Registry PE TID: Enrollment Flows and Petitions (revision 3)

Overview

Registry PE Enrollment Flows have been completely redesigned. Instead of a complex, proprietary, mostly fixed workflow, Enrollment Flows are now plugin based, making them highly flexible. While the overall design is much simpler, there are significant implementation details to keep track of; those are described in this TID.

Actors

There are three types of Actors defined:

1. The **Petitioner** is the person who initiated the Enrollment Flow. The Petitioner need not be a registered Person, and can be anonymous.
2. The **Enrollee** is the person who is the subject of the Enrollment Flow, ie: the person being Enrolled. The Enrollee *might* be a registered Person for certain types of Flows (eg: Account Linking or Additional Role Enrollment). The Petitioner and Enrollee could be the same person.
3. The **Approver** is the Person who approves a Petition. A Petition can have more than one Approver, and more than one Person can be eligible to Approve a Petition. Approvers must be registered People. The Approver cannot be the Enrollee, and should not be the Petitioner.

An **unregistered** Actor does not have a corresponding Person record.

An **unauthenticated** Actor does not have a strongly authenticated Identifier provided by the web server.

Plugin Based Architecture and Flow

An Enrollment Flow consists of two "anchors" (`start` and `finalize`) and one or more instantiated plugins. `start` always runs, and each instantiated plugin runs in the configured order. `finalize` only runs if all plugins complete successfully.

Step Processing

- `start` is handled by EnrollmentFlowsController.
- Instantiated plugin Steps are handled by the `dispatch` action in the plugin that implements the Step.
- `finalize` is handled by PetitionsController.

Each instantiated Step has an Actor type configured to indicate who should run the step.

Each instantiated Step must have a unique `ordr` value (AR-EnrollmentFlowStep-1) so that the next step may be deterministically calculated, and so that the authorization for `finalize` can be calculated (see below).

Step Handoff

When a Step is completed, what happens next depends on whether the next Step has the same configured Actor as the Step that just finished. If so, a redirect directly to the next Step is issued. If not, a Notification is generated to the next Actor, and the current Actor is redirected to a landing page.

Step handoff is performed by `EnrollmentControllerTrait::transitionToStep()`. The next Step to perform is calculated by `EnrollmentFlowsTable::calculateNextStep()`.

Resuming a Petition

Enrollment Flows can be resumed, and Enrollment Flow Steps can be rerun, subject to the following constraints:

1. A Petition considered complete (see *Petition Status*, below) cannot be resumed.
2. A Step cannot be resumed if it has not yet run, unless it is the next Step to run.
3. The anchors can not be rerun.

Until a Petition is finalized, Plugins should allow actors to update their previously entered data, and should *not* update operational data.

Petition resumption is performed by `PetitionsController::resume()`, and is available via the Petition view.

Petitions

The main Petition artifact itself is now quite thin, with only a few core attributes such as the overall Petition status and information about some of the Actors.

Petition Email Address

Enrollment Flows can be configured to *Collect Enrollee Email*. When enabled, the Petitioner will be prompted to enter an email address for the Enrollee when starting the Flow. This email address *does not* automatically become part of the operational record, it is stored in the Petition for purposes of handing off to an unregistered Enrollee. This setting is intended for Flows that will register new users, Flows that will act on existing People should use the *Person Selector* plugin or a self service Flow where the Petitioner is the Enrollee.

Petition Status

The anchor steps are only responsible for two Petition statuses:

1. When `start` creates a new Petition, the Petition status is set to `PetitionStatusEnum::Created`.
2. When `finalize` completes a Petition, the Petition status is set to `PetitionStatusEnum::Finalized`.

All other statuses are set by plugin steps.

Several statuses can mark a Petition as completed: `Declined`, `Denied`, `Duplicate`, `Failed`, and `Finalized`. Petitions in any of these statuses are considered read only and cannot be resumed.

Authentication and Identification

Where possible, Enrollment Flows will use the integrated web server authentication to strongly identify actors. Actors need not be registered. If authentication is not available, a Petition-specific token will be issued. The order of preference is:

1. Authenticated user with a corresponding Person ID.
2. Authenticated, unregistered user. (The web server user Identifier will be stored in the Petition.)
3. Petition Token.

Approvers can only be identified using a corresponding Person ID, unregistered Approvers are not supported.

Token Generation

A token will be generated and attached to the Petition the first time `EnrollmentControllerTrait::transitionToStep()` needs to hand off to an unauthenticated Enrollee. For unauthenticated Petitioners, this will take place immediately after completion of `start`. Otherwise, for unauthenticated Enrollees, this will take place on the first Step configured for an Enrollee Actor.

Token Validation

Token validation is handled by `EnrollmentControllersTrait::getCurrentActor()`, when

1. There is a valid Petition in the request.
2. There is no authenticated identifier provided by the web server.
3. There is a token in the request.
4. There is a token stored in the Petition.

When a token is validated, the Actor is assigned the following roles:

- Petitioner, if there is no authenticated Petitioner stored in the Petition.
- Enrollee, if there is no authenticated Enrollee stored in the Petition.

An Actor authenticated by a token can be assigned both Petitioner and Enrollee roles, if appropriate.

Token Disuse

Once an authenticated identifier or Person ID is attached to the Petition, the token will no longer be used. This will typically happen at finalization, when operational attributes are created from the Petition attributes (and a Person record is created for new Enrollees), but it could happen sooner in accordance with plugin configurations.

willHandleAuth and calculatePermission

`RegistryAuthComponent` offers two hooks for controllers to customize authentication and authorization capabilities. `willHandleAuth()` is called before web server authentication might be triggered, and the controller can then return one of several flags to affect behavior. The controller can indicate that it will handle both authentication *and* authorization, in which case RegistryAuthComponent will *not* trigger web server authentication. `calculatePermission()` is called after authentication, to give the controller an opportunity to perform authorization.

This complicates things for Petition handling, since whether or not to use a Petition Token will be used must be determined in `willHandleAuth`. (This generally applies to Plugin `dispatch` and Petition `finalize` actions, as `start` does not require a token to get started.) To do so, the requested Petition and Enrollment Flow Step are examined. `Petition::useToken()` contains the relevant logic, which is basically if the requested Actor role is `Petitioner` or `Enrollee` (Approvers cannot be authenticated via tokens) *and* there is no authenticated Identifier or Person ID in the Petition, then Token authentication is permitted.

Authorization

Starting a Flow

Authorization to start an Enrollment Flow is controlled by the *Petitioner Authorization* configuration on the Enrollment Flow.

If the Enrollment Flow configuration allows for unregistered users (*None* or *Any Authenticated User*), the Petitioner is also considered to be the Enrollee.

Executing an Enrollment Flow Step

While each plugin implements its own `dispatch` action, generally plugin controllers should extend `StandardEnrollerController`, which implements common authorization via `willHandleAuth()` and `calculatePermission()`.

Finalizing a Petition

The Actor Type for `finalize` is determined from the last step configured to run. This implies (1) there must be at least one instantiated step and (2) there cannot be multiple steps that jump to `finalize`.

Viewing a Petition

`PetitionsController` and the related view are responsible only for basic Petition metadata. Each Plugin implements a `display` action that renders Plugin-specific information. Currently, only CO Administrators can view Petitions.

To simplify link generation, the URL `/petitions/result/x?enrollment_flow_step_id=y` will redirect to `/plugin/controller/display/z?petition_id=x`.

See Also

- Enrollment Flows Revision 3
- Registry PE Enrollment Flows
- Enrollment Flow Plugins (PE)
- Registry Table: enrollment_flow_steps
- Registry Table: enrollment_flows
- Registry Table: petition_step_results
- Registry Table: petitions