Schema, attributes and mapping

> ⚠ **Version Warning**
>
> The content below is for Apache Syncope <= 1.2 - for later versions the Reference Guide is available.

Introduction

The primary purpose of identity management systems is to manage data belonging to *users*; it is common practice in such systems to define as well entities called *roles* that helps in defining and enforcing security policies. In addition to this, Syncope explicitly represents the fact that users can be assigned to roles by mean of *memberships*.

In summary, Syncope manages data about three kind of entities:

1. **User**
2. **Role**
3. **Membership**

When saying "data", Syncope refers to a collection of so-called *attributes*.

This means that Syncope will manage User attributes, Role attributes and Membership attributes.

Attribute

An attribute is a *(key,values)* pair where

- *key* is a string label (i.e. *Surname*)
- *values* is a (possibly singleton) collection of data (i.e. [Doe] but also [john.doe@syncope.apache.org, jdoe@gmail.com])

The type of values that can be assigned to each attribute is defined via *schemas*.

Syncope will manage plain attributes, derived attributes and virtual attributes for users, roles and memberships.

Plain attributes

In this case attribute values are persisted into the Syncope internal storage.
Plain attribute values are:

- updated via synchronization from external resources
- available for propagation towards external resources

Read-only

Read-only attribute value(s) cannot be changed via standard operations (e.g. REST, synchronization from external resources, ...). Such attributes are reserved for internal usage, so their value(s) can be changed only in the underlying DBMS or from inside the workflow.

Derived attributes

Sometimes it is handful to obtain values as arbitrary combination of other attributes' values: for example, with 'Firstname' and 'Surname' plain attributes, it is natural to think that 'Fullname' could be somehow defined as the concatenation of Firstname's and Surname's values, separated by a blank space.

You can define a derived attribute via a JEXL expression combining values of some plain attributes.

Derived attribute values are:

- indirectly updated via synchronization from external resources (when component plain attributes' values are updated via synchronization)
- available for propagation towards external resources

Virtual attributes

With virtual attributes, values are not kept into the Syncope internal storage but somehow *linked* from an external resource.

The typical usage of virtual attributes is when an attribute can change on an external resource without notice and there is need of having access to the most updated value without relying upon synchronization.

Furthermore, for performance reason, the best practice is to keep the number of plain and derived attributes as low as possible: Apache Syncope should declare plain attributes just for data on which it must have the ownership; the rest should be declared virtual.

A virtual attribute can be mapped among several resources.
The values of a virtual attribute are the composition (in a distinct way) of values coming from each resource the virtual attribute is mapped on.

Virtual attribute values are always retrieved from an external resource either in case of *SYNCHRONIZATION*, *PROPAGATION* or *BOTH* mapping purpose.
The only way to avoid virtual attribute values retrieving from a certain resource is to remove *SEARCH* capability from the resource connector itself.

Virtual attribute values are:

- unaffected by synchronizing the resource where they come from (if and only if the values are coming from one resource only)
- available for propagation towards external resources

For performance optimization, virtual attributes are managed by an internal cache to limit access to external resources.
Virtual attribute cache is not configurable and cannot be disabled.

Each entry into the cache is key/values pair.
The key is composed of:

- attributable type (USER or ROLE)
- attributable id
- virtual schema name.

The entry expire time is one minute. By the way, it can expire before if the referenced virtual attribute is interested by a propagation.
Entry expiration could be forced by interacting directly with VirAttrCache bean. This can be done just by exploiting available Syncope extension points.

Read-only

When attribute value(s) from an external resource are needed only to be read within Syncope, and can only be changed from the own resource, virtual read-only attributes are fit for the job.

Schema

An attribute schema describes the values that attributes with that schema will held:

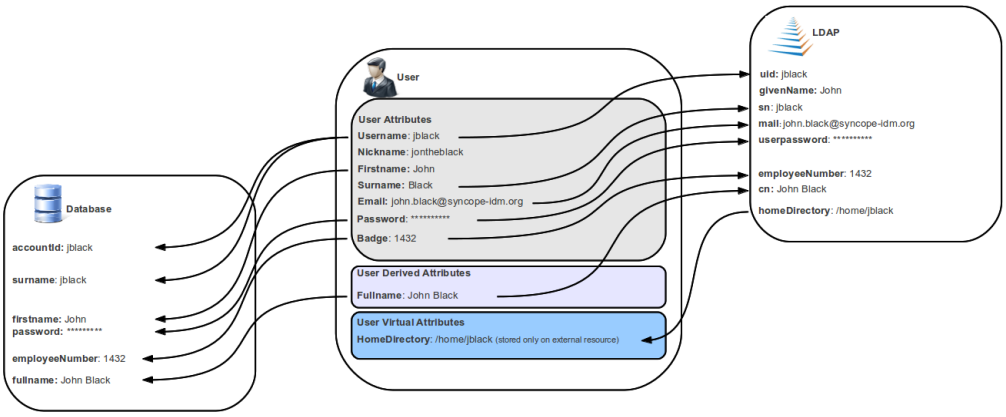
- *type* (String, Enum, Boolean, Long, Double, Date)
- whether values must respect UNIQUE constraint or not
- whether values must be singleton or not
- whether providing a value is mandatory or not
- whether input is accepted or not (*read-only*)
- whether values must be validated by some provided validator (like as *EMailAddressValidator*)
- how non-string values must be converted into / parsed from strings (*conversion pattern*)

This means that Syncope will manage schemas, derived schemas and virtual schemas for users, roles and memberships.

Schema Mapping

If Syncope was only able to define schemas and manage attributes for its internal storage, there would have been little to profit from by deploying an IdM solution.

One of most important features is about to link such attributes to external resources (LDAP server, Database, ...) so that propagation and synchronization can take place effectively.



Mapping purposes

Each mapping item can be configured for a specific purpose:

- **SYNCHRONIZATION** - mapping item will be considered just during synchronization.
- **PROPAGATION** - mapping item will be considered just during propagation.
- **BOTH** - mapping item will be considered always.