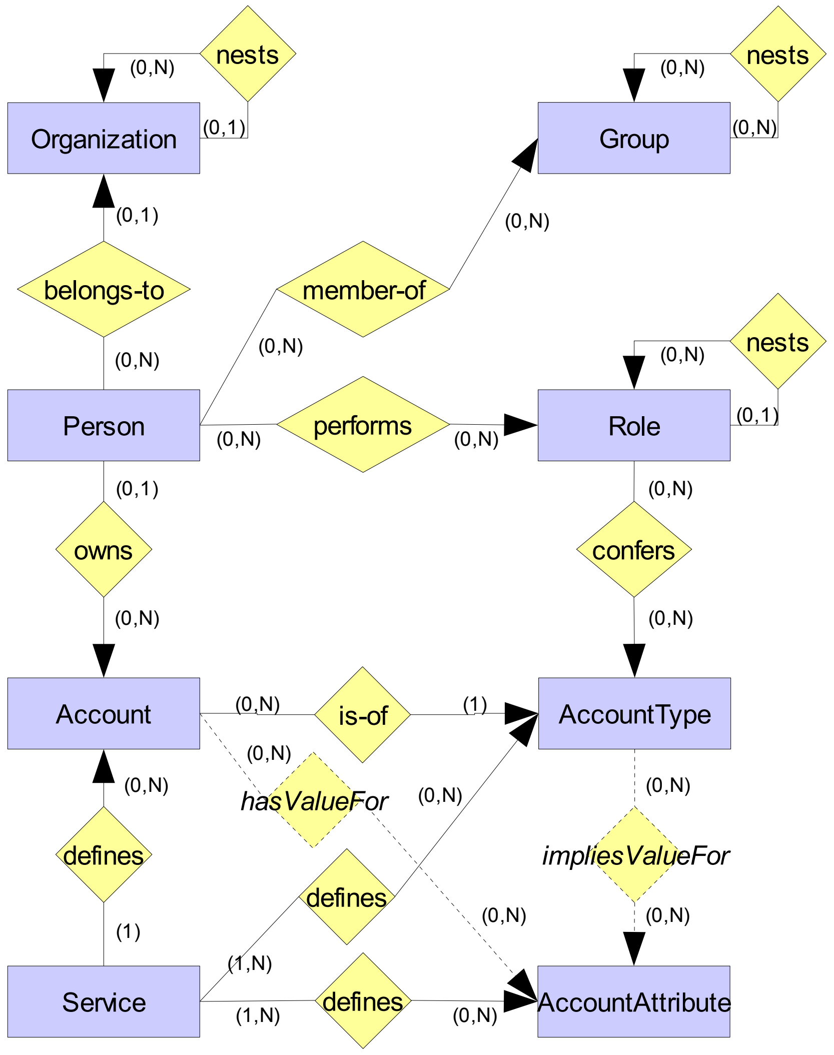
Domain Model for Identity Management

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

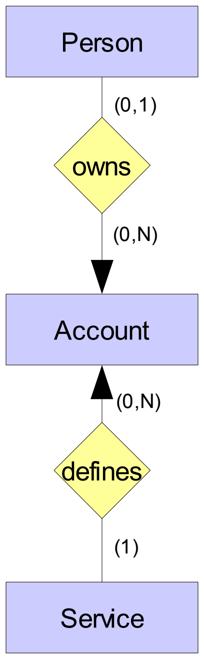


Each of the following subsections presents a subset of the domain model, beginning with the most familiar:

- The first subsection below presents Person, Account and Service.
- The next subsection below presents Organization, Group and Role.
- A third subsection below presents AccountType and AccountAttribute.

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

1.1. Person, Account and Service

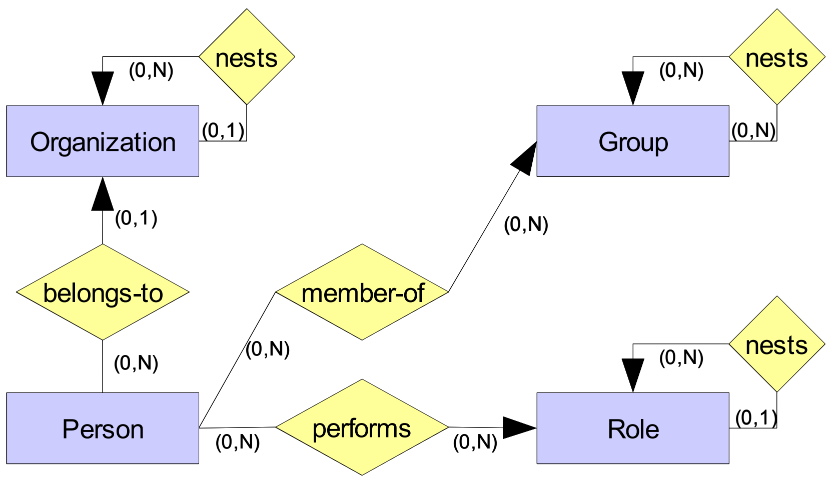


The Person and Account schema entities are fundamental to Identity Management. An instance of Person normally represents a human being. An instance of Account normally represents a person *within the scope of a particular computer system or application*. **Each person may *own* (that is, may be responsible 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**.

1.2. Organization, Group and Role



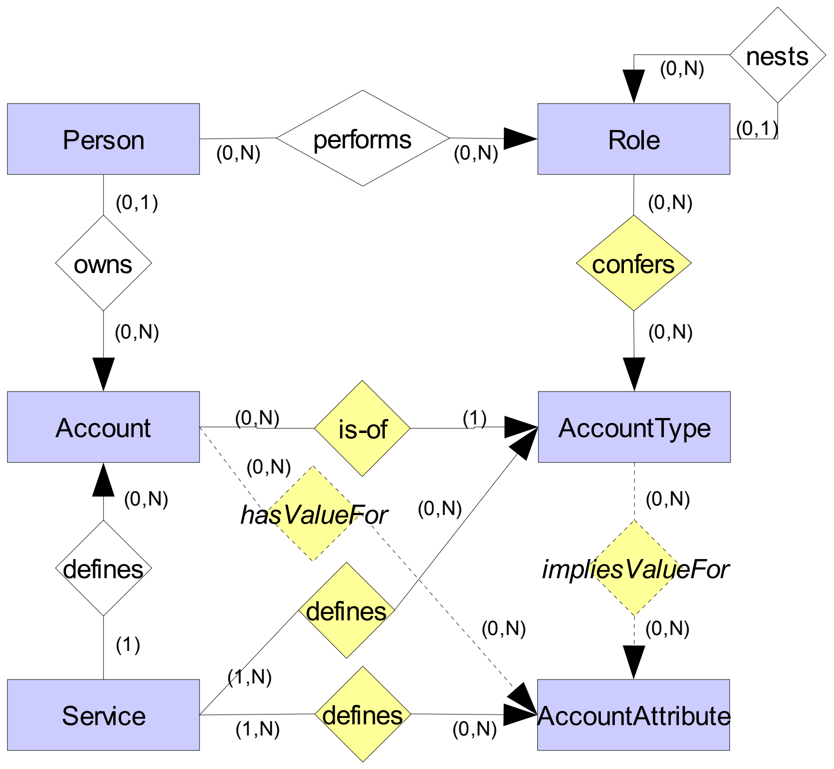
The Organization schema entity is ubiquitous in directory services (and therefore is common in identity management systems). An instance of Organization usually represents the management structure of a corporate entity—that is, an entity that consists of more than one person. The most common management 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 "AccountType and AccountAttribute" discusses this further.

1.3. AccountType and AccountAttribute



This section describes the entities and relationships that are the least 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 types that are available on a service.) Each account type represents a named category of account. For example, the “default” type of account may imply only basic or standard access to that service, whereas an “administrator” account 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 types. At least one service must define each account type.**

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" type of account—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 types. Any number of roles may confer each account type.**

Each type of account (for example, an "administrator" account type) may imply a set of values for (each of any number of) attributes that grant additional access on the service. (An "administrator" account-type 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 type may imply values for any number of account attributes. Any number of account types may imply values for each account attribute.**

Every account is of some account type. By default, an account is an instance of the default account type 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 type, then the account must reflect its account type 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 of some account type. Any number of accounts may be of 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.

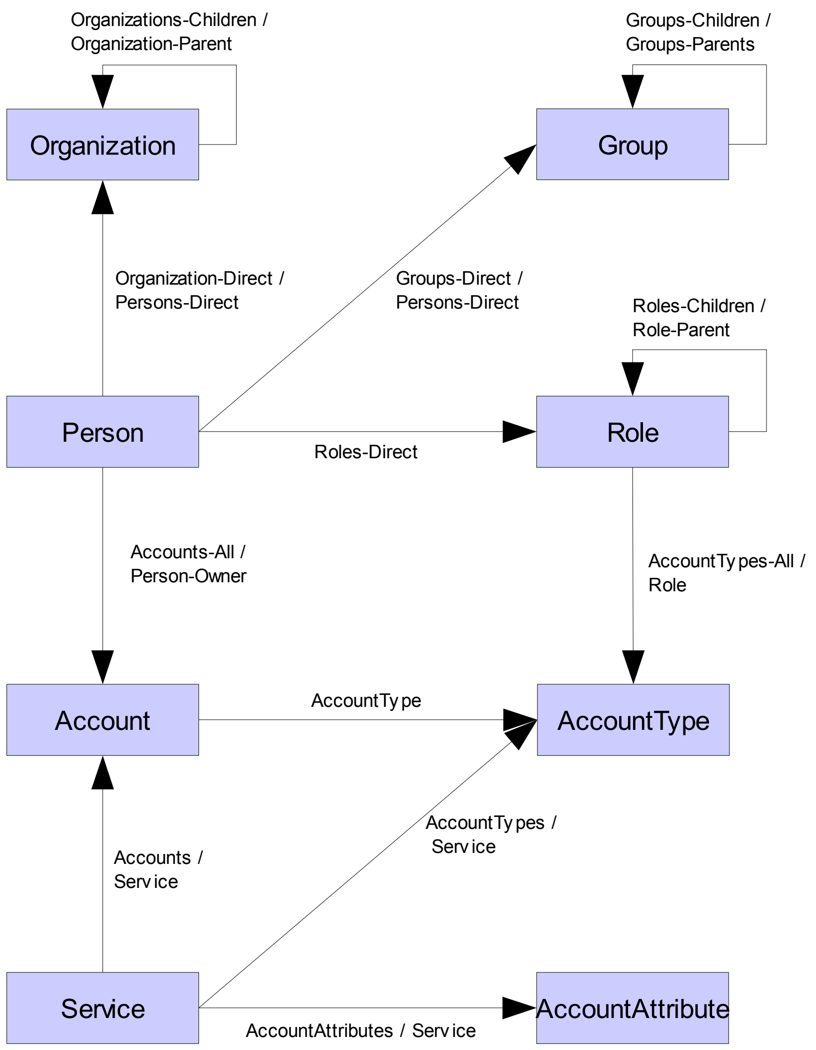
Imagine the following situation:

- 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.

1.4. 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.


Organizations-Children / Organization-Parent
Groups-Children / Groups-Parents
Organization
Group
Organization-Direct / Persons-Direct
Groups-Direct / Persons-Direct
Roles-Children / Role-Parent
Person
Roles-Direct
Role
Accounts-All / Person-Owner
AccountTypes-All / Role
Account
AccountType
AccountType
AccountTypes / Service
Accounts / Service
Service
AccountAttributes / Service
AccountAttribute

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.

AccountType 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 "AccountType" as an object class. Each Service may expose a multi-valued attribute called "AccountTypes". Each value of "AccountTypes" identifies an instance of AccountType that the Service defines. Each instance of AccountType defines a category of account on that service. Each instance of Account has a single-valued "AccountType" attribute that identifies the type of the account.

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