OASIS Service Provisioning Markup Language (SPML) v2 – Standard Schema: SIMPLEST

Draft 0.04
26 March 2007

Document identifier: draft-pstc-spml2-standard-schema-04.doc

Location: http://www.oasis-open.org/committees/provision/docs/

Send comments to: pstc-comment@lists.oasis-open.org

Editor:

Gary Cole, Sun Microsystems (Gary.P.Cole@Sun.com)

Contributors:

Doron Cohen, BMC
Gary Cole, Sun Microsystems
Rami Elron, BMC
Ron Jacobsen, CA
Martin Raepple, SAP
Richard Sand, Tripod Technology Group
Gavenraj Sodhi, CA
Kent Spaulding, Sun Microsystems
Alex Wang, Tripod Technology Group

Abstract:

This specification defines a standard schema for SPMLv2 in order to promote interoperability. This schema defines *operational attributes* that are functionally equivalent to standard capabilities defined by SPMLv2 specification. This schema also defines *object classes* that represent entities common to the identity management domain, as well as *naming attributes* that identify instances of those entities and *relationship attributes* that represent relationships between instances of those entitities.

Status:

This draft is a work product for the PSTC Standard Schema Work Group.

If you are on the provision list for committee members, send comments there. If you are not on that list, subscribe to the provision-comment@lists.oasis-open.org list and send

comments there. To subscribe, send an email message to provision-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

Table of contents

1. Introduction ... 3
1.1. Benefits of a Standard Schema ... 3
1.2. Organization of this Document ... 4
1.3. Terminology ... 4
1.4. Notation ... 4
2. Domain Model for Identity Management ... 6
2.1. Person, Account and Service ... 7
2.2. Organization, Group and Role ... 8
2.3. AccountTemplate and AccountAttribute ... 9
2.4. SIMPLEST Relationships ... 11
3. SIMPLEST Schema ... 14
3.1. Person ... 14
3.2. Account ... 17
3.3. Service ... 20
3.4. Organization ... 21
3.5. Group ... 23
3.6. Role ... 24
3.7. AccountTemplate ... 26
3.8. AccountAttribute ... 27
3.9. Answer ... 28
3.10. Question ... 29
4. Conformance (normative) ... 30
Appendix A. References ... 31
Appendix B. Acknowledgments ... 32
Appendix C. Revision history ... 33
Appendix D. Notices ... 34

1. Introduction

This document defines a “Standard Identity Management Protocol Lightweight Extensible Schema Template” (SIMPLEST) schema for SPML2 **[SPML2]**.

The SPMLv2 protocol is flexible and rich in function. It is also complex. SPMLv2 operations support arbitrary XML payloads for managed objects. A provider may expose multiple targets, of which each target specifies a schema. SPML2 also defines an extensible set of optional capabilities, of which each capability may imply additional operations or semantics.

SIMPLEST defines a set of object-classes that represent entities that commonly occur within the functional domain of identity management. For each object class, SIMPLEST defines attributes that represent features, functions and relationships typical of (instances of) these entities.

Note that SIMPLEST is not a *profile* of SPML. A profile specifies the manner in which a requester and a provider agree to use the SPML2 protocol. For example, a profile may specify a particular schema language, identifier format and query language. SIMPLEST is a schema that can be used in combination with (new or existing) profiles of SPMLv2.

1.1. Benefits of a Standard Schema

SIMPLEST defines *object classes* that represent entities common to the identity management domain. SIMPLEST also defines several types of attributes:

- *naming attributes* that identify instances of any object-class;
- *relationship attributes* that allow instances to refer to other instances;.
- *data attributes* that represent typical features of (instances of) each object-class;
- *operational attributes* that direct a provider to perform some action.

Simplicity. SIMPLEST defines operational attributes that are functionally equivalent to the Password Capability and Suspend Capability of SPMLv2. SIMPLEST relationship attributes represent common types of associations between objects that would otherwise require the Reference Capability. Requesters and providers that use SIMPLEST need only the core operations of SPMLv2 in order to perform operations that would ordinarily require optional capabilities.

Power. SIMPLEST operational attributes allow a single request to perform (what would otherwise require) several capability-specific operations. For example, an Add Request that uses SIMPLEST can 1) *create* a Person 2) in a *disabled* state 3) setting a *future enablement date* 4) and setting a *password* value 5) where that password value is *pre-expired*. Combining operations optimizes protocol flow. Moreover, where a provider must perform business processes that apply only to enabled objects, creating an object in a disabled state may save a significant amount of work (and work to undo earlier work) on the part of the provider.

Granularity. SIMPLEST operational attributes allow a provider to declare support for a subset of the (functions of the) equivalent capability. For example, a provider that wishes to support only immediate suspend and resume (but neither suspend-on-a-specific-date nor resume-on-a-specific-date) may declare in that provider's target schema the attribute “Is-Disabled” (and omit the attributes “Disable-Date” and “Enable-Date”).

Interoperability. Adopting a canonical schema simplifies the mapping that requesters and providers must perform. Where each requester must otherwise map its proprietary schema to the proprietary schema of each provider that it uses (which is a problem on the order of “M-times-N”), SIMPLEST allows each requester or provider to map its proprietary schema to a single canonical schema (which is a much simpler problem on the order of “M-plus-N”).

1.2. Organization of this Document

This document presents a Domain Model for Identity Management. The domain model describes entities and relationships common in identity management.

This document next presents a Schema. Each entity from the domain model is represented as an object-class. For each object-class, the Schema defines attributes that represent features, functions, and relationships typical of (instances of) the corresponding entity.

Finally, this document illustrates using the SIMPLEST schema with Directory Service Markup Language (DSML) as specified in the DSML profile of SPMLv2 **[SPML-DSML]**. The SIMPLEST schema could also be used with SAML attribute value assertions. The SIMPLEST schema could also be used with XML Schema as specified in the XML Schema Profile of SPMLv2 **[SPML-XSD]**.

1.3. Terminology

Within this document:
- The term “requester” always refers to a Requesting Authority (RA).
- The term “provider” always refers to a Provisioning Service Provider (PSP).
- The term “target” always refers to a Provisioning Service Target (PST).
- The term “object” (unless otherwise qualified) refers to a Provisioning Service Object (PSO).
- The term “client” (unless otherwise qualified) refers to a Requesting Authority (RA).
- The term “server” (unless otherwise qualified) refers to a Provisioning Service Provider (PSP).

1.4. Notation

This specification contains schema conforming to W3C XML Schema and normative text to describe the syntax and semantics of XML-encoded policy statements.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF RFC 2119 **[RFC2119]**

> *"they MUST only be used where it is actually required for interoperation or to limit behavior which has potential for causing harm (e.g., limiting retransmissions)"*

These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

This specification uses the following typographical conventions in text:

Format	Description	Indicates
`attributeName`	monospace font with *first letter lower-cased*	The name of an XML *attribute*.
`SPMLElementName`	monospace font with *first letter capitalized*	The name of an XML *element* that is defined as part of SPMLv2.
`ns:ForeignElementName`	monospace font *with namespace prefix*	The name of an XML element that is *defined by another specification.*
`<SPMLElement>`	monospace font *surrounded by* <>	*An instance of an XML element* that is defined as part of SPMLv2.

`<ns:ForeignElement>`	`monospace font` *`with namespace prefix`* `surrounded by <>`	An instance of an XML element that is *defined by another specification*.

Terms in ***italic bold-face*** are intended to have the meaning defined in the Glossary.

```
Listings of SPML schemas appear like this.
```

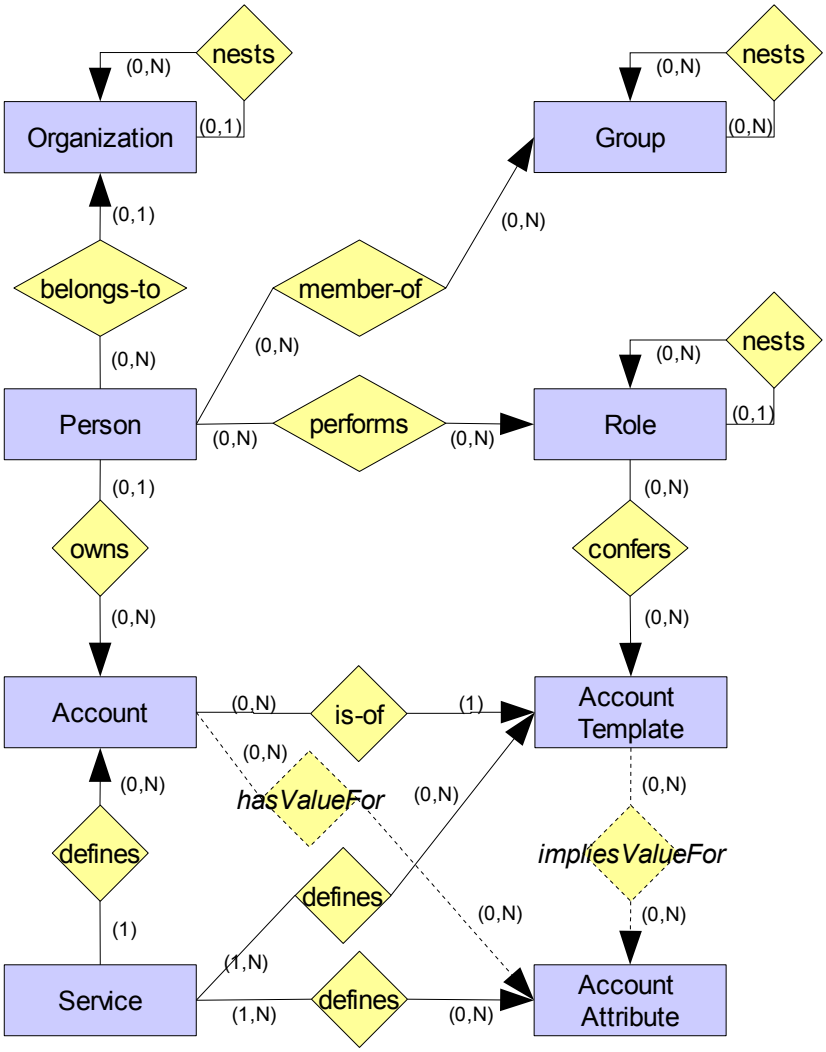
```
Example code listings appear like this.
```

Conventional XML namespace prefixes are used throughout the listings in this specification to stand for their respective namespaces as follows, whether or not a namespace declaration is present in the example:

- The prefix `saml:` stands for the SAML assertion namespace [**SAML**].
- The prefix `ds:` stands for the W3C XML Signature namespace [**DS**].
- The prefix `xsd:` stands for the W3C XML Schema namespace [**XS**].

2. Domain Model for Identity Management

147 This section introduces entities and relationships common to the domain of identity management.

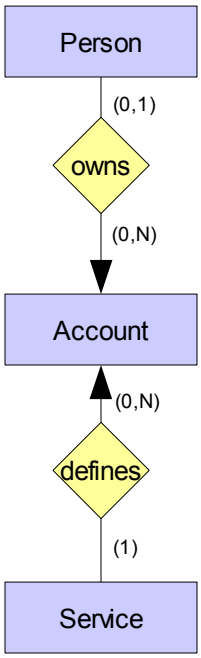


149 Each of the following subsections presents a subset of the domain model, beginning with the most
150 familiar:

151 - The first subsection below presents Person, Account and Service.
152 - The next subsection below presents Organization, Group and Role.
153 - A third subsection below presents AccountTemplate and AccountAttribute.

154 A final subsection entitled "SIMPLEST Relationships" discusses how the SIMPLEST Schema uses
155 object classes and attributes to represent these entities and relationships.

2.1. Person, Account and Service

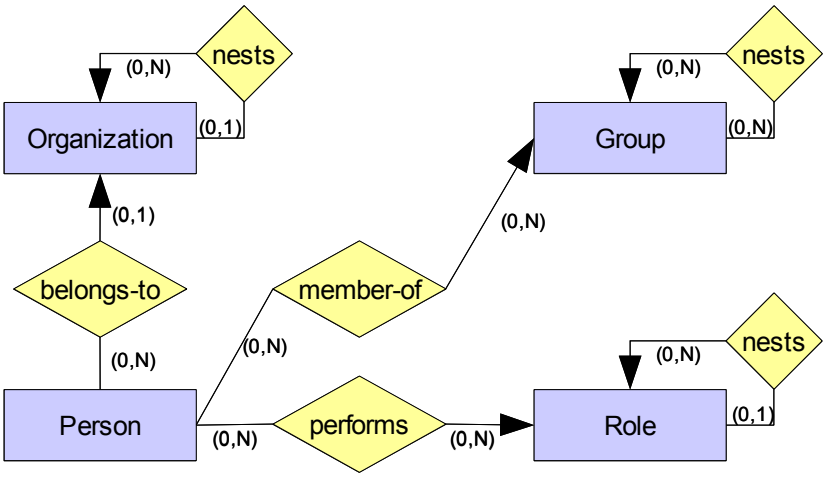


The Person and Account schema entities are fundamental to Identity Management. An instance of Person normally represents a human being. An instance of Account normally represents a person *within the scope of a particular computer system or application*. **Each person may *own* (that is, may be responsible for) any number of accounts. At most one person may own each account.**

A Service is a computer system or application that defines accounts. **A service may define any number of accounts. Exactly one service defines each account.**

The concept of a Service is closely related to SPML's concept of a Target. A Service is a *physical endpoint* for provisioning, whereas a Target is a *logical endpoint* for provisioning that a provider exposes to requesters. An SPML provider may expose a service as a target. On the other hand, rather than expose an actual service, an SPML provider may expose as a target an abstract collection of services or (may expose as a target) a functional description that is more like a role. In short: **A service may be a target, but a target is not necessarily a service**.

2.2. Organization, Group and Role



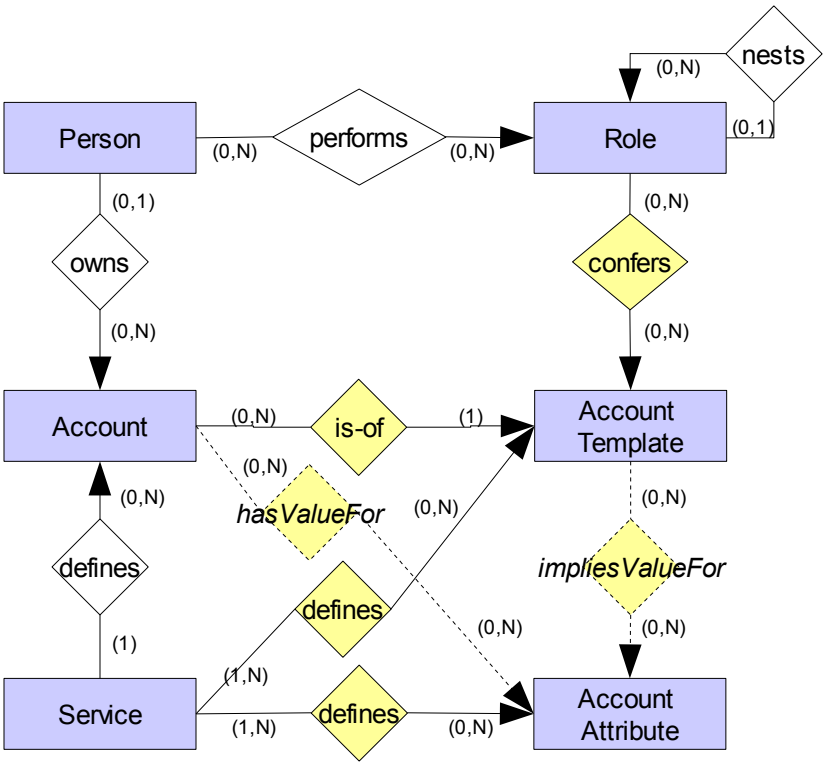
The Organization schema entity is ubiquitous in directory services (and therefore is common in identity management systems). An instance of Organization usually represents the management structure of a corporate entity—that is, an entity that consists of more than one person. The most common management structure is a hierarchy: **Each organization may nest any number of organizations. Exactly one organization nests each organization (except the topmost, which none nests).**

Persons are "leaf" nodes in an organizational hierarchy. **Each person may belong to at most one organization. Any number of persons may belong to each organization.**

The Group schema entity usually represents an arbitrary collection of persons. (A group need not contain persons, but typically does.) **Each person may be a member of any number of groups. Any number of persons may be a member of each group.** Classically (as derived from Unix groups) a group cannot contain other groups, but many modern systems and applications allow this. Many modern groups may form hierarchies—or may form structures more flexible than hierarchies. **Each group may contain any number of groups. Any number of groups may contain each group.** Whoever contains groups is responsible for preventing cycles—that is, a group must not contain itself directly or indirectly. The most important difference between Group and Organization or Role is *semantic*: Group membership is assumed to be orthogonal to (that is, a dimension independent of) both organizational hierarchy and job function.

The Role schema entity represents a job function that a person may perform. Like group membership, role membership is not exclusive. **Each person may perform any number of roles. Any number of persons may perform each role.** Like organizations, roles may be nested to form a hierarchy. **Each role may nest any number of roles. At most one role may nest each role.** However, role is assumed to be *orthogonal to organization*. That is, a role hierarchy represents (a taxonomy of job function that is) a dimension independent of management hierarchy. The semantic difference between Group and Role is that group membership is generally "shallow"--that is, group membership entails little or no data beyond the fact of membership. Role membership is usually "deeper": a role may confer specific types of access to specific services. The section entitled "AccountTemplate and AccountAttribute" discusses this further.

2.3. AccountTemplate and AccountAttribute



This section describes the entities and relationships that are not well-formalized in the industry. Nonetheless, almost every commercial identity management system has some notion of the schema (that is, a defined set of attributes) for accounts on a service. Furthermore, any identity management system that allows a person to own multiple accounts on a single service, and that allows a role to specify (that a person who performs the role should own an account on) a particular service, must have some notion of different types of accounts. A note at the end of this section discusses this in more detail.

A service may define more than one type of account. (That is, the identity management system may define specific account templates that are available on a service.) Each AccountTemplate represents a named type of account. For example, the “default” AccountTemplate may imply only basic or standard access to that service, whereas an “administrator” AccountTemplate may imply additional access. (The underlying system or application that the Service represents may not define specific categories of account, or may define categories that differ from those that the identity management system chooses to expose.) **Each service may define any number of account templates. At least one service must define each account template.**

A service may define a set of account attributes. Each AccountAttribute represents a managed characteristic of accounts on that Service. The identity management system models these attributes explicitly—e.g., in order to enable special policy or control. The identity management system may map these attributes to native—i.e., service-specific—characteristics of an account . (Accounts on that service may have additional characteristics that are not managed, or that are not modeled explicitly.) **Each service may define any number of account attributes. At least one service must define each account attribute.**

A role often confers some type of account. (That is, each job function that is modeled as a role often requires that the person be granted some level of access—or some specific *type* of access—to a particular service.) In the simplest case, a role specifies that any person who performs the role should have at least basic access to a service. That unqualified assignment of access to a service —the "default" account template—confers a normal or standard account for that service. In some cases, however, a role may confer a specific type of account—for example, an "administrator" account. **Each role may confer any number of account templates. Any number of roles may confer each account template.**

Each account template (for example, an "administrator" account template) may imply a set of values for (each of any number of) attributes that grant additional access on the service. (An "administrator" account template might be allowed to affect resources that are not available to other accounts, might be allowed to affect resources that are owned by other accounts, or might be allowed to change the characteristics of other accounts.) **Each account template may imply values for any number of account attributes. Any number of account templates may imply values for each account attribute.**

Every account is based on some account template. By default, an account is an instance of the default account template for the service that defines the account. (If a Person owns a particular account because the person performs a role that confers a specific account template, then the account must reflect its account template in order to maintain the association with the role. Otherwise, it may not be clear which accounts a Person should keep when that Person's roles change. See the note below at the end of this section.) **Every account is based on exactly account template. Any number of accounts may be based on each account type.**

NOTE: Identity management systems differ in the extent to which each supports Role-Based Access Control and (identity management systems also differ) in the manner in which each supports it. However, the fact that a role implies a specific type of account for a service (rather than conferring privileges onto whatever accounts for that service that person owns) becomes clear when a role (or when the set of roles that a particular person performs) implies more than one type of account for the same service. This is especially clear when a person must use each type of account for a distinct purpose.

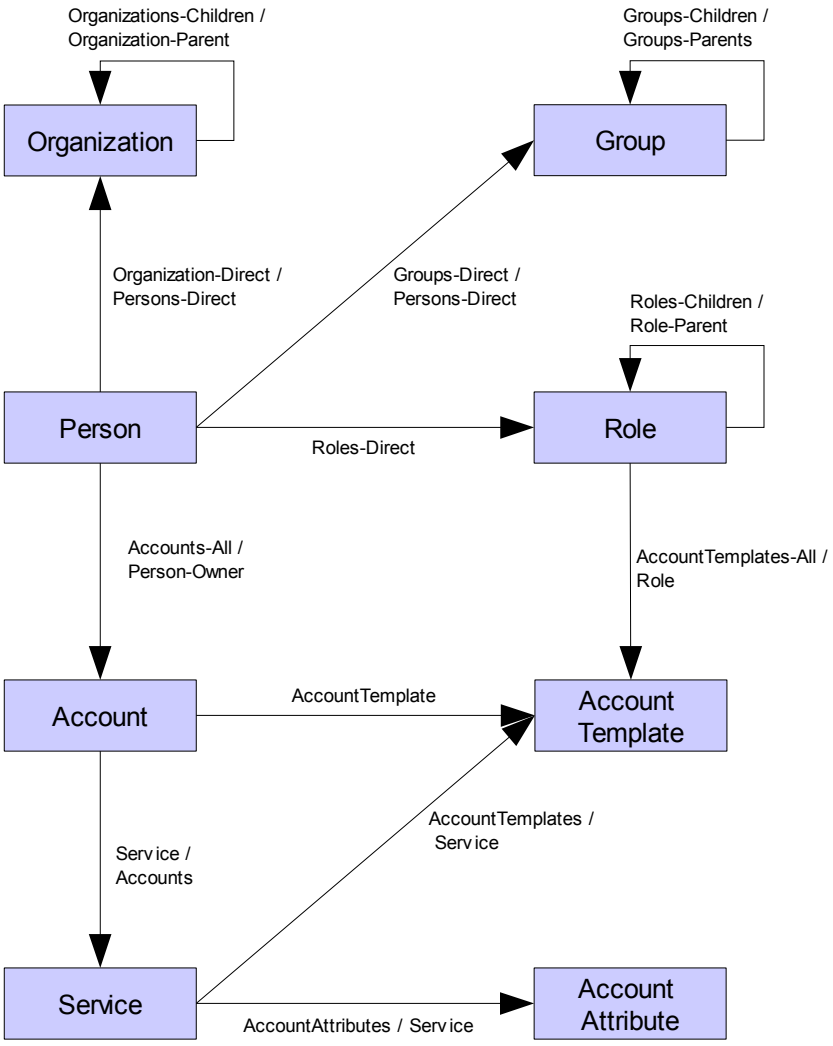
Imagine the following situation:

- An "HRUser" role implies a normal "user" account on the "HR" service.
- An "HRAdministrator" role implies a special "administrator" account on the "HR" service.
- A person who has both roles—and who is therefore both an administrator and a user—must use the special "administrator" account to perform all administrative functions and must user the normal "user" account to perform all "end-user" functions. This enables the company to keep a clean audit log of who did what when—and in what capacity.

If the person gains a "GlobalAdmin" role that also implies a special "administrator" account on the "HR" service, then there should be no net change (even if that person subsequently loses the "HRAdministrator" role). If the person loses both the "HRAdministrator" role and the "GlobalAdmin" role, that person should lose the special "administrator" account on the "HR" service but that person should keep the normal "user" account.

2.4. SIMPLEST Relationships

SIMPLEST defines an *object class* to represent each of the schema entities in the domain model for identity management. SIMPLEST defines (for each of these object classes) *attributes* that represent relationships between (instances of) these object classes. Reworking the domain model to show relationships in terms of attributes yields the following diagram.



Person, Account and Service.

SIMPLEST defines Person, Account and Service as object classes. SIMPLEST uses attributes of these object classes to represent relationships between (instances of) Person and Account. An instance of Person may expose an “Accounts-All” attribute. The “Accounts-All” attribute may have multiple values. Each value of the “Accounts-All” attribute identifies an instance of Account for which the person is responsible. SIMPLEST also represents the inverse relationship: an instance of Account may expose an “Person-Owner” attribute. The “Person-Owner” attribute may have at most one value. Any value of the “Person-Owner” attribute identifies the (instance of Person that represents the) person who is responsible for the account.

NOTE: Many identity management systems conflate (that is, do not distinguish between) Person and Account. The SIMPLEST schema distinguishes between Person (an identity independent of any system or application) and Account (an identity in the context of a specific system or application). An SPML requester or provider that uses the SIMPLEST schema SHOULD clearly distinguish clearly between Person and Account.

SIMPLEST similarly uses attributes to represent relationships between (instances of) Account and Service. An instance of Account always has a “Service” attribute that contains a single value. The value of the “Service” attribute identifies the (Service object that represents the) system or application that defines the account.

NOTE: SIMPLEST could expose an “Accounts” attribute on the Service object-class that would allow a service to refer to every account that it defines. However, this would scale poorly because an “Accounts” attribute may have a very large number of values.

Organization, Group and Role.

SIMPLEST represents the hierarchical nesting of organizations using the “Organization-Parent” and “Organizations-Children” attributes of Organization. SIMPLEST allows an instance of Person to refer to an instance of Organization using the “ou” attribute (A.K.A. “Organization-Direct”).

NOTE: SIMPLEST Organization could also expose a “Persons-Direct” attribute that would allow an organization to refer to each person that the organization contains. However, this approach tends to scale poorly because a “Persons-Direct” attribute may have a large number of values. This approach also introduces a requirement to synchronize the “Persons-Direct” attribute with any inverse attribute such as the “Organization-Direct” attribute of the Person object class. It is usually better simply to have each instance of Person refer to an instance of Organization.

SIMPLEST allows group nesting using the “Groups-Parents” and “Groups-Children” attributes of Group. SIMPLEST allows a person to refer to any number of groups by means of the “Groups-Direct” attribute of Person. This approach scales better than having a Group refer to each of its members—see the discussion of “Persons-Direct” above in this section.

SIMPLEST allows a role nesting using the “Role-Parent” and “Roles-Children” attributes of Role. The “Roles-Direct” attribute of Person allows a person to refer to any number of roles. This approach scales better than having a Role refer to each of its members—see the discussion of “Persons-Direct” above in this section.

NOTE: Group and Role are sometimes conflated--much as Person and Account are sometimes conflated. SIMPLEST therefore defines the Group and Role schema entities with many of the same attributes. Nonetheless, an SPML requester or provider that uses the SIMPLEST schema SHOULD clearly distinguish clearly between Group and Role.

AccountTemplate and AccountAttribute.

SIMPLEST defines AccountAttribute as an object class. Each Service may expose a multi-valued attribute called “AccountAttributes”. Each value of “AccountAttributes” identifies an instance of AccountAttribute that the Service defines. Each AccountAttribute defines a managed characteristic of accounts on that service.

NOTE: An instance of Account may have attributes that correspond to instances of AccountAttribute, but this relationship is *implicit*—no attribute represents a relationship between Account and AccountAttribute. Instead, an instance of account may simply have attributes that correspond by name to AccountAttributes that the Service defines.

SIMPLEST also defines a “AccountTemplate” as an object class. Each Service may expose a multi-valued attribute called “AccountTemplates”. Each value of “AccountTemplates” identifies an instance of AccountTemplate that the Service defines. Each instance of AccountTemplate defines a category of account on that service. Each instance of Account has a single-valued “AccountTemplate” attribute that identifies the type of the account.

NOTE: An instance of AccountTemplate may imply a set of values for an AccountAttribute, but this relationship is *implicit*—no attribute represents a relationship between AccountTemplate and AccountAttribute. Instead, an instance of AccountTemplate may simply have attributes that correspond by name to AccountAttributes that the Service defines.

3. SIMPLEST Schema

This section defines a Standard Identity Management Multi-Purpose Lightweight Extensible Schema Template (SIMPLEST). SIMPLEST assumes that each object is represented as a collection of attributes. SIMPLEST specifies a standard set of attributes to use in representing several classes of object that are essential to the domain of identity management: Person, Account, Organization, Group and Role. The specified attributes describe common features of (and common relationships between) such objects. Identity Management commonly involves manipulating these features and relationships.

For a non-normative overview, see the section entitled "Schema Entities and Relationships" earlier within this document.

Extensibility. We call this an "Extensible Schema Template" because neither the standard set of attributes nor the standard set of object-classes is exhaustive. Requester and provider may use additional attributes, but those attributes must not conflict with (and should not bypass) the standard attributes defined here. Requester and provider may use additional object-classes, but those object-classes must not conflict with (and should not bypass) the standard object-classes defined here. Please see the section entitled "Conformance (normative)".

Naming attributes. Naming attributes include "Name" and "GUID", both of which are required, as well as "CN" and "DN", which are optional.

Naming attributes apply regardless of object-class. Every object, even an instance of an extended object-class, should have a "Name" attribute and a "GUID" attribute. An instance of any object-class that supports Common Name and Distinguished Name should have both "CN" and "DN" attributes.

Operational attributes. Operational attributes may also apply regardless of object-class. SIMPLEST specifies (the full set of) password-related attributes only for the Account object-class. However, if a provider intends to support the Password Capability for another object-class, then that object-class should declare the full set of password-related attributes (i.e., "Password", "Password-Is-Expired", "Password-Expire-Date", "Password-To-Validate", "Password-Is-Reset"). A provider that wishes to support only a subset of the operations within the Password Capability may declare in that provider's target schema only the attributes that correspond to that subset.

3.1. Person

An instance of the "Person" class normally represents a human being independent of any association with a particular computer system or application. (In some cases, an instance of "Person" may represent a pseudo-user or an entity other than an actual human being.)

The following attributes are defined for the Person object class. (The case of attribute names is not significant.)

AttributeName	Required/ Optional	Syntax	Multi-Valued?	Description
Name	R	string	SV	A friendly and mutable identifier for this Person.
GUID	R	string	SV	A globally unique and immutable identifier for this Person.

Organization-Direct	O	string	SV	Identifies the Organization to which this Person belongs primarily (i.e., reflects the position of this Person within the management hierarchy.)
Organizations-Indirect	O	string	MV	Each value identifies an Organization with which this Person is associated indirectly (e.g., because the organization contains an organization that contains this Person).
Organizations-Dynamic	O	string	MV	Each value identifies an Organization with which this Person is associated dynamically (e.g., because the organization's membership rules match this Person).
Organizations-All	O	string	MV	Each value identifies an Organization with which this Person is associated, whether directly or indirectly or dynamically. (This attribute is the union of “Organization-Direct”, “Organizations-Indirect” and “Organizations-Dynamic”.)
Groups-Direct	O	string	MV	Each value identifies a Group with which this Person is associated directly. (For comparison, see the “Groups-All” attribute.)
Groups-Indirect	O	string	MV	Each value identifies a Group with which this Person is associated indirectly (e.g., because the group contains a group that contains this Person).
Groups-Dynamic	O	string	MV	Each value identifies a Group with which this Person is associated dynamically (e.g., because the group's membership rules match this Person).
Groups-All	O	string	MV	Each value identifies a Group with which this Person is associated, whether directly or indirectly or dynamically. (This attribute should be the union of “Groups-Direct”, “Groups-Indirect” and “Groups-Dynamic”.)
Roles-Direct	O	string	MV	Each value identifies a Role that is assigned explicitly to this Person. (For comparison, see the “Roles-All” attribute.)
Roles-Indirect	O	string	MV	Each value identifies a Role that is assigned to this Person indirectly (e.g., because the role contains a role that is assigned to this Person).

Roles-Dynamic	O	string	MV	Each value identifies a Role that is assigned to this Person dynamically (e.g., because the role's membership rules match this Person).
Roles-All	O	string	MV	Each value identifies a Role that is assigned to this Person, whether directly or indirectly or dynamically. (This attribute should be the union of “Roles-Direct”, “Roles-Indirect” and “Roles-Dynamic”.)
Accounts-All	O	string	MV	Each value identifies an Account for which this Person is responsible. (Inverse of Account:Person-Owner.)
Accounts-Direct	O	string	MV	Each value identifies an Account that this Person owns because an Administrator directly assigned the Account.
Accounts-Via-Roles	O	string	MV	Each value identifies an Account that this Person owns because a Role implied the Account.
Services-All	O	string	MV	Each value identifies a Service on which this Person owns an Account.
Services-Direct	O	string	MV	Each value identifies a Service on which this Person owns an Account because an Administrator directly assigned the Account.
Services-Via-Roles	O	string	MV	Each value identifies a Service on which this Person owns an Account because a Role implied the Account.
Email-Address	O	string	MV	Electronic mail address for this Person or Account.
Login-ID	O	string	SV	A short name (usually eight characters or fewer) that is used to access a computer system or application.

Password	O	string	SV	A token that is used (along with a logonID) to access a computer system or application. Set to a value in order to specify a new password for this Person. Expect an error if the Provider cannot set this value as the password. Any value that is specified for the Password attribute on a Person becomes the default value of the Password attribute for any Account that this Person owns. See the “Password” attribute of Account.
Is-Disabled	O	boolean	SV	If read as "true", indicates that all access by this Person is prevented. Set "true" in order to prevent all access by this Person or Account. Set "false" in order to allow all access by this Person or Account.
Disable-Date	O	dateTime	SV	Date and time at which all access by this Person is scheduled to be prevented. Set a new value (in UTC format with no timezone) in order to change the date and time at which all access by this Person or Account is scheduled to be disabled. Set to an empty value in order to remove the scheduled disablement.
Enable-Date	O	dateTime	SV	Date and time at which all access by this Person is scheduled to be allowed. Set a new value (in UTC format with no timezone) in order to change the date and time at which all access by this Person or Account is scheduled to be allowed. Set to an empty value in order to remove the scheduled enablement.

360

3.2. Account

361 An instance of the "Account" class represents the identity of a Person *within the scope of a specific*
362 *computer system or application*.

363 The following attributes are defined for the Account object class. (The case of attribute names is
364 not significant.)

AttributeName	Required/ Optional	Syntax	Multi-Valued?	Description
Name	R	string	SV	The "friendly" identifier for this object.
GUID	O	string	SV	A globally unique and immutable identifier for the object.
CN	O	string	MV	Any Common Name associated with the object. (Normally matches "name", although CN may have multiple values.)
DN	O	string	SV	Any DistinguishedName associated with this object. Normally unique within a particular service instance.
Email-Address	O	string	MV	Electronic mail address associated with this Account.
Person-Owner	O	string	SV	Identifies the Person who owns this Account (i.e., the Person who uses this Account and who is responsible for this Account). (INVERSE of “Person:Owns-Account”.)
Service	R	string	SV	Identifies the computer system or application that defines this Account.
AccountTemplate	O	string	SV	Identifies any AccountTemplate (that is, any named purpose) for which a Role assigns ownership of this Account (to the Person identified by “Person-Owner”).
Via-Role	O	string	SV	Identifies any Role that (specifies the AccountTemplate that) assigns responsibility for this Account to the Person identified by “Person-Owner”. (NOTE: This convenience attribute saves finding AccountTemplate identified by “AccountTemplate” and getting from that AccountTemplate the attribute named “Role”.)
Login-ID	O	string	SV	A short name (usually eight characters or fewer) that is used to access a computer system or application. Default value is the value of the “Login-ID” attribute for the Person identified by the value of the “Person-Owner” attribute, if any. Otherwise, defaults to the value of the “Name” attribute.

Password	O (write-only)	string	SV	A token that is used (along with a logonID) to access a computer system or application. Not available when the object is read. (Read as base-64-encoded-hash-of-password?) Set to a value in order to specify a new password for this Account. Expect an error if the Provider cannot set this value as the password.
Password-Is-Expired	O	boolean	SV	If read as "true", indicates that the password value is no longer valid. Set "true" in order to mark the current password value as invalid. (Set "false" in order to mark the current password value as valid.)
Password-Expire-Date	O	date-Time	SV	Date and time the current password value is scheduled to expire. Set a new value (in UTC format with no timezone) in order to change the date and time the current password value should expire. Set to an empty value in order to remove the scheduled expiration.
Password-To-Validate	O (write-only)	string	SV	Not available when the object is read. Set to a new value in order to determine whether the new value would be valid as a password for this Account. Expect an error if the password would not be valid.
Password-Is-Reset	O	boolean	SV	If read as "true", indicates that the current password value was generated (and was communicated out-of-band to the Person who owns this Account). Set to "true" in order to replace the current password value with a generated value. (Expect the new password value to be communicated to the Person by other means.) The value will remain "true" until the Person sets a new password, at which time the value will be read as "false".
Is-Disabled	O	boolean	SV	If read as "true", indicates that all access by this Account is currently prevented. Set "true" in order to prevent all access by this Account. Set "false" in order to allow all access by this Account.

Disable-Date	O	date-Time	SV	Date and time at which all access by this Person or Account is scheduled to be prevented. Set a new value (in UTC format with no timezone) in order to change the date and time at which all access by this Person or Account is scheduled to be prevented. Set to an empty value in order to remove the scheduled disablement.
Enable-Date	O	date-Time	SV	Date and time at which all access by this Person or Account is scheduled to be allowed. Set a new value (in UTC format with no timezone) in order to change the date and time at which all access by this Person or Account is scheduled to be allowed. Set to an empty value in order to remove the scheduled enablement.
Is-Locked	O	boolean	SV	If read as 'true', indicates that the Service currently (and temporarily) prevents access by this Account. Set to 'true' in order to prevent temporarily access by this Account. Set to 'false' in order to re-enable all access by this Account (if access was temporarily disabled due to native lockout).
Is-Direct	O	boolean	SV	If read as 'true' then this Account was assigned directly to the Person that “Person-Owner” identifies (and can therefore be deleted directly or unlinked from the owning Person). If read as 'false' then ownership of this Account (by the Person that “Person-Owner” identifies) was implied by a Role (via the AccountTemplate that the value of the “AccountTemplate” attribute identifies)--or by another mechanism--and may not be directly deleted or unlinked from the owning Person.

365

3.3. Service

366 An instance of the “Service” class normally represents a system or an application that defines
367 Accounts.

368 The following attributes are defined for the Service object class. (The case of attribute names is not
369 significant.)

AttributeName	Required/ Optional	Syntax	Multi-Valued?	Description
Name	R	string	SV	The "friendly" identifier for this object.
GUID	R	string	SV	A globally unique and immutable identifier for this object.
Type-Description	O	string	SV	The name or label for this type of this Service (IDM Resource)--e.g., “ActiveDirectory”, “Solaris”, or “RACF”.
Instance-Description	O	string	SV	The name or label for this (instance of) Service —e.g., a specific hostname or application instance.
AccountTemplates	O	string	MV	Each value identifies a AccountTemplate that this Service defines (or a AccountTemplate that is defined for this Service).
Accounts	O	string	MV	Each value identifies an Account that this Service defines. (INVERSE of Account:Service. SCALES poorly.)

370

3.4. Organization

371 An instance of the "Organization" class normally represents a container node in a hierarchical
372 management structure of a corporation. Organization objects can be used to represent companies,
373 departments, geographies, or other organizational constructs.

374 The following attributes are defined for the Organization object class. (The case of attribute names
375 is not significant.)

AttributeName	Required/ Optional	Syntax	Multi-Valued?	Description
Name	R	string	SV	The "friendly" and mutable identifier for this object.
GUID	R	string	SV	A globally unique and immutable identifier for this object.
CN	O	string	MV	Any Common Name associated with the object. (Normally matches "Name", although CN may have multiple values.)
DN	O	string	SV	Any Distinguished Name associated with this object. Normally unique within a particular service instance.
Description	O	string	SV	Text that describes this object.
Organization-Children	O	string	MV	Each value identifies an organization that this organization contains directly. INVERSE of Organizations-Parent.

Organization-Descendants	O	string	MV	Each value identifies an organization that this organization contains (i.e., a child- or sub-org), whether directly or indirectly. INVERSE of Organizations-Ancestors.
Organization-Parent	O	string	SV	Identifies any Organization that directly contains this Organization. INVERSE of Organization-Children.
Organization-Ancestors	O	string	MV	Each value identifies a Organization that contains this Organization (i.e., a parent- or super-org), whether directly or indirectly. INVERSE of Organization-Descendants.
Persons-Direct	O	string	MV	Each value identifies a Person that is assigned directly to this Organization. (INVERSE of Person:Organization-Direct. SCALES poorly.)
Persons-Indirect	O	string	MV	Each value identifies a Person that is assigned to this Organization indirectly—e.g.., belongs to a descendant of this Organization. (INVERSE of Person:Organizations-Indirect. SCALES very poorly.)
Persons-Dynamic	O	string	MV	Each value identifies a Person that is assigned to this Organization dynamically--e.g., because the membership rules for this organization match this Person. (INVERSE of Person:Organizations-Dynamic. SCALES very poorly.)
Persons-All	O	string	MV	Each value identifies a Person that is assigned to this Organization, whether directly or indirectly or dynamically. (This attribute should be the union of “Persons-Direct”, “Persons-Indirect” and “Persons-Dynamic”. SCALES very poorly.)
Email-Address	O	string	SV	The primary electronic-mail address for this Organization.
Telephone-Number	O	string	SV	The primary phone number for this Organization.
Address-Street	O	string	SV	The street that is part of the address for this Organization.
Address-PO-Box	O	string	SV	Any post-office-box address for this Organization.
Address-City	O	string	SV	The city that is part of any address for this Organization.

Address-State-Or-Province	O	string	SV	The state or province that is part of the address for this Organization.
Address-Postal-Code	O	string	SV	The zip code (or other postal code) that is part of the address for this Organization.
Address-Country	O	string	SV	The nation (or territory) that is part of the address for this Organization.
Contact-Persons	O	string	MV	Each value of this attribute identifies (by name or by GUID) a contact Person for this Organization.

376

3.5. Group

377 An instance of the "Group" class normally represents a collection of Persons or Accounts *within the*
378 *scope of a computer system or application*. (In some cases, a Group may contain objects of other
379 classes.)

380 The following attributes are defined for the Group object class. (The case of attribute names is not
381 significant.) The first set of attributes (related to naming) is common to all object classes.

AttributeName	Required/ Optional	Syntax	Multi-Valued?	Description
Name	R	string	SV	The "friendly" and mutable identifier for this object.
GUID	R	string	SV	A globally unique and immutable identifier for the object.
CN	O	string	MV	Any Common Name associated with the object. (Normally matches "name", although CN may have multiple values.)
DN	O	string	SV	Any DistinguishedName associated with this object. Normally unique within a particular service instance.
Description	O	string	SV	Text that describes this object.
Group-Children	O	string	MV	Each value identifies a Group that this Group contains directly. (INVERSE of Group-Parent.)
Group-Descendants	O	string	MV	Each value identifies a Group that this Group contains (i.e., a child or sub-group), whether directly or indirectly. INVERSE of Group-Ancestors.
Group-Parent	O	string	SV	Identifies a Group that contains this Group (i.e., a parent or super-group) directly. INVERSE of Group-Children.

Group-Ancestors	O	string	MV	Each value identifies (by Name or by GUID) a Group that contains this Group (i.e., a parent or super-group), whether directly or indirectly. INVERSE of Group-Descendants.
Persons-Direct	O	string	MV	Each value identifies a Person that is assigned directly to this Group. (INVERSE of Person:Groups-Direct. SCALES poorly.)
Persons-Indirect	O	string	MV	Each value identifies a Person that is assigned to this Group indirectly—e.g.., belongs to a descendant of this Group. (INVERSE of Person:Groups-Indirect. SCALES very poorly.)
Persons-Dynamic	O	string	MV	Each value identifies a Person that is assigned to this Group dynamically--e.g., because the membership rules for this Group match this Person. (INVERSE of Person:Groups-Dynamic. SCALES very poorly.)
Persons-All	O	string	MV	Each value identifies a Person that is assigned to this Group, whether directly or indirectly or dynamically. (This attribute should be the union of “Persons-Direct”, “Persons-Indirect” and “Persons-Dynamic”. SCALES very poorly.)

382

3.6. Role

383 An instance of the "Role" class normally represents a specific set of job functions that one or more
384 Persons may perform.

385 The following attributes are defined for the Role object class. (The case of attribute names is not
386 significant.)

AttributeName	Required/ Optional	Syntax	Multi-Valued?	Description
Name	R	string	SV	The "friendly" and mutable identifier for this object.
GUID	O	string	SV	A globally unique and immutable identifier for the object.
CN	O	string	MV	Any Common Name associated with the object. (Normally matches "name", although CN may have multiple values.)

DN	O	string	SV	Any Distinguished Name associated with this object. Normally unique within a particular service instance.
Description	O	string	SV	Text that describes this object.
Role-Children	O	string	MV	Each value identifies (by name or by GUID) a Role that this Role contains directly. (INVERSE of Role-Parent.)
Role-Descendants	O	string	MV	Each value identifies (by name or by GUID) a Role that this Role contains (i.e., a child or sub-role), whether directly or indirectly. INVERSE of Role-Ancestors.
Role-Parent	O	string	SV	Identifies any Role that contains this Role (i.e., a parent or super-group) directly. INVERSE of Role-Children.
Role-Ancestors	O	string	MV	Each value identifies (by Name or by GUID) a Role that contains this Role (i.e., a parent or super-role), whether directly or indirectly. INVERSE of Role-Descendants.
Persons-Direct	O	string	MV	Each value identifies a Person that is assigned directly to this Role. (INVERSE of Person:Roles-Direct. SCALES poorly.)
Persons-Indirect	O	string	MV	Each value identifies a Person that is assigned to this Role indirectly—e.g.., belongs to a descendant of this Group. (INVERSE of Person:Roles-Indirect. SCALES very poorly.)
Persons-Dynamic	O	string	MV	Each value identifies a Person that is assigned to this Role dynamically--e.g., because the membership rules for this Role match this Person. (INVERSE of Person:Roles-Dynamic. SCALES very poorly.)
Persons-All	O	string	MV	Each value identifies a Person that is assigned to this Role, whether directly or indirectly or dynamically. (This attribute should be the union of “Persons-Direct”, “Persons-Indirect” and “Persons-Dynamic”. SCALES very poorly.)
AccountTemplates-Direct	O	string	MV	Each value identifies an AccountTemplate that this Role implies directly.

AccountTemplates-All	O	string	MV	Each value identifies an AccountTemplate that this Role—or an ancestor of this Role--implies. (SCALES poorly.)
Services-Direct	O	string	MV	Each value identifies a Service that this Role conveys directly--i.e., via a AccountTemplate that this Role implies. (This attribute is a convenience that saves a client the effort of looking up each AccountTemplate that “AccountTemplates-Direct” identifies and then collecting unique values of the “Service” attribute from those instances of AccountTemplate.)
Services-All	O	string	MV	Each value identifies a Service that this Role conveys, whether directly or indirectly--i.e., via a AccountTemplate that an ancestor of this Role implies. (This attribute is a convenience that saves a client the effort of looking up each AccountTemplate that “AccountTemplates-All” identifies and then collecting unique values of the “Service” attribute from those instances of AccountTemplate.)
Roles-Excluded	O	string	MV	Each value identifies a Role that conflicts with this Role--e.g., for purposes of Separation of Duties (SOD). A Person who has this Role must not have any role that “Roles-Excluded” identifies.

3.7. AccountTemplate

An instance of the "AccountTemplate" class represents a specific purpose for which a Role assigns an Account to a Person.

An AccountTemplate may have any number of additional attributes. Each attribute specifies values that the Provider should assign (for the same attribute) to any Account of this AccountTemplate. Syntactically, any attribute is valid. Semantically, however, only an instance of an AccountAttribute (that is defined by the Service AccountTemplate) is sound. The values of each such attribute must conform to the cardinality and syntax of the corresponding AccountAttribute. The provider reserves the right to disallow (and to fail an attempt to set a value for) any attribute that does not meet these requirements.

The following attributes are defined for the AccountTemplate object class. (The case of attribute names is not significant.)

AttributeName	Required/ Optional	Syntax	Multi-Valued?	Description
Name	R	string	SV	The "friendly" and mutable identifier for this object.

GUID	O	string	SV	A globally unique and immutable identifier for the object.
Description	O	string	SV	Text that describes this object.
Service	R	string	SV	Identifies the Service that defines this AccountTemplate (or the Service for which this AccountTemplate is defined).
Role	R	string	MV	Identifies the Role that defines this AccountTemplate. INVERSE of Role:AccountAttributes.
<anyAccountAttribute>	O	<per AccountAttribute>	<per AccountAttribute>	An AccountTemplate may have any number of additional attributes. Each attribute specifies values that the Provider should assign (for the same attribute) to any Account of this AccountTemplate. Syntactically, any attribute is valid. Semantically, however, only an instance of an AccountAttribute (that is defined by the Service AccountTemplate) is sound. The values of each such attribute must conform to the cardinality and syntax of the corresponding AccountAttribute. The provider reserves the right to disallow (and to fail an attempt to set a value for) any attribute that does not meet these requirements.

3.8. AccountAttribute

An instance of the "AccountAttribute" class represents an attribute that a Service defines as valid for accounts on that Service. Each instance of AccountAttribute may map to an attribute in the native schema for the system or application that the Service represents.

The following attributes are defined for the AccountAttribute object class. (The case of attribute names is not significant.)

AttributeName	Required/ Optional	Syntax	Multi-Valued?	Description
Name	R	string	SV	The "friendly" and mutable identifier for this object.
GUID	O	string	SV	A globally unique and immutable identifier for the object.
Description	O	string	SV	Text that describes this object.
Service	R	string	SV	Identifies the Service that defines this AccountTemplate (or the Service for which this AccountTemplate is defined).
Syntax	O	string	SV	Identifies the Syntax of this attribute—e.g., one of String, Binary. If not specified, then the default value is “String”.

Is-Multi-Valued	O	boolean	SV	If true, then this attribute may have multiple values. If false, then this attribute may have at most one value. If not specified, then the default value is false.
Is-Required	O	boolean	SV	If true, then a request to create an instance of Account (on the Service that defines this AccountAttribute) must specify this attribute. If false, then a request (to create an instance of Account on the Service that defines this AccountAttribute) may omit this attribute. If not specified, then the default value is false.

3.9. Answer

An instance of Answer records a Person's response to an authentication question. An Answer may refer to a Question that is common or shared (and to which the Answers of many other Persons may therefore also refer), or an Answer may specify question text that is private (and is therefore specific to one Person).

The following attributes are defined for the Answer object class. (The case of attribute names is not significant.)

AttributeName	Required/ Optional	Syntax	Multi-Valued?	Description
Name	R	string	SV	The "friendly" and mutable identifier for this object.
GUID	R	string	SV	A globally unique and immutable identifier for the object.
Person-Owner	R	string	SV	Identifies the Person who recorded this Answer.
Question-GUID	O (required unless Question-Text is set)	string	SV	Identifies the Question for which the Person (that "Person-Owner" identifies) recorded this Answer. If Question-GUID has a value, then the Question is common or shared (and is therefore defined as a separate Question object), which allows the Answers of any number of Persons to refer to the same question. If Question-GUID has no value, then the question text is specific to the Person (that "Person-Owner" identifies). If Question-GUID has no value, then a request to create an Answer must specify a value for the Question-Text attribute.

Question-Text	O (required unless Question-GUID is set)	string	SV	The text of the authentication question. If the Question-GUID attribute has a value, then the value of the Question-Text attribute will match (the value of the Text attribute of) the Question that Question-GUID identifies. If the Question-GUID attribute has no value, then this text represents a question that is specific to the Person that Person-Owner identifies. If Question-GUID has no value, then a request to create an Answer must specify a value for the Question-Text attribute.
Text	O	string	SV	The text that the Person (that Person-Owner identifies) recorded in response to the question (that Question-GUID identifies or that Question-Text specifies).

3.10. Question

409 An instance of Question represents an authentication question that is common or shared (i.e.,
410 defined as a separate object), and to which the Answers of any number of Persons may therefore
411 refer.

412 The following attributes are defined for the Question object class. (The case of attribute names is
413 not significant.)

AttributeName	Required/ Optional	Syntax	Multi-Valued?	Description
Name	R	string	SV	The "friendly" and mutable identifier for this object.
GUID	R	string	SV	A globally unique and immutable identifier for the object.
Text	O	string	SV	The text of this (shared) question.
Message-Key	O	string	SV	Identifies a message in a catalog. This key allows a requester to localize (the text of) this Question.

414

4. Conformance (normative)

Compliance versus Conformance. For the purposes of this document, “compliance” means avoiding conflict with (the letter of) this specification. “Conformance”, on the other hand, means embracing (the spirit of) this specification. That is, “compliance” refers to behavior that is *required* of requesters and providers, while “conformance” refers to the behavior that is *recommended* for requesters and providers.

To be concrete, a compliant provider obeys all of the MUST and MUST NOT statements in this section. A conformant provider, in addition to obeying all of the MUST and MUST NOT statements, also obeys all of the SHOULD and SHOULD NOT statements.

Declaring support for SIMPLEST. A provider that supports the SIMPLEST schema for a target MUST specify the URN of SIMPLEST (urn:oasis:names:tc:SPML:2.0:SIMPLEST) as the value of the “`ref`” attribute of the `<schema>` element of that `<target>` element within that provider's List Targets Response.

A target that refers to SIMPLEST MUST NOT conflict with SIMPLEST. That is, any target schema that refers to the SIMPLEST schema MUST NOT contain (and the provider must not support for that target) any object-class that conflicts with SIMPLEST. Furthermore, any target schema that refers to the SIMPLEST schema MUST NOT contain (and the provider must not support for that target) any attribute that conflicts with SIMPLEST.

- **No conflicting object-class.** Any object-class (that is declared within a target schema that refers to SIMPLEST and) that has (a name that matches) the name of a SIMPLEST object-class MUST require (an attribute with a name that matches the name of) each attribute that the corresponding SIMPLEST object-class requires.
- **No conflicting attribute**. Any attribute (that is declared within a target schema that refers to SIMPLEST and) that has (a name that matches) the name of a SIMPLEST attribute MUST have the same properties (e.g., syntax, required/optional, single-valued/multi-valued) and the same semantics (i.e., meaning as specified by the description) as the corresponding SIMPLEST attribute.

No target should conflict with SIMPLEST. A target SHOULD NOT conflict with SIMPLEST, whether a target refers to the SIMPLEST schema or not.

- Any object-class that has (a name that matches) the name of a SIMPLEST object-class SHOULD require (an attribute with a name that matches the name of) each attribute that the corresponding SIMPLEST object-class requires.
- Any attribute that has (a name that matches) the name of a SIMPLEST attribute SHOULD have the same properties (i.e., syntax, required/optional, single-valued/multi-valued) and the same semantics (i.e., as specified by the description) as the corresponding SIMPLEST attribute.

Providers may extend SIMPLEST. A provider MAY declare, within a target schema that refers to SIMPLEST, object classes or attributes in addition to those that SIMPLEST defines. These extended object-classes and attributes SHOULD NOT bypass SIMPLEST.

- **Extended object-classes.** A provider MAY declare, within a target schema that refers to SIMPLEST, additional object-classes that do not correspond to object-classes that SIMPLEST defines.
    - **Must have Naming attributes.** An extended object-class MUST have a “Name” attribute and MUST have a “GUID” attribute.

- **Should not bypass SIMPLEST**. An extended object-class SHOULD NOT *bypass* (that is, should not duplicate the purpose or behavior that SIMPLEST specifies for) any object-class that SIMPLEST defines UNLESS the provider also supports the functionally equivalent SIMPLEST object-class and UNLESS the provider automatically synchronizes the instances of both (the SIMPLEST object-class and the extended object-class) within that target.

- **Extended attributes.** A provider MAY declare, within a target schema that refers to SIMPLEST, additional attributes that do not correspond to attributes that SIMPLEST defines.

    - **Should not bypass SIMPLEST**. An extended attribute SHOULD NOT *bypass* (that is, should not duplicate the purpose or behavior that SIMPLEST specifies for) any attribute that SIMPLEST defines UNLESS the provider also supports the functionally equivalent SIMPLEST attribute for the same object-class and UNLESS the provider automatically synchronizes the values of both (the SIMPLEST attribute and the extended attribute) within each instance of that object-class.

- **Operational attributes**. A provider MAY declare a SIMPLEST operational attribute within a SIMPLEST object-class that does not normally contain that attribute. A provider MAY declare a SIMPLEST operational attribute within an extended object-class.

  For example, SIMPLEST specifies the full set of password-related attributes only for the Account object-class. If a provider intends to support the Password Capability for another object-class, then that object-class should declare the full set of password-related attributes (i.e., “Password”, “Password-Is-Expired”, “Password-Expire-Date”, “Password-To-Validate”, “Password-Is-Reset”). A provider that wishes to support only a subset of the operations within the Password Capability may declare in that provider's target schema only the attributes that correspond to that subset.

Arbitrary object-classes and attributes. A requester MAY pass (and a provider MAY return) instances of object-classes that are not declared within a target schema that refers to SIMPLEST. A requester MAY pass (and a provider MAY return) values for attributes that are not declared within a target schema that refers to SIMPLEST.

- **Arbitrary object-classes**. An arbitrary (i.e., undeclared) object-class SHOULD NOT bypass (that is, should not duplicate the purpose or behavior that SIMPLEST specifies for) any object-class that SIMPLEST defines UNLESS the provider also supports the functionally equivalent SIMPLEST object-class and UNLESS the provider automatically synchronizes the instances of both (the SIMPLEST object-class and the extended object-class) within that target..

- **Arbitrary attributes**. An arbitrary (i.e., undeclared) attribute SHOULD NOT bypass (that is, should not duplicate the purpose or behavior that SIMPLEST specifies for) any attribute that SIMPLEST defines UNLESS the provider also supports the functionally equivalent SIMPLEST attribute for the same object-class and UNLESS the provider automatically synchronizes the values of both (the SIMPLEST attribute and the extended attribute) within each instance of that object-class.

Appendix A. References

[ARCHIVE-1] OASIS Provisioning Services Technical Committee., email archive, **http://www.oasis-open.org/apps/org/workgroup/provision/email/archives/index.html**, OASIS PS-TC

[SPML-REQ] OASIS Provisioning Services Technical Committee., Requirements, **http://www.oasis-open.org/apps/org/workgroup/provision/download.php/2277/draft-pstc-requirements-01.doc**, OASIS PS-TC

[RFC2119] S. Bradner., *Key words for use in RFCs to Indicate Requirement Levels*, **http://www.ietf.org/rfc/rfc2119.txt**, IETF

[DSML] OASIS Directory Services Markup TC., *DSML V2.0 Specification*, **http://www.oasis-open.org/apps/org/workgroup/dsml/documents.php**, OASIS DS-TC

[SAML] OASIS Security Services Technical Committee., *XMLTitle*, **http://www.oasis-open.org/apps/org/workgroup/sstc/documents.php**, OASIS SS-TC

[DS] IETF/W3C., *W3C XML Signatures*, **http://www.w3.org/Signature/**, W3C/IETF

[XS] W3C Schema WG ., *W3C XML Schema*, **http://www.w3.org/TR/xmlschema-1/** W3C

[ATTR] OASIS Security Services Technical Committee., *SAML V1.0 Assertion Schema*, **http://www.oasis-open.org/committes/security/docs/cs-sstc-schema-assertion-01.xsd**

[SPML-Bind]] OASIS Provisioning Services Technical Committee., SPML V1.0 Protocol Bindings, **http://www.oasis-open.org/apps/org/workgroup/provision/download.php/1816/draft-pstc-bindings-03.doc**, OASIS PS-TC

Appendix B. Acknowledgments

The following individuals were voting members of the Provisioning Services committee at the time that this version of the specification was issued:

List Members Here:

Appendix C. Revision history

Rev	Date	By whom	What
D-01	22 July 2005	Editor	0.1 Rough Draft
D-02	22 July 2005	Editor	0.2 Rough Draft
D-03	3 April 2005	Editor	Reformat headings. Add Overview subsection that discusses schema entities and relationships.
D-04	26 March 2007	Editor	Discuss SIMPLEST schema (but not SIMPLEST as a profile of SPML). Update “Domain Model for Identity Management”. Update “Schema” section. Remove sections describing (and giving examples of) SPMLv2 operations. Enhance “Conformance” section.

544

Appendix D. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification, can be obtained from the OASIS Executive Director.

OASIS has been notified of intellectual property rights claimed in regard to some or all of the contents of this specification. For more information consult the online list of claimed rights.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.