LDAP Provisioning Plugin

The LDAP Provisioning Plugin is designed to provision Registry data into an LDAP server.

Understanding LDAP Attribute Management

LDAP attributes are grouped into collections called object classes. The LDAP Provisioning Plugin supports several object classes, and various attributes within those object classes. Depending on the object class, it may possible to select some (but not all) attributes within an object class for export. The Plugin assumes full control over any enabled attribute within an object class.

Prior to v2.0.0, the Plugin assumed that if an object class is enabled, it controls all attributes within that object class are within its control, even if they are not configured. However, this can cause problems (eg: if you are using an older version of an object class than what the Plugin supports, or if you have another application that you want to manage an attribute). As of v2.0.0, two modes are supported, selected via the *Unconfigured Attribute Mode* setting:

- **Ignore**: Unconfigured (disabled) attributes within an enabled object class are ignored. Note that if you subsequently disable an attribute after having previously enabled it, existing values of that attribute will *not* be removed. You will need to manually clean them up. This is the default behavior beginning with Registry v2.0.0.
- **Remove**: Unconfigured attributes within an enabled object class are removed. This is the default behavior prior to Registry v2.0.0.

Regardless of this setting, attributes associated with object classes not enabled are left alone (except as described in *Operations*, below).

LDAP Attribute Options

As of v3.2.0, Registry supports LDAP Attribute Options, mostly in accordance with voPerson recommendations. Enabling attribute options in the LDAP Provisioning Plugin requires support for attribute options to be enabled on your LDAP server. The specifics for doing so are beyond the scope of this document, but note that it may be necessary to specify which options are in use when doing so. For example, this global configuration directive is necessary with OpenLDAP (here using OLC/cn=config):

```
# Note lang- becomes undefined by default when using this directive
olcAttributeOptions: lang- app- internal prior role- scope- time- type-
```

> ⚠ As of version 2.4.47 OpenLDAP has a bug that prevents the value for olcAttributeOptions from being modified once set.

> ⚠ If LDAP Attribute Options are enabled and then subsequently disabled, the LDAP Provisioning Plugin may not be able to correctly rewrite existing LDAP entries. It may be necessary to manually clean up existing entries.

Operations

⚠ Versions prior to Registry v2.0.0 may not be consistent with this documentation.

Externally Managed Attributes are those not managed by this Plugin. This includes all attributes except:

- Attributes enabled for export, within object classes enabled for export.
- Attributes defined by LDAP Schema Plugins and enabled for export.
- If *Unconfigured Attribute Mode* is *Remove*, all other defined attributes within object classes enabled for export (including those defined by Schema Plugins).

Registry CO Person Transaction	LDAP Action	Externally Managed Attributes
Add	Add entry to LDAP (if entry already exists it will be deleted and replaced)	Deleted
Edit	Update configured attributes only	Untouched
Status Set To *Grace Period*	No changes (unless attributes change as part of grace period)	Untouched
Status Set To *Expired* or *Suspended*	Update entry to maintain only Person attributes (include Unix Cluster Account attributes) for referential integrity (no Role or Group attributes, including Entitlements)	Untouched
Status Set Back To *Active*	Restore Role, Group, and Entitlement attributes, or add entry to LDAP if not present	Untouched
Delete, or Status Set To *Deleted* (or any other status not specified above)	Remove entry from LDAP	Deleted
Manual Provision	If entry exists: Update configured attributes only If entry does not exist: Add entry to LDAP ⚠ Attributes are subject to CO Person and Person Role Status ⚠ To completely erase and rewrite a record, an administrator must remove the record from LDAP (manually or by setting the person status to eg *Deleted*) before manually provisioning	Untouched

Registry CO Group Transaction	LDAP Action
Add	Write CO Group record (including memberships) to LDAP, but only if there is at least one member*
Edit	Write CO Group record (not including memberships) to LDAP, but only if there is at least one member*
Delete	Write CO Group record to changelog (attributes will be empty)
Manual Provision	Write CO Group record (not including memberships) to LDAP, but only if there is at least one member*

* The `groupOfNames` schema requires at least one member. If there are no members of a group, the Provisioner will delete the group.

Note that adding or deleting group memberships will trigger edit provisioning on both the affected CO Person and the affected CO Group.

Configuration

> ⚠ When using this plugin, it is recommended to add database encryption for the `password` column in the table cm_co_ldap_provisioner_targets.

The LDAP Provisioning Plugin automatically converts the internal Registry data model into the following LDAP object classes:

- `person`
- `organizationalPerson`
- `inetOrgPerson`
- `eduPerson` (must be enabled)
- `eduMember` (must be enabled)
- `groupOfNames` (must be enabled)
- `posixAccount` (experimental, must be enabled)
- `ldapPublicKey` (must be enabled)
- `voPerson` (must be enabled; see voperson.org)
- `voPosixAccount` (experimental, must be enabled; see voperson.org)

When configuring the Plugin, you can select which object classes to use and which attributes within those object classes to export to LDAP. When attributes come from data model attributes that are typed, a specific type can be selected, or all types can be selected. When multiple values are not supported, the first obtained value will be exported. Unless otherwise noted, only attributes attached to the CO Person record are exported. (Org Identity attributes are not.)

Attributes are mapped as follows:

Attribute	Object Class	Data Model	Multiple Values Exported?	Attribute Options Supported	Introduced
cn	person, posixAccount, voPosixAccount	cm_names	Only the primary name attached to the CO Person is exported	lang	v0.8
cn	groupOfNames	cm_co_groups name	❌		v0.8.2

Attribute	ObjectClass	Source	Note	Options	Version
cn	voPosixGroup	cm_identifiers identifier	Only if attribute options are enabled	scope	v3.3.0
description	groupOfNames	cm_co_groups description	❌		v0.8.2
displayName	inetOrgPerson	cm_names	❌	lang	v2.0.0
eduPersonAffiliation	eduPerson	cm_co_person_roles affiliation (possibly mapped via cm_co_extended_types)	✅	role	v0.8
eduPersonEntitlement	eduPerson	cm_co_services (according to member cm_co_groups)	✅		v2.0.0
eduPersonNickname	eduPerson	cm_names	✅	lang	v2.0.0
eduPersonOrcid	eduPerson	cm_identifiers identifier where type is `orcid`	✅		v2.0.0
eduPersonPrincipalName	eduPerson	cm_identifiers identifier	❌		v0.8
eduPersonPrincipalNamePrior	eduPerson	cm_identifiers identifier	✅		v2.0.0
eduPersonScopedAffiliation	eduPerson	cm_co_person_roles affiliation (possibly mapped via cm_co_extended_types, with scope appended)	✅	role	v2.0.0
eduPersonUniqueId	eduPerson	cm_identifiers identifier (with scope appended)	❌		v2.0.0
employeeNumber	inetOrgPerson	cm_identifiers identifier	❌		v0.8
employeeType ⓘ While not deprecated, as of v3.2.0 the use of *voPersonAffiliation* is recommended instead	inetOrgPerson	cm_co_person_roles affiliation	✅	role	v0.9.2
facsimileTelephoneNumber	organizationalPerson	cm_telephone_numbers	✅	role	v0.8
gecos	posixAccount	cm_names	❌		v0.9
gidNumber	posixAccount	cm_identifiers identifier where type is `gidNumber`	❌		v0.9
givenName	inetOrgPerson	cm_names given	Only the primary name attached to the CO Person is exported	lang	v0.8
hasMember	eduMember	cm_identifiers identifier	✅		v0.8.2
homeDirectory	posixAccount	cm_identifiers identifier where type is `homeDirectory`	❌		v0.9
isMemberOf	eduMember	cm_co_groups name (where cm_co_group_members member is true)	✅		v0.8
l	organizationalPerson	cm_addresses locality	✅	lang, role	v0.8
labeledURI	inetOrgPerson	cm_urls url and description (if set)	✅		v3.1.0
loginShell	posixAccount	Currently hard coded	❌		v0.9
mail	inetOrgPerson	cm_email_addresses mail	✅	role	v0.8
member	groupOfNames	cm_co_ldap_provisioner_dns DN	✅		v0.8.2
mobile	inetOrgPerson	cm_telephone_numbers	✅	role	v0.8
o	inetOrgPerson	cm_co_person_roles o	✅	role	v0.8
ou	organizationalPerson	cm_co_person_roles ou	✅	role	v0.8
owner	groupOfNames	cm_co_ldap_provisioner_dns DN	✅		v0.8.2
postalCode	organizationalPerson	cm_addresses postal_code	✅	lang, role	v0.8
pwdAccountLockedTime	n/a (see pwdPolicy)	cm_co_people status (set when status is Expired or Suspended)	❌		v2.0.0
roomNumber	inetOrgPerson	cm_addresses room	✅	lang, role	v0.9.4
sshPublicKey	ldapPublicKey	cm_ssh_keys	✅		v0.9

sn	person	cm_names family	Only the primary name attached to the CO Person is exported	lang	v0.8
st	organizationalPerson	cm_addresses state	✅	lang, role	v0.8
street	organizationalPerson	cm_addresses street	✅	lang, role	v0.8
telephoneNumber	organizationalPerson	cm_telephone_numbers	✅	role	v0.8
title	organizationalPerson	cm_co_person_roles title	✅	role	v0.8
uid	inetOrgPerson, posixAccount, voPosixAccount	cm_identifiers identifier	✅	scope	v0.8
uidNumber	posixAccount	cm_identifiers identifier where type is `uid Number`	❌		v0.9
userPassword	person	cm_passwords password where type is `CRYPT`, if Password Authenticator Plugin is enabled	✅		v3.1.0
voPersonAffiliation	voPerson	cm_co_person_roles affiliation	✅	role	v3.2.0
voPersonApplicationPassword	voPerson	cm_passwords password, see below for more	Only if attribute options are enabled	app	v3.3.0
voPersonApplicationUID	voPerson	cm_identifiers identifier If attribute options are enabled, see note below	✅	app	v3.2.0
voPersonAuthorName	voPerson	cm_names	✅	lang	v3.2.0
voPersonCertificateDN	voPerson	cm_certificates subject_dn	✅	scope	v3.2.0
voPersonCertificateIssuerDN	voPerson	cm_certificates issuer_dn	✅	scope	v3.2.0
voPersonExternalID	voPerson	cm_identifiers identifier	✅		v3.2.0
voPersonID	voPerson	cm_identifiers identifier	✅		v3.2.0
voPersonPolicyAgreement	voPerson	cm_co_t_and_c_agreements	✅	time	v3.2.0
voPersonSoRID	voPerson	cm_identifiers identifier	✅		v3.2.0
voPersonStatus	voPerson	cm_co_people status If attribute options are enabled, cm_co_person_roles status	Only if attribute options are enabled	role	v3.2.0
voPersonToken	voPerson	cm_totp_tokens serial	✅	type	v4.0.0
voPosixAccountGecos	voPosixAccount	cm_unix_cluster_accounts gecos	Only if attribute options are enabled	scope	v3.3.0
voPosixAccountGidNumber	voPosixAccount	cm_identifiers identifier	Only if attribute options are enabled	scope	v3.3.0
voPosixAccountHomeDirectory	voPosixAccount	cm_unix_cluster_accounts home_directory	Only if attribute options are enabled	scope	v3.3.0
voPosixAccountLoginShell	voPosixAccount	cm_unix_cluster_accounts login_shell	Only if attribute options are enabled	scope	v3.3.0
voPosixAccountUidNumber	voPosixAccount	cm_unix_cluster_accounts uid	Only if attribute options are enabled	scope	v3.3.0

⚠️ `posixAccount` support is experimental, and as of Registry v3.3.0 has changed significantly. For more information, see *Managing Unix Clusters*, below.

ⓘ For `ldapPublicKey` integration with OpenSSH, you may find this discussion helpful. Also note that recent releases of OpenSSH include a script that queries LDAP for authorized keys.

ⓘ

> ⓘ If attribute options are enabled, the export of `voPersonApplicationUID` changes so that an appropriate `app-` label can be appended. The value for the label is taken from Registry Services, and a matching Service must be identified in order for the identifier to be exported. If no matching Service is found, the identifier will *not* be exported. A Service is considered "matching" if it is configured with a *Service Identifier Type* of the same type as the identifier to be exported. If a match is found, the identifier will be exported with with the attribute option `app-shortlabel`, where *shortlabel* is the *Short Label* as configured within the Service.
>
> In voPerson terms, `uid` should hold the *General Application Identifier*, and `voPersonApplicationUID` should hold *Application Specific Identifiers.*

Configuring DNs

Base DNs must be configured for each LDAP Provisioning Target. A People Base DN is mandatory. A Group Base DN is only required if the `groupOfName` objectclass is enabled.

For People entries, an identifier label and type must be selected which will be used to create the person-specific portion of the DN. Be sure to pick an identifier that will always be defined for all people, as the Plugin will be unable to export records for which it cannot generate a DN. You may wish to use an identifier that you have configured Registry to assign automatically. The selected identifier must also be exported as part of the record (the Plugin will do this automatically if you don't configure it).

For Group entries, the name of the group is placed into `cn` and used to construct the DN. Thus, all Groups will have DNs of the form `cn=Group Name, Group Base DN`.

If an element of a DN changes for a CO Person or a CO Group, the Plugin will automatically assign a new DN and rename the entry the next time the entry is provisioned.

> ⚠ **LDAP v3 Required**
>
> The LDAP Provisioning Plugin requires LDAP protocol v3 in order to rename an entry when its DN changes.

Adding Additional ObjectClasses

As of Registry v2.0.0, populating object classes and attributes other than those described above is supported via LDAP Schema Plugins.

Adding Additional ObjectClasses (Externally Managed)

You may write to LDAP via other services or applications to maintain attributes that are not managed by COmanage Registry. For example, you might use a mailing list manager to maintain list memberships in LDAP, or you might use the Grouper Provisioning Plugin and Grouper's PSP to provision group memberships from Grouper to LDAP.

As of Registry v1.0.3, it is possible to specify additional objectClasses as part of a Person or Group record in order to allow for the external management of a portion of the record. Note the following considerations:

1. The objectClass must have no required attributes, since the LDAP Provisioning Plugin will write the initial record with no awareness as to the characteristics of the schema. If the objectClass has any required attributes, the record will fail to be written due to schema violation. (Supporting schemas with required attributes can be done via LDAP Schema Plugins).
2. Be aware of the implications of the operations described above. For example, if the LDAP Provisioning Plugin decides to delete an entry from LDAP, the attributes managed by external applications in that entry will also be deleted.

Removing ObjectClasses

⚠ When removing an objectclass (whether via configuration or by disabling LDAP Schema Plugins), keep in mind you may receive schema compliance errors from the LDAP server. This can happen because (eg)

1. COmanage had previously included an attribute `foo` in the objectclass `fooclass`.
2. When the objectclass is deconfigured, COmanage will emit a list of objectclasses that no longer includes `fooclass`.
3. However, the LDAP record still contains the attribute `foo`. COmanage does not touch this attribute because it is not configured to do so.
4. The LDAP server complains because the record does not contain an objectclass that defines `foo`.

In this scenario, it will be necessary to manually clean up the LDAP records to remove `foo` before COmanage can update the record.

Managing Unix Clusters

> ⚠ As of Registry v3.3.0, posixAccount attributes are obtained from the Unix Cluster Plugin. For documentation of prior behavior, see this earlier version of the documentation.

⚠️

Unix Clusters may be provisioned using the `posixAccount` objectclass ([RFC 2307](#)) or the `voPosixAccount` objectclass, part of [voPerson](#). The attribute mapping is described above, but be sure to note the following:

- Both `posixAccount` and `voPosixAccount` require `cn` and `uid` to be populated. The LDAP Provisioner will always provision `cn`, but `uid` must be explicitly configured.
- While a CO Person can have more than one Account associated with a particular Cluster, the LDAP Provisioner can only export one Account per CO Person due to how the objectclasses are defined. If more than one Account is found, it is non-deterministic as to which Account is exported.
- Because both `groupOfNames` and `posixGroup` are defined as `STRUCTURAL`, they cannot both be enabled within the same LDAP Provisioner configuration. Either can be used with `voPosixGroup`, however.

Attaching a Single Unix Cluster to an LDAP Provisioner

To attach a single Unix Cluster, simply enable the `posixAccount` objectclass and then select the desired Cluster from the popup. When the provisioner constructs values for the `posixAccount` attributes, it will use those associated with the specified Unix Cluster.

Similarly, `posixGroup` may be enabled for a single Unix Cluster. Note that `posixGroup` will use the same Cluster specified for `posixAccount`, so enabling `posixGroup` effectively requires enabling `posixAccount`.

A single Unix Cluster may also be provisioned using the `voPosixAccount` objectclass, but doing so requires the use of Attribute Options. Similarly, the `voPosixGroup` objectclass may also be provisioned for a single Unix Cluster but also requires the use of Attribute Options. See the next section for details on how to configure provisioning for the `voPosixAccount` and `voPosixGroup` objectclasses.

Attaching Multiple Unix Clusters to an LDAP Provisioner

Because the attributes defined for `posixAccount` are specified as `SINGLE-VALUE`, it is not possible to attach records for multiple Unix Clusters to a single LDAP record using it. Instead, enable the `voPosixAccount` objectclass (and make sure the objectclass is defined on the LDAP server). The use of `voPosixAccount` also requires the use of Attribute Options.

Similarly, `voPosixGroup` may be enabled to create group records for multiple Unix Clusters in a single LDAP record. Like `voPosixAccount`, the use of Attribute Options is required.

To attach multiple Unix Clusters to a single LDAP Provisioner instance:

1. Attribute Options must be enabled
2. Each Unix Cluster to be provisioned must have an associated [CO Service](#) defined. (The CO Service must have the Unix Cluster set for the *Cluster* configuration option.) A Unix Cluster can only be associated with one CO Service.
3. The CO Service must have a *Short Label* defined. The short label will becomes the scope in the attribute option.

Deprovisioning Unix Cluster Accounts

As of Registry v4.0.0, in order to maintain referential integrity, Unix Cluster Accounts are provisioned to LDAP for the following statuses:

- Active
- Expired
- Grace Period
- Locked
- Suspended

Additionally, the `posixAccount` and `voPosixAccount` objectclasses both require a primary group to be set. Therefore, the presence of a Unix Cluster Account record in LDAP should not be sufficient in and of itself to permit login. One of the following should also be checked at login, depending on local deployment capabilities and need:

- A membership in any group other than the Primary Group
- `voPersonStatus`
- `pwdAccountLockedTime`

To remove a Unix Cluster Account from LDAP either set the CO Person status to another value (such as `Deleted`), or set the Cluster Account status to something other than `Active` (such as `Suspended`).

Application Specific Passwords

As of Registry v3.3.0, Password Authenticators can be used to populate `voPersonApplicationPassword` when the following conditions are met:

1. Attribute Options are enabled.
2. There is a PasswordAuthenticator attached to a CO Service.
3. The CO Service is Active, and has a Short Label defined.
4. The CO Person has a Password set for the configured PasswordAuthenticator.
5. If the CO Service has a Service Group defined, the CO Person is in the associated CO Group.

See Also

- [cm_co_ldap_provisioner_targets](#)
- [cm_co_ldap_provisioner_attributes](#)

- cm_co_ldap_provisioner_dns
- Registry Services
- Service Tokens
- Unix Cluster Plugin
- voPerson
- RECIPE: SSH Public Key Management