Configuring Registry Identifier Assignment

COmanage Registry is capable of automatically assigning identifiers for objects within COs. Identifiers may be assigned on demand, as part of an Enrollment Flow, or as part of a Pipeline. If you only want to assign identifiers manually (ie: without the help of auto assignment), this page does not apply to you.

Identifiers can be automatically assigned to CO Person records, not to Organizational Identity records. If you don't know what this means, review Understanding Registry People Types. As of Registry v3.3.0, identifiers can automatically be assigned to CO Groups and CO Departments.

Identifier Assignments generally assume a roman character set (ie: ASCII-7, not UTF-8).

See also: Expectations For Identifiers

Identifier Assignment Contexts

As of Registry v3.3.0, Identifier Assignments apply within contexts. The supported contexts are

- CO Department
- CO Group
- CO Person

Identifier Assignments will only be applied to objects of the specified context. Prior to v3.3.0, Identifier Assignments only applied to CO Person records.

Assigning Identifiers to Automatic CO Groups requires Registry v4.1.0 or later.

Defining Identifier Assignments

To define an identifier assignment, select *Identifier Assignments* from your CO's menu. Any already defined assignments will be listed. Click *Add Identifier Assignment* to create a new one.

Fill in the form, paying careful attention to the following fields:

- **Context**: The type of object this Identifier Assignment will apply to.
- **CO Group**: As of Registry v4.1.0, Identifier Assignments can be restricted to members of a specified CO Group. CO People who are not members of this CO Group will not have an Identifier generated using this Identifier Assignment. (Only applies to CO Person context.)
- **Type**: The type you wish to be populated in cm_identifiers:type. Note that Extended Types are available for Identifiers.
- **Email Type**: If you select a type of "Mail" (ie: email), this field will become active. If you select an email type, then cm_email_addresses will be populated instead of cm_identifiers. (Does not apply to CO Group context.)
- **Login**: In general, CO Person identifiers are not used to log in to COmanage services (Organizational Identities are), so this should generally be left unchecked. (Only applies to CO Person context.)
- **Algorithm**: See *Specifying Identifier Formats*, below.
- **Format**: See *Specifying Identifier Formats*, below. If no format is specified, identifiers will simply be assigned as an integer, eg `109` or `523788`.
- **Permitted Characters**: See *Specifying Identifier Formats*, below.
- **Minimum**: For Random identifiers, the minimum value that may be assigned. For Sequential identifiers, the first value to be assigned.
- **Maximum**: For Random identifiers, the maximum value that may be assigned. Currently, the maximum may not exceed the value returned by PHP's `mt_getrandmax()` function, which is likely 2,147,483,647.

Specifying Identifier Formats

To understand identifier formats, you should understand the following concepts:

- Collision Numbers: A number used to take a string that is not unique and makes it unique. For example, the string `j.smith` with a collision number added becomes (eg) `j.smith.3`. Collision numbers can be assigned *sequentially* or *randomly*.
- Parameters: In a format specification, a parameter is replaced with some other string.
- Sequenced Segment: In a format specification, sequenced segments are incorporated into an identifier in order to generate a unique string. An identifier is first generated from a format without any sequenced segments. If that identifier is not unique, sequenced segments are added until a unique identifier is generated.

Collision Numbers

Identifier formats can be a bit tricky, so let's start with the easier ones. The parameter `(#)` means "replace with a collision number". A *collision number* is the next number that will generate a unique identifier. If your identifier is assigned using the sequential algorithm, it is the next unassigned integer beginning with the *minimum* value you configured. For identifiers assigned using the random algorithm, the collision number is selected randomly. Only one collision number is permitted in a format.

For example, if a format is specified like `C(#)`, then the character `C` will be prefixed to the collision number, eg `C109` or `C523788`.

The collision number can be made fixed width by specifying the number of characters *n* in the parameter as `(#:`*n*`)`. For example, the format `C(#:8)` will generate `C00000109` or `C00523788`.

Name Substitutions

It is also possible to generate identifiers based on one or more components of a CO Person name (as defined in cm_names). The following parameters are available:

- `(G)`: Given Name
- `(M)`: Middle Name
- `(F)`: Family Name
- `(g)`: Given Name (lowercased)
- `(m)`: Middle Name (lowercased)
- `(f)`: Family Name (lowercased)

For CO Department and CO Groups, the name of the Department or Group may be used:

- `(N)`: Object Name
- `(n)`: Object Name (lowercased)

So `(G).(F)@myvo.org` might generate `Albert.Einstein@myvo.org`. To use initials instead of a full name, simply limit the length of the name to 1 character. `(g:1).(f)@myvo.org` would generate `a.einstein@myvo.org` instead.

Identifier Substitutions

As of Registry v3.3.0, existing identifiers can be embedded in the format string, using the parameter `(I/name)` where *name* is the alphanumeric name of the Extended Type (not the display name), for example `(I/uid)@myvo.org`. This capability is available in all contexts. Note that an identifier of the specified type must already exist or the Identifier Assignment will fail. Also as of Registry v3.3.0, Identifier Assignments can be ordered, so it is possible to ensure that the first identifier is generated before the second identifier that uses it.

Note that while a length specifier for `(#)` specifies a fixed width padded with zeros, when used with name-based parameters such as `(G)`, the length specifier indicates a *maximum* width.

Random Substitutions

As of Registry v3.3.0, several types of random strings can be generated:

- `(h)`: Hexadecimal characters (0-9a-f)
- `(L)`: Random letters (A-Z, but no O to avoid confusion with zero)
- `(l)`: Random letters (a-z, but no l to avoid confusion with one)

Random substitutions support width specifiers, so (eg) `(l:5)` will generate a five character string of lowercase letters, such as `hxnwp`.

⚠️ It's important to understand the difference between random substitutions and collision numbers. Random substitutions are generated once as part of the identifier construction, and are not guaranteed to make a unique string. In contrast, if a collision number does not generate a unique string, it will be replaced until a unique string is found (or a limit is reached).

Random sequences can, and probably should, be combined with collision numbers. For example, `(L:3)(#:2)` will generate a string like `DGP23`. If that Identifier is already in use, the random string portion (`DGP`) will be preserved, but a new collision number will be generated, resulting in an Identifier like `DGP 77`.

Sequenced Segments

These formats can't guarantee a unique identifier if your organization is non-trivial in size, so a collision number should be added. `(G).(F)(#)@myvo.org` would generate `Albert.Einstein1@myvo.org`.

The problem here is you might not want to append a number for the first `Albert.Einstein`, only for the second. Or you might want to try a middle name first. The solution is to add a *sequenced segment*. A sequenced segment is denoted in brackets as a number followed by a colon, and includes the text (including parameters) to be used when that sequenced segment is in effect. When assigning identifiers, all sequenced segments will initially be ignored. Then, starting with `1` and incrementing by 1 each time, sequenced segments will be added in until a unique identifier is generated. Currently, up to 9 sequenced segments may be defined.

For example, consider the format `(G)[1:.(M:1)].(F)[2:.(#)]@myvo.org`. This somewhat confusing string will first generate `Werner.Heisenberg@myvo.org`. If that isn't unique, it will then generate `Werner.K.Heisenberg@myvo.org`. Finally, it will generate `Werner.K.Heisenberg.1@myvo.org`. You should probably set the minimum value in the identifier assignment configuration to start at 2 when used with sequenced segments. That would generate `Werner.K.Heisenberg.2@myvo.org` instead, which is presumable less confusing if there is already a `Werner.K.Heisenberg@myvo.org` assigned.

As of v2.0.0, there are two types of sequenced segments: *additive* and *single use*. Additive sequenced segments are denoted with `[` and `]`, and are inserted starting with their designated sequence and remain in place for future identifier attempts. Single use sequenced segments are indicated with an additional = inserted after the open bracket. So, for example, the segment `[1:.(M:1)]` will be inserted into the second and each subsequently generated identifier candidate, while the segment `[=1:.(M:1)]` will only be inserted into the second generated candidate (and no subsequent candidates).

The good news is you may not need to know all of this. Various common default formats are available via a drop down menu, and you may be able to just use one of those.

Permitted Characters

The substitutions described above are controlled by the *permitted characters*. Consider someone with the given name "Mary Anne" and the family name "Johnson-Smith". You might not want to allow spaces and dashes in the generated identifier, so specifying "AlphaNumeric Only" as the permitted characters will result in identifiers like "maryanne.johnsonsmith" instead of "mary anne.johnson-smith". "AlphaNumeric and Dot, Dash, Underscore" would generate "maryanne.johnson-smith".

Furthermore, if any *sequenced segment* generates text consisting only of non-permitted characters, it will be skipped.

⚠ Auto-generated identifiers are subject to Identifier Validation. Identifier Validator Plugins can be used to further constraint auto-generated identifiers.

Assigning Identifiers on Demand

Identifiers can be assigned on demand by viewing the identifiers associated with the CO Department, CO Group, or CO Person. An *Assign Identifiers* button will be available.

It is also possible to autogenerate identifiers for all CO Departments, CO Groups, and CO People (via *CO* > *Configuration* > *Identifier Assignments* > *Autogenerate Identifiers For All*). How this operation is handled depends on the version:

- Registry v4.0.0 and later: A Registry Job using the Core Job Plugin will be scheduled.
- Registry v3.3.x: A Registry Job using the Identifier Assignment Job Plugin will be scheduled.
- Registry v3.2.x and earlier: Identifier assignment for all CO People runs synchronously.

Assigning Identifiers via Enrollment Flows and Pipelines

When a petition created from an enrollment flow is approved, any identifier assignments defined for the CO will automatically be run.

Similarly, when a CO Person is created via Registry Pipelines, identifier assignment defined for the CO will automatically be run.

Assigning Identifiers Automatically

For CO Departments and CO Groups, identifier assignments will automatically be run when a new Department or Group is added. This is *not* true for CO People – identifier assignments are only run for CO People as described above.

Constraints

1. Identifiers of a given type must be unique within a CO.
2. If a CO Person already has an identifier of a given type, no additional identifier will be created.

Identifier Reassignment

Identifiers can be reassigned if the original identifier was actually deleted (as opposed to being set to status *suspended*). If you do not wish identifiers to be reassigned, set the status of identifiers that are no longer needed to *suspended*; do not actually delete them.

Prepopulating Identifier Assignment Collision Numbers

⚠ You don't need this section unless you know you need this section.

It is possible to manually prepopulate sequential collision numbers, which may be useful if you are migrating data from another system. There is not currently a user interface to handle this (CO-386), so the steps must be assigned manually.

First, if you haven't already, define the Identifier Assignment as described above. You will need the ID for the Identifier Assignment you wish to work with. In a URL like the following, the ID is **3**:

http://localhost/registry/co_identifier_assignments/edit/3/co:2

Next, determine the affix or affixes. These are equivalent to the format with parameters substituted (with %s replacing (#)). For example, a format used to generate identifiers consisting of a person's initials might be `(G:1)(M:1)(F:1)(#)`. You would need to create a row for *each* initial sequence you wish to set the sequence number for. eg:

```
SQL> insert into cm_co_sequential_identifier_assignments
     (co_identifier_assignment_id, last, affix)
     values (3, 122, 'jms%s');
SQL> insert into cm_co_sequential_identifier_assignments
     (co_identifier_assignment_id, last, affix)
     values (3, 176, 'rdm%s');
SQL> insert into cm_co_sequential_identifier_assignments
     (co_identifier_assignment_id, last, affix)
     values (3, 143, 'rlm%s');
```

Note that rows in this table are not automatically created until an identifier with a given affix is assigned.