the idea that individuals, about whom attributes are describing assertions, should have some say in the propagation of their identity-related information. This is an important and somewhat complicated aspect for I-R source providers and service providers (relying parties).

**Specificity of Consent:** Consent is given by individuals for information to be used for a specific purpose or purposes for a period of time.

**Consent and Quality:** Gaining consent by itself from an individual does not improve the accuracy of the information.

**Unidirectional Liability of Consent:** Consent by an individual only grants that I-R-related information may only be used for a specific purpose. Consent does not imply that a Service Provider (or relying party) is released from liability for using that information. Service Providers must employ other means to guarantee the accuracy of the I-R information.

**Implied Consent:** Information which is volunteered by clients without a specific contract means that the service provider directly receiving the information may have access to the information but may not share it with any other service or system.

**Increasing Scope of Consent:** If a new use for I-R data is needed, then consent must be re-obtained for the new purpose.

**Ability to Audit Consent:** Service providers need to be able to prove that they have appropriate consent for the information obtained.

**Grandfathered Consent:** Many internal systems, such as HR systems, already have I-R-related data. As long as the usage of the information falls within the original scope of understanding and agreements for which the information was originally gathered, then the information can be used. If it falls outside the original scope, this is an increasing scope of consent situation.

Attribute authorities will manage consent and its interpretation and impact on attribute use as it applies to attribute policy. Consent information can also be propagated as attributes for web policy or as data for use within web applications.

Note: How consent is collected and stored is out-of-scope for the current Id-Governance use case MRD. This objective of this draft is to be able to interpret any piece of information as consent as requirements dictate.

Consent information has the purpose of providing a more granular access control constraint concerning the release of data about an individual user. Instead of making the statement:

(A) - Every user account number and bank balance can be provided to every application in the sales division.

The following statement might be more appropriate:

(A\*) - User account number and bank balance can be provided to every application in the sales division, provided user consent has been obtained under agreement

`urn:consent:example-agreement-09-12-2006`

The understanding, here, is that the attribute provider would maintain a record for the user that maps to `urn:consent:example-agreement-09-12-2006`.

### 3.18 Propagation Between Attribute Authorities

When information is propagated between attribute authorities and identity clients (particularly across administrative domains), it is quite possible that some identity attributes, particularly those regarding consent and delegation, will likely need to be mapped (see section 4.4).

When crossing between different administrative domains, context for confidentiality and privacy will change. When information flows across domains, the attributes (including consent and obligations) must be adjusted to account for changes in legal, business, and application contexts. Changes in business or security domains may drive the need for interpretation and mapping:

- Interpretation of values (e.g., Mgmt Level A = Mgmt Level 8),
- Interpretation of quality or trust,
- While Entity A can share with its Partner B, B cannot further share,
- Jurisdiction – information may not be stored across legal boundaries, and
- Ability to cache or store and for how long.

### 3.19 Expired Information

If a policy has been defined that determines information expiry, then access to information should not be granted to that information once the expiry period has been exceeded. After that time, potentially only management functions such as archive, deletion, or recovery should be allowed access.

Depending on jurisdiction of the subject record, different policy requirements may need to be enforced with regards to deletion vs. archiving and recovery.

## 4 Appendix

### 4.1 End-to-End Scenarios

#### 4.1.1 HealthCare Service Scenario

*[Source: HP]*

A Healthcare Service Provider collects personal and medical information of patients to provide diagnostic services, to answer patients' online health-related queries, to carry on research activities, and, potentially, to use sanitized data for marketing purposes. Data is stored in various data repositories, including RDBMS databases, LDAP directories, multimedia data storages, and data warehouses – depending on business and operational needs.

This sensitive information is collected via a secure web portal during patients self-service registration, user accounts management, and subsequent updates of their profiles. Users (patients) disclose their data along with their expressions of consent to use their data for predefined purposes (e.g., research and query-answering but not marketing),

This data can be accessed by means of the same portal (and applications/services running on its back-end) and a set of federated services (hosted by other companies involved in diagnostic exams, data processing, query answering, etc.). In the latter case, the healthcare service provider acts as a "Identity Provider."

Various authorized people (within the organization and outside it) can access portions of sensitive and personal data, for different purposes, via the portal and/or programmatically via federated services/applications. These could include GPs, practitioners, researchers, clerks, call-center/help-desk employees, etc.

Services/applications can provide multiple functions, each of them requiring access to specific sets of attributes. The same application/service could be potentially accessed by different people but only a subset of their functionalities (and related accessible data) would be available, depending on people's roles (and, potentially, their intent), consent given by patients, and policies defined by the Healthcare Service Provider.

For example, a practitioner, with an implicit "Diagnostic" intent, can access only a patient's medical data (no other personal details such as address or location), assuming consent has been given by the patient,

For emergency reasons (such as the urgent need to contact a patient or intervene to help him), this practitioner or another one could bypass access control and retrieve the entire data, by explicitly declaring the "Emergency" intent and submitting a description of the reasons. This access is audited and later might require reconciling this with the patient.

A help-desk employee, for "support" purposes, could just access a subset of data, via a related application, retrieving, for example, "sanitized" login information and financial data (e.g., credit card number where only the last 4 digits are visible).

An external person with a "Researcher" role could just access, via a federated service, a subset of this data, with no explicit personal references – assuming consent for this purpose has been given.

In a Governance Framework, each application/service requiring access to sensitive data is associated to one or more CARML files. These files document the default behavior of this application/service, including required data and potentially allowed default intent.

At runtime, the context (e.g., role of the entity accessing the application, intent, other attributes such as location, etc.) determines the relevant type of attributes request that can be made by applications (and/or one of its functions). However, some contextual properties, such as Intent, could be "overridden," in strictly-audited situations (e.g., the Emergency case described above).

CARML requests (potentially instantiated with runtime properties, if required) are sent by the Service Provider to the Attribute Authority that will actually mediate the access to data depending on stated attribute authority policies and the current context (inclusive of the intent, role and other properties).

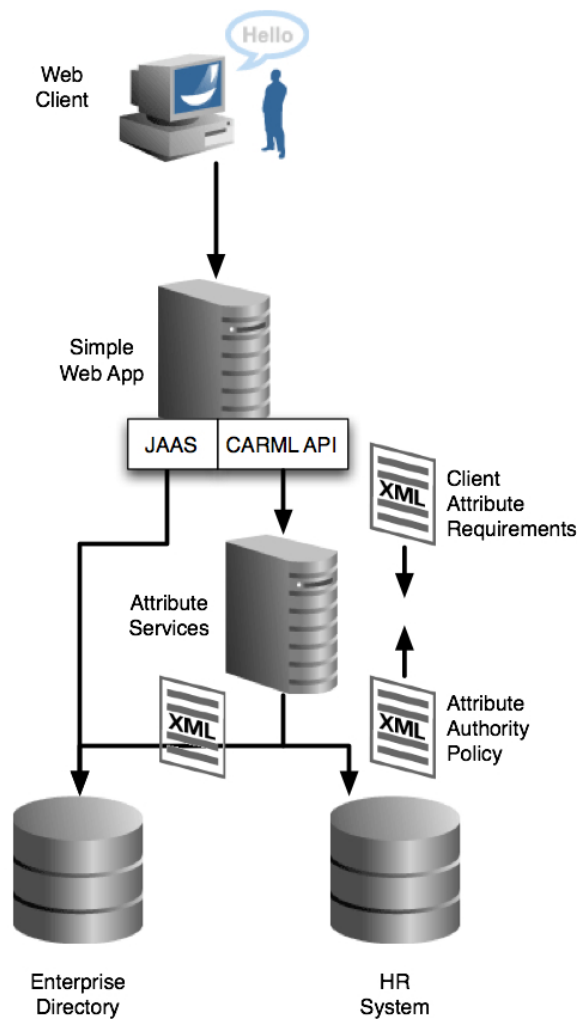
Decisions made by the Attribute Authority via attribute authority policies might dictate on-the-fly transformation of retrieved attributes (such as data minimization, filtering out part of the information, etc.). These decisions might be enforced by the policy enforcement point with the data repositories, when possible, or directly by the Attribute Authority.

The added value of the Governance Framework is ensuring compliance to policies and users' requirements by taking into account both static and dynamic information and aspects involved in the retrieval and processing of identity information.

#### **4.1.2 JAAS Authentication / Split Profiles**

*[Source: Oracle IGF]*

In this enterprise scenario, a web application for reporting employee vacation and sick time authenticates users using Java Authentication and Authorization Service (JAAS) communicating with an enterprise directory. The web application also requires information from the Human Resources system to determine appropriate user role information based on job classification and management chain relationships.



**Figure 10**

In this scenario, the web application server receives HTTP requests from the user. Using JAAS, the web application authenticates the user to the enterprise-directory as per typical application configuration. Information from HR regarding the user's job role and reporting structure will be used to enable delegation and other privileged management functions within the application, e.g., a manager cannot modify the records of an employee they do not manage.

The employee time management application has declared via CARML that it requires current job title, employee number, and reporting employees as well as an email address for the user. The email address is to be provided by the enterprise directory while the other information will be provided by the HR system.

The HR system has registered with the attribute service indicating, via an AAPML declaration, that employee number, title, and direct reports can be made available provided that the application has authenticated itself and the credential of the requesting user has been verified. The enterprise directory, through its AAPML specification, makes general "telephone book" type information available to all internal applications. In this case, the user-id processed by JAAS will be used to retrieve the email address and employee number for the user - enabling the call to HR to fetch the management and direct report information.

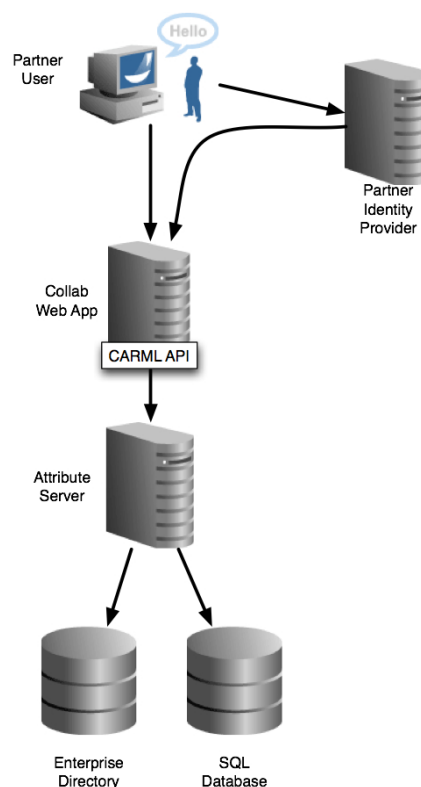
**Legacy Note: To integrate JAAS itself, it is quite possible that a CARML provider be used to allow JAAS/AuthZ/AuthN to function using Identity Attribute Services. This raises the issue of IGF's role in authentication. As with LDAP, we likely expect that IGF is merely a system for accessing authentication attributes and is not itself an authenticator.**

### 4.1.3 Partner Users Access Corporate Resources via SAML

[Source: Oracle IGF]

An engineering services company is providing contract employees for a client. In this case, contractor employees need to be able to access the engineering collaboration applications. To achieve this, SAML federation is being used along with CARML to retrieve local RBAC information important to the engineering projects.

In this scenario, the partner engineer user goes to the collaboration web application and is challenged to authenticate. The user is re-directed to their employer's identity provider server where they authenticate and a SAML assertion is generated. The SAML assertion includes confirmation of authentication and includes some contract information to indicate to the collaboration web application on which contracts the employee is working.



**Figure 11**

The web application receives the SAML token and makes a call into the CARML API to retrieve all of the information about the SAML-asserted identity. In this case, the SAML assertion provides the attribute server with a way to identify the user and provides the contract information included in the assertion. The contract information is then used by the attribute service to look up local RBAC information found in the SQL database.

#### 4.1.4 Corporate Travel Booking Site (multi-mode)

[Source: Oracle IGF]

##### 4.1.4.1 Introduction

The diagram below shows an example use case of a corporate travel booking service. In this scenario, both user-centric identity, in the form of CardSpace (using WS-Trust), and Identity Governance Framework are used. Additionally, a web policy server is used to control access rights and authentication for the web application.

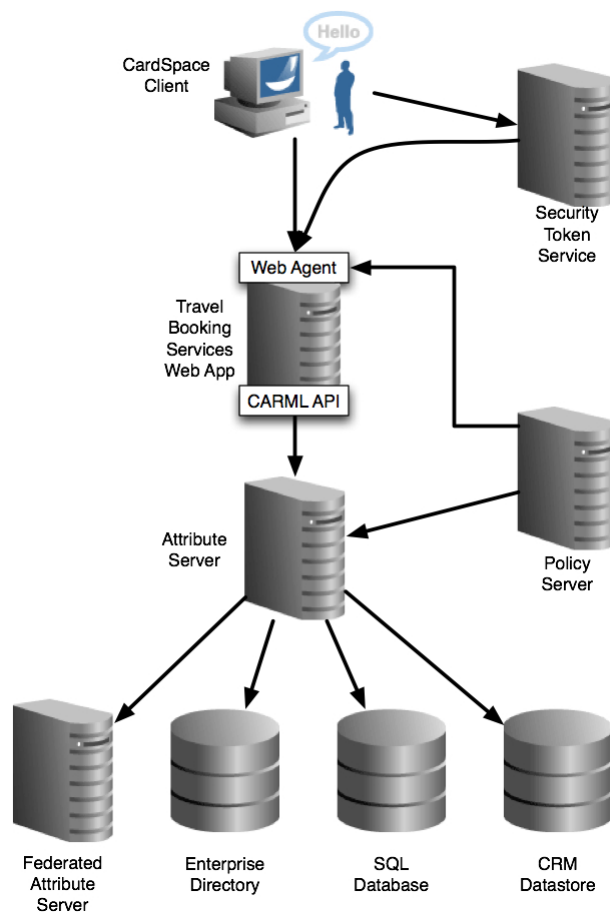


Figure 12

In this scenario, an employee using an identity selector client goes to the corporate travel web site which presents the user with a requirement to provide claims assertions for their "employee" status. The web application's security policy specifies the user is to provide an authentication assertion and some basic information (claims) regarding the employment status and management level in order to allow the travel application to apply the correct travel policy. The travel site will use employment status and management level to determine if the user can book business-class tickets. To complete this process, the user's identity selector activates and begins the process of obtaining a signed token that includes the requested claims from their employer's approved identity providers. The identity provider authenticates the user and provides a signed assertion with the information (claims) requested.

#### 4.1.4.2 Security Token Service and IGF

The security token service at the employer site itself is an IGF-enabled service. To obtain the information necessary, the following must happen:

1. The user is authenticated either through a typical authentication challenge or by accepting a claim provided by the user.
2. The information required by the travel booking site application must be gathered to construct the appropriate claims token. In this case, the employee's management level must be obtained. In obtaining this information, certain AAPML policy would be evaluated to determine if this action is appropriate. The policy evaluates if user consent has been obtained and if the information can be propagated to the Travel Booking Service.
3. If consent is obtained, the management level is added to the claims. If consent is not obtained the value is not included.
4. The STS returns to the employee a signed assertion that they are an employee and depending on consent, includes other claims such as management level.

#### 4.1.4.3 Travel Web Site Processes Federated Token

After obtaining the correct token, the user's browser presents the token to the Travel Booking Web site. The web site's web policy server (e.g., Oracle Access Manager) intercepts the request, evaluates the token received, and applies any web site access rules as needed. The data from the token now forms part of the data available to the application.

**Note: The Web Policy server and database security servers could all be consumers of identity-related data and therefore could all use data supplied by the IGF Attribute Service. In the diagram above, the web policy server is shown obtaining information from the Attribute Server. In this scenario, both the web application and the web policy servers are clients of the attribute service. Note that these two different applications will have different requirements of the attribute service. The**

**web policy server reviews authentication and RBAC-related information, while the application uses application-specific data such as management level, travel preferences, and travel memberships.**

After the web policy server web agent passes control back to the web application, the web application now understands who the user is and what their corporate affiliation and management level is. The travel site needs to locate additional information such as:

- The user's previously registered travel preferences and airline memberships that are stored in a SQL database.
- Information from a customer relationship management system in order to provide integrated telephone support.
- Airline data such as verifying frequent-flier membership status information, e.g., is the user entitled to a premium seat as a frequent flier?

#### **4.1.4.4 Multiple User Populations**

In addition to supporting "traveling" users, the application may also need to support internal employees who use the service to provide customer support through the web application. The Travel Corporation's own enterprise directory is used to obtain employee role-based access control information to determine what actions employees may perform in connection with travelers.

#### **4.1.4.5 Travel App Access Consolidates Identity Information**

The IGF Attribute Server integrates all of the required sources together into a combined "view" for the application. When the policy server passes control to the web application, it may then obtain the desired user attributes by calling the CARML client APIs and requesting the declared CARML attributes for the user previously validated by the policy server. To comply with the request, the attribute server pulls the previously defined information from the CRM server and SQL database.

#### **4.1.4.6 Back-channel Circle-of-Trust Federated Attribute Request**

In the case of the airlines, the attribute service asks if the customer (frequent flier) identified is eligible for premium seat selection. In this case, the information requested is a confirmation of status (a true/false condition) and does not actually ask for their frequent flier classification. This not only improves confidentiality by minimizing the information but it also helps the travel app avoid issues such as how to interpret different frequent flier membership levels from different airlines. In processing this request, the airline service reviews their own consent information and determines whether the travel booking company is authorized to obtain this information on behalf of this client.

#### 4.1.4.7 Consent Handling In Multiple Domains

In this example, there are potentially three or more separate identity attribute servers where policy, consent, and mapping may take place:

- The employer's security token service,
- The travel agent's identity attribute service, and
- The federated airline frequent flier service.

At each of these locations, separate policy is enforced within a specific organizational domain. Additionally, for each event, each domain must also evaluate the transaction and determine if user consent is required for the transaction to proceed. When user-consent is not validated, then the appropriate failure action must be determined such as refuse the entire request, simply filter the specific attribute or property requested, or default the attribute or property.

When information is returned, consent rules must also be interpreted in order to apply transactional conditions so that proper usage and consent information may be propagated. For example, an employee may have consented with the airline (or frequent flier program) to share information with the travel agency. However, when returning the information to the agency, the airline attribute service may need to indicate that the information may not be propagated further. This depends on the airline's overall agreement with the traveler and whether or not specific consent is gathered for this type of scenario. What this discussion is intended to highlight is that it is important to note that consent records themselves must be mapped interpreted when information is propagated between services in order to meet the terms of overall legal terms between the traveler, the airline, and its partners.

Finally, in terms of propagation of information, it might be reasonable to expect that, while consent was granted by the traveler to share information with partners, the airline itself may wish to express concerns about how and if the travel agency may retain the information (e.g., for the purposes of caching). The airline may have an interest or concern:

- To track each usage of information,
- To ensure the information is accurate at the time of use (e.g., the traveler may have been "promoted" since the last transaction), and/or
- To minimize queries to optimize performance between its services and its partners.

#### 4.1.4.8 Conclusion

This example had several different types of attribute service clients:

- Security Token Service,
- Web Policy Server,
- Web Application, and

- Attribute Service.

The employer's token server, the travel company's web policy server, the travel web application, and even the travel company's attribute service itself were all clients to identity attribute services. The travel company identity attribute service was a client to a federated airline attribute service for the purpose of confirming frequent flier booking rights which demonstrated a need for chained or recursive queries that can span multiple organizational boundaries. Each system consumed information about the same individual (the user), but the attributes needed and used by each were substantially different to perform their respective functions. The Attribute Service should be able to handle multiple clients because it "understands" the needs of each system (via their CARML declarations) and uses mapping and routing to support the different needs of each application and infrastructure component.