Multitenancy Considerations

Registry is a multi-tenant application (each CO is a tenant, and the COmanage CO is a special tenant used to manage the platform), and as such some thought should be given to how data is managed within and across tenants.

Platform-Wide Configurations

Configurations that apply to the entire platform (eg: a default theme) should be made from within the COmanage CO. The *Platform* menu is deprecated.

See also: Special Characteristics of the COmanage CO

Data Opaqueness

In general, data should reside within a single CO, and data should be opaque (ie: not visible) to other COs. This is generally accomplished by having a foreign key to some sort of parent record such as CoPerson or Co, and using standard authorization logic in `isAuthorized()`. For data models with indirect relations to these parent keys, it may be necessary to override functions like `AppModel::findCoForRecord()` and `AppController::calculateImpliedCoId()`.

Previously, an exception to this model applied to Org Identities, however this capability is no longer available. See Organizational Identity Pooling.

Unfreezing Foreign Keys

Registry v3.3.3 through v4.x

As of Registry v3.3.3, foreign keys are by default frozen when a record is initially saved. That is, if a record has a foreign key to `co_person_id`, once the record is created it can't be reassigned to a new CO Person. Freezing foreign keys prevents records from being moved across COs, in particular via the REST API (since the UI typically doesn't expose this capability).

In general, this is the correct default behavior, and should not be changed. However, there are some foreign keys that should be unfrozen. For example, if a model has an option pointing to a CO Group (say, to configure who is allowed to use the plugin), this foreign key might change from time to time (say, to point to a new group). In general, foreign keys that show up in the UI (probably as a SELECT) should probably be unfrozen.

Foreign keys can be unfrozen by updating the `content` validation rule for the relevant attribute. Two settings are currently supported:

- `CO`: The new value for the foreign key must point to a record that is in the same CO. This is typically the correct setting.
  - ℹ️ As of Registry v4.0.1, this setting is no longer required. All foreign keys are automatically constrained to the current CO.
- `yes`: Any value is permitted. This should only be used if the foreign key controls a cross-tenant configuration, or if the attribute name for some reason looks like a foreign key, but actually isn't.

⚠️ If the attribute name does not directly inflect to the parent model (eg: `co_group_id` to `CoGroup`), make sure the association rule is correctly defined in `$belongsTo`.

Example

```
public $belongsTo = array(
  'MembershipCoGroup' => array(
    'className' => 'CoGroup',
    'foreignKey' => 'membership_co_group_id'
  )
);

public $validate = array(
  'membership_co_group_id' => array(
    'content' => array(
      'rule' => 'numeric',
      'unfreeze' => 'CO'
    )
  )
);
```

Registry v5.0.0 and Later

As of Registry v5, foreign key freezing is based on the primary link for the table.