Rocky RESTful API - Draft 0.3

Josh Levinger, CEL
John Sebes, OSDV
Rock the Vote

Feb 6, 2012

## Overview

Rocky is web application that provides a web user interface that U.S. citizens can use for assistance in preparing a voter registration application document that is complete and correct for all Federal and state-specific requirements, and therefore should be accepted in most cases by the election officials to whom the user submits the completed, printed, signed form.

This specification defines an application programming interface that provides exactly the same functions as the Rocky web UI, but intended for a complementary purpose: use by other web applications that have or collect personal info via their own web UI, and which rely on the Rocky back-end for data validation, error reporting (particularly for state-specific conditions), PDF generation, and data storage. Via this API, Rocky provides a web service to service clients (web applications that use this API to interact with Rocky) that are software operated by Rocky "partner organizations".

Partner organizations also have access to aggregated data and reporting. These are provided by the Rocky Web UI in two ways partner organization staff can use to obtain statistics and summary information: a CSV file download provides aggregated registration data for all the registrations done via that partner; and a summary web page provides statistics about this partner-specific aggregated data set. This API provides analogous bulk data extraction, with the "*partner_registrations"* interface that returns the same data as is provided in the web UI CSV file download.

The primary interface is this API is the "*registrations*" interface, in which callers provide all the personal data needed to create a person's voter registration application form. Rocky checks the validity of each field, including state-specific validation rules. If there are errors, Rocky returns to the caller a set of error reporting on individual fields of personal data. If there are no errors, Rocky returns the URL of a PDF file that contains the validated information, properly formatted as a voter registration request form.

This API also includes an optional interface "*state_requirements*," used to pre-check the user data based on location and date of birth. One intended use of the pre-check function is for cases where service clients that already know the user's location, and perhaps also DOB. In such cases, the service client can find out from Rocky whether a user is ineligible, and if so, not even begin the web UI dialogues with the user to obtain personal information. Location ineligibility is the result of states that do not allow voter registration by mail, or do not accept Rocky-generated forms for some other reason. DOB is used to determine age ineligibility in the user's state.

When the user is not ineligible based on location or DOB, "*state_requirements*" returns several kinds of state-specific information that the caller could use as part its web UI with the user -- for example, the list of political parties that user can select for affiliation, or an indication that certain fields are optional for the state, or not collected by the state.

Finally, the API includes the "*partner_registrations"* interface to obtain records of past registration transactions. Like the similar UI in the Rocky Partner Portal, access requires authentication, and the records returned are only those pertinent to the authenticated entity.

## Interface Definitions

All requests and responses are JSON format. All fields are required unless otherwise specified as optional. All string fields have no length or format restrictions, unless specifically stated. Some field values have specific formats; others have parenthesized notations on field values.

*registrations*

Most fields are personal information. A few fields merit additional description:

- partner_id is a number that uniquely identifies a partner organization that uses Rocky as a service for the organization's members or community. Each transaction is logged by partner_id and every registrant record is tagged by partner_id.
- partner_tracking_id is similar, but the value is a tag defined by the partner organization and not interpreted by Rocky. It is use for tracking origin of a request, for example, for implementing "leader board" tracking. **NOTE:** partner tracking is a new Rocky feature that is not yet released.
- id_number is typically a state driver's license number state ID card number, or social security number; length and format vary by state, as does the explanation of what forms of ID are acceptable.
- opt_in_email is determined by the caller, and simply recorded; the caller can determine whether to always obtain the info from the user, or have a default. The Rocky web UI has a default of true that applies if the user does not provide a preference.
- opt_in_sms is determined by the caller, and simply recorded; the caller can determine whether to always obtain the info from the user, or have a default. The Rocky web UI has a default of false that applies if the user does not provide a preference.
- some fields are required in some states, optional in others, and not recorded in others; similarly some fields have permitted values that vary by state. State-specific information is provided by other interfaces in this API.
- pdfurl is an out parameter that is the URL of a PDF file containing the voter registration application form with the personal information provided in the request.
    - The URL itself is a large number, large enough to make it very difficult for adversaries to guess URLs, for example https://register.rockthevote.org/pdf/456136972787234.pdf
    - One intended use is that the caller embed the URL in a link in dynamically

generated HTML that is presented to the user.
- The URL is not returned until the URL is valid, with the PDF ready for download

**POST /api/v1/registrations.json**

POST { registration: {
lang: locale-compatible string *(en/es, etc)*,
partner_id: string (*series of digits, no specific length),*
partner_tracking_id: string*, (optional),*
date_of_birth: 'mm-dd-yyyy',
id_number: string (*optional*, *series of alnum, length and format state specific),*
email_address: string, *(optional, RFC syntax)*
first_registration: boolean,
home_zip_code: 'zzzzz' *(5 digit)*
us_citizen: boolean,
name_title: string (*must be one of "Mr.", "Mrs.", "Miss", "Ms.", "Sr.", "Sra.", "Srta.")*
first_name: string (optional),
middle_name: string (optional),
last_name: string,
name_suffix: string (*optional, must be one of "Jr.", "Sr.", "II", "III", "IV"*)
home_address: string,
home_unit: string (optional),
home_city: string,
home_state_id: string(2),
has_mailing_address: boolean,
mailing_address: string (required if has_mailing_address),
mailing_unit: string (optional),
mailing_city: string (required if has_mailing_address),
mailing_state_id: string(2) (required if has_mailing_address),
mailing_zip_code: string (required if has_mailing_address),
race: string *(required/optional is state-specific; value must be one of*
*"American Indian / Alaskan Native", "Asian / Pacific Islander",*
*"Black (not Hispanic)", "Hispanic", "Multi-racial", "White (not Hispanic)",*
*"Other", "Decline to State", "Indio Americano / Nativo de Alaska",*
*"Asiatico / Islas del Pacifico", "Negra (no Hispano)", "Hispano",*
*"Blanca (no Hispano)", "Otra", "Declino comentar")*
party: string *(required/optional and choices are state specific)*,
phone: string (optional),
phone_type: string (*optional, must be one of "Mobile", "Home", "Work", "Other",*
*"Movil", "Casa", "Trabajo", "Otro")*
change_of_name: boolean,
prev_name_title: string (optional),
prev_first_name: string (optional),
prev_middle_name: string (optional),
prev_last_name: string (required if change_of_name),
prev_name_suffix: string (optional),
change_of_address: boolean,
prev_address: string (required if change_of_address),
prev_unit: string (optional),
prev_city: string (required if change_of_address),
prev_state_id: string(2) (required if change_of_address),

prev_zip_code: string (required if change_of_address),
opt_in_email: boolean,
opt_in_sms: boolean
}}

Success: return 200
{ pdfurl: URL *(for the generated PDF)* }

Validation Error: return 400
{ field_name: string (*field where error occurred)*,
message: string (*determined by Rocky, for display by caller, in specified lang)* }

Syntax Error: return 400
{ field_name: string (*name of field that is not defined for this request)*,
message: string (*"Invalid parameter type")* }

Unsupported language: return 400
{ message: string }

*state_requirements*

Most input fields are personal information; most output fields are indicated as state specific. A few fields merit additional description:

- For input, one of *home_state_id* and *home_zip_code* is required. If both are provided, they must match. Really only one is needed.
- fields named "<foo>_msg" contain a string with state-specific information that explains the state-specific situation of <foo>, such as what the requirements are for reporting race, or stating party affiliation, or why the state does not allow party affiliation, or what the state's rules are about ID numbers for voter registration, or for minimum age for filing a voter registration application.
- fields named "sos_<foo>" refer to the contract information for the office of Secretary of State for the state specified in the home_state_id input field
- invalid state error is for a home_state_id input field value that is not a state, e.g. AJ
- non participating state error is for a correct state id, but for a state that does not participate in mail-in NVRA-form registration; the explanatory msg is state-specific
- to validate the age, make sure the person is 18+ yo and return localized error message if she's not
- no_party_msg is the name of the last option for party affiliation; in some states it is "none" while in others it is "decline to state" and over time any state can change its preferences
- requires_party_msg explains the state's rules for whether an application must or may include a party affiliation selection

**GET /api/v1/state_requirements.json** *checks state eligibility and provides state-specific fields*

*information*

GET: { lang: locale-compatible string *(en/es, etc),*
home_state_id: string(2),
home_zip_code: ‘zzzzz’ *(5 digit)*
date_of_birth: ‘mm-dd-yyyy’ *(optional)*
}

Success: return 200
{ requires_race: boolean,
requires_race_msg: string,
requires_party: boolean,
requires_party_msg: string,
no_party: boolean,
no_party_msg: string,
party_list: [ party_name: string],
id_length_min: integer,
id_length_max: integer,
id_number_msg: string,
sos_address: string,
sos_phone: string,
sos_url: string,
sub_18_msg: string
}
Invalid state ID: return 400
{ message: string }
Invalid ZIP code: return 400
{ message: string }
Incorrect ZIP code: return 400 *(ZIP does not match state)*
{ message: string }
Non-participating state: return 400
{ message: string *(state specific)* }
Invalid age: return 400
{ message: string *(state specific)*}
Unsupported language: return 400
{ message: string }
Syntax Error: return 400
{ field_name: string (*name of field that is not defined for this request)*,
message: string (*“Invalid parameter type”)* }

*registrations*

For a given partner, checks partner_id (ID in the “partners” table) and corresponding password, and returns partner-specific registration records. Partner account is created using Rocky web UI; password is set then, and can be reset later via UI. Optional “since” parameter limits returned records to those **created** after the date-time provided as parameter value; the records include those that were started but not completed, and a noted via the “status” out parameter.

**GET /api/v1/registrations.json** *returns registration records associated with the given partner*

GET: { partner_id: string (*series of digits, no specific length),*
partner_password: string*, (no specific length),*
since: string *(optional, UTC datetime format)*
}

Invalid Partner or Password: return 400
{ message: string }

Invalid since: return 400
{ field_name: “since”, message: string *(“Invalid parameter value }*

Syntax Error: return 400
{ field_name: string (*name of field that is not defined for this request)*,
message: string (*“Invalid parameter type”)* }

Success: return 200
{ registrations:
[status: string *(complete, or reason for incomplete)*,
create_time: string *(UTC datetime format),*
complete_time: string *(UTC datetime format),*
lang: locale-compatible string *(en/es/fr, etc),*
first_reg: boolean,
citizen: boolean,
first_registration: boolean,
home_zip_code: ‘zzzzz’ *(5 digit),*
us_citizen: boolean,
name_title: string,
first_name: string,
middle_name: string,

last_name: string,
name_suffix: string,
home_address: string,
home_unit: string,
home_city: string,
home_state_id: string(2),
has_mailing_address: boolean,
mailing_address: string,,
mailing_unit: string,
mailing_city: string,
mailing_state_id: string(2),
mailing_zip_code: string,
race: string,
party: string,
phone: string (optional),
phone_type: string (optional),
email_address: string,
opt_in_email: boolean,
opt_in_sms: boolean,
survey_question_1: string,
survey_answer_1: string,
survey_question_2: string,
survey_answer_2: string,
volunteer: boolean]
}