Vetter Plugins

See Also: Writing Registry Plugins and Registry Person Vetting

The interface for Vetter Plugins is *experimental* and may change across minor releases.

Some additional conventions are required when writing a Dashboard Widget Plugin.

1. The name of the Plugin should match the format `FooVetter`.
2. The Plugin must implement a `FooVetters` model and a corresponding Controller.
    a. The Controller should extend `StandardPluginController`, which provides some behind the scenes common functionality.
    b. When a new Vetting Step is instantiated, a skeletal row in the corresponding `foo_vetters` table will be created. There is no add operation or view required. The skeletal row will point to the parent Vetting Step.
    c. The entry point to the Vetter Plugin configuration will be `foo_vetter/foo_vetters/edit/#`. This will be called immediately after the Plugin is first instantiated.
3. The table `cm_foo_vetters` should include a foreign key to `cm_vetting_steps:id`.
4. The plugin must implement `FooVetter::vet($vettingStepId, $coPersonId, $coPetitionId=null)`, which will be called to perform the vetting. This function will be called from Registry Job Shell (via VetJob), and so should be able to operate without user interaction.
    a. This function should return an array with three elements:
        i. **result**: A VettingStatusEnum. See *Vetting Results*, below.
        ii. **comment**: A human readable comment describing the results.
        iii. **raw**: A text blob representing the actual vetting response generated/received, if appropriate. For example, this might be the full JSON returned from a remote API.
5. The plugin should implement a view for *review*, ie `foo_vetter/foo_vetters/review/#`. This view should parse the Vetting Result (available in `$vv_result`) and render a display suitable for an administrator to understand the request status. The Vetting Step configuration will be available in `$foo_vetter`. (The controller function is implemented in StandardPluginController, only the view file itself is required.)

Vetting Data Elements

It is up to the Plugin to determine which elements of the CO Person record to use for vetting, and how to use them.

Vetting Results

The `vet()` call should return a VettingStatusEnum as follows:

- `Canceled`: The request was canceled by an administrator.
- `Error`: An error occurred during vetting and no result could be determined.
- `Failed`: The subject failed vetting.
- `PendingManual`: The request requires administrator review. This result will cause Registry to send a notification to an administrator to take further action. If the system called by the Plugin will handle administrator review, `PendingResult` should be returned instead.
- `PendingResult`: The request is pending on further action. This result will cause Registry to stop processing the request. In order to resume processing, the Plugin should take the following steps:
    1. Call `VettingStep::resolve()` with the appropriate results.
    2. Call `VettingRequest::queue()` with `$requeue=true`. Processing will resume (or complete) at the next run of VetJob.