Registry Transmogrification

Transmogrification is the process by which data is migrated from a Registry v4 instance to Registry v5.

Preparing for Transmogrification

> Transmogrification *only* supports upgrading from Registry v4.x to Registry v5.0.x. Deployments on versions prior to v5.x must upgrade to the latest v4.x release first. If Registry has moved on to version v5.1.0 or later, upgraders must first upgrade to the latest v5.0.x before upgrading again to the latest release.

1. Unpool Organizational Identities

Platforms brought up on Registry v3.0.0 or earlier had the option to enable Organizational Identity Pooling. This setting is no longer supported. Org Identities must be unpooled prior to Transmogrification.

2. Instantiate Extended Types

COs that were created using Registry v0.9.1 or earlier may be relying on default attribute types. For any such CO, click "Add/Restore Default Types" (via *Configuration* > *Extended Types*) for each available attribute.

From v0.9.2, Extended Types were enabled by default.

3. Migrate Login Identifiers

Login Identifiers are now attached to the *Person*, not the Organizational Identity. Transmogrification can be passed flags to migrate login Identifiers. Multiple flags can be provided. If no flags are provided Transmogrification will not perform any actions, except that login flags will be removed from Identifiers not associated with a Person.

- `--login-identifier-copy`: For any Identifier associated with an Org Identity that is flagged for login, a copy will be made attached to the associated Person record.
- `--login-identifier-type`: For any Identifier associated with an Org Identity that is of the specified type, a copy will be made attached to the associated Person record. The specified type should be the legacy v4 type string (eg: `eppn`) and not the transmogrified foreign key.

4. Check Group Names

Group name restrictions are more strictly enforced, and non-conformant names will cause the Group to not Transmogrify correctly. Check for any problematic names and fix them prior to running Transmogrification. Specifically:

1. Standard Groups may not be named starting with the string "`CO:`". This prefix is reserved for System Groups created by Registry, and existing System Groups (such as Administrator Groups) should not be renamed. (AR-Group-9)
2. Two Groups within the same CO may not have the same name. (AR-Group-1)

5. Consider Archival Requirements

As part of Transmogrification, certain database tables that are no longer required will not be migrated to the new database. This includes

- cm_co_org_identity_links: With the elimination of pooled Organizational Identities, OrgIdentities are linked directly to a CoPerson, and this intermediate table is no longer required.
- cm_org_identities: Org Identities that are not linked to a CO Person will not be migrated to the new database, nor will their associated data (such as cm_names). This includes Org Identities that were created from an Org Identity Source, but not linked (typically via a Pipeline) to a CO Person. Additionally, Org Identities that do have null (or empty) affiliations will not be migrated.

Some tables are also changed in a significant enough way that it is not possible to migrate all old records. This includes

- cm_co_group_members: This table is split into two (group_members and group_owners), and handling of non-manual memberships (eg: from group_nestings) is changing. Certain rows, including empty rows and changelog archives, are not migrated as there is not a consistent way to migrate these records.
- cm_co_jobs: Legacy job types (those prior to v4.0.0) cannot be migrated as they are no longer supported. Additionally, queued or in progress jobs are not migrated to avoid potential upgrade conflicts.

Additionally, deprecated or obsolete columns that were left in place to facilitate upgrading older versions will not be migrated to the new database.

It may be desirable to create a read-only archive of the old database for future auditing or historical needs.

Setting Up Databases

Transmogrification only supports MySQL/MariaDB and Postgres.

For deployments considering switching from MySQL/MariaDB to Postgres, Transmogrification may be an easier way to do so (in comparison to other migration techniques).

Transmogrification requires two database configurations in `local/config/database.php`:

- `default`: The new, target database that will be used for v5+ going forward. This database should initially be completely empty (ie: newly `CREATE`d).
- `transmogrify`: The source v4 database that will be converted. Transmogrification will not modify this database in any way, but the database does need to be in a consistent state (ie: no transactions running), so it may make sense to use a snapshot for test loads rather than pointing to a production instance. Also, make sure no cron jobs might cause changes to state.

(A template for this file can be found in `app/config/database.php.dist`.)

Install Database Schema

Install the database schema as follows;

```
$ cd $REGISTRY/app
$ ./bin/cake database
```

Do *not* use the `setup` command.

Running Transmogrification

```
$ ./bin/cake transmogrify [-vq] [--login-identifier-copy] [--login-identifier-type <type>] [table [...]]
```

where

- `--login-identifier-copy`: See *Migrate Login Identifiers*, above
- `--login-identifier-type <type>`: See *Migrate Login Identifiers*, above

A list of tables may be provided, however this should only be used by developers or those with very specific needs. Tables must be specified in dependency order.

Supported Tables

Not all tables can be automatically transmogrified, additional steps may be required to migrate certain configurations from v4.

Supported	Not Supported
• ad_hoc_attributes • addresses	• dashboards • external_identity_sources (formerly org_identity_sources)

- api_users
- authentication_events
- co_settings
- cos
- cous
- email_addresses
- external_identities
- group_members
- group_nestings
- groups
- history_records
- identifiers
- job_history_records
- jobs
- names
- people
- person_roles
- telephone_numbers
- types (formerly extended_types)
- urls

- identifier_assignments
- provisioning_history_records (formerly provisioning_exports)
- provisioning_targets
- servers

Transmogrification Warnings and Errors

Transmogrification may generate various warnings, which will not stop processing, and errors, which will. In general, warnings and errors will go to STDERR, so it is possible to use shell redirects to capture output for subsequent review. eg:

```
$ ./bin/cake transmogrify 2> ~/stuff-to-look-at.log
```

`-v` and `-q` can be used respectively to increase and decrease the output level.

Found existing CO Person for Org Identity #, skipping

In trying to build an internal map of Org Identities to CO People, Transmogrification found an existing CO Person for the specified Org Identity. This most likely indicates the Org Identities was not correctly unpooled prior to running Transmogrification.

Skipping record #: Could not find value for org_identities co_person_id

Transmogrification could not map the specified OrgIdentity to a CoPerson (via cm_co_org_identity_links). ie: This is an unlinked OrgIdentity, and will not be migrated.

Skipping record #: CO ID not provided for Identifier.type

The Identifier record could not be mapped to a CO, most likely because it is attached to an OrgIdentity that was not linked to a CO Person (and therefore not migrated).

Skipping record # due to invalid foreign key

A dependent record was not loaded. For example, an Identifier derived from a Source OrgIdentity was not migrated because the Source OrgIdentity did not migrate due to not being linked to a CO Person. See the provided SQL error for specifics.

Skipping record #: member not set on GroupMember

In Registry v4.x, the table cm_co_group_members included flags for both membership and ownership. As of Registry v5, these attributes are stored in separate tables. This warning indicates that a record was found in the source table without a membership flag, and that row is not being copied. (A record indicating ownership but not membership will be correctly processed.)

Skipping record #: Type not found for #+Identifier.type+epuid+

The legacy {type} field could not be mapped to a foreign key in the types table, probably because Add/Restore Extended Types was not executed within the CO prior to Transmogrification.