Rocky RESTful API
Version 3.0 - Draft
May, 2015

## 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.

**Release Notes**

**V3:**

Added these interfaces:

- **GET /api/v3/registrations/pdf_ready**
- **POST /api/v3/registrations/stop_reminders**
- **POST /api/v3/registrations/bulk**

Added parameter for registration and gregistration to indicate async PDF generation or response blocking PDF generation. Added parameter for custom stop_reminders url. Added UID to the response.

Modified format from **/api/v3/partners/partner.json** to **/api/v3/partners/[partner_id].json** Added application/registration/partner CSS urls, finish_iframe_url, external_tracking_snippet, registration_instructions_url to /**api/v3/partners/[partner_id].json** response. Added new fields to match column names for ActiveResource expectations and general standardization.

Modified request for

- **GET /api/v3/partners/[partner_id].json**

to allow for all survey question languages.

**V2.2:**

Added these interfaces:

- **GET /api/v2/partnerpublicprofiles/partner.json**
- **GET /api/v2/gregistrationstates.json**
- **POST /api/v2/gregistration.json**

- **GET /api/v2/gregistrations.json**

Since this addition did not modify the existing API, did not rev from v2 to v3.

## 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*

Creates a new registrant record using the input parameter data; and also triggers the side effects of a new record, e.g. sending confirmation email. 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.
- source_tracking_id is used to identify the source of a request. For example, in the Web UI, it is typically used as query parameter embedded in a URL in email messages, and identifies the request as part of a particular email campaign. The string value is un-interpreted, however, and may be used for any similar purpose
- partner_tracking_id is similar, but intended for use for partner-specific purposes, such as implementing a leader-board, that may require an a tag in addition to the source_tracking_id.
- 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. However, id_number is not optional. (Note: it is true that some states' localities will accept NVRA forms without an ID number, for the special case of people with neither state ID or an SSN. However, since we cannot record precisely how each state handles this special case, we require some ID number, to avoid creating registration requests that may be rejected.)
- email_address is required by default, even though it is optional for the paper form; however, we need it to contact the user to send them a link to the PDF we generate for them.
- opt_in_email is determined by the caller, and simply recorded in the registration record created by the call; the caller can determine whether to always obtain the info from the user, or have a default in the case of no user input. Regardless, the opt_in_email field is required to record true or false for opt-in, regardless of how the caller determined the user's preference. This field is used to record a user preference for email from RockTheVote.

- opt_in_sms is required in exactly the same way as opt_in_email.
- opt_in_volunteer is required in exactly the same way as opt_in_email.
- partner_opt_in_email, partner_opt_in_sms, partner_opt_in_volunteer are used in an analogous way to record user preference for communication with the partner organization.
- party is optional because it is not required on the form; however, the allowable values are state specific.
- 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 returned regardless of whether the PDF is ready for download.
- callback is an optional string parameter that, if present and not empty, changes the return value from JSON format to jsonp; that is, the JSON return value would be a string J, and if the callback parameter's value is F, then the return value will be F(J);
- custom_stop_reminders_url is an optional URL string that will be used in a link in reminder emails to registrants to cancel any further reminder emails. If the provided URL includes <UID>, that string will be replaced by the registrant's UID. If left blank, the default core system stop_reminders URL will be used.
- async specifies whether or not the request should return a response immediately (async=false) or wait for a PDF to be generated (async=true)

**POST /api/v3/registrations.json**

POST { registration: {

lang: locale-compatible string *(en/es, etc)*,
partner_id: string (*series of digits, no specific length),*

send_confirmation_reminder_emails: boolean,
collect_email_address: 'no' for false (optional)
source_tracking_id: string*, (optional),*
partner_tracking_id: string*, (optional),*
short_form: boolean (optional, default false),
state_ovr_data: (optional, hash, only used for stats and in cases where registrant went through finish-with-state steps but ended up finishing with rock the vote)

created_at: 'mm-dd-yyyy hh:mm:ss' (for statistics),
updated_at: 'mm-dd-yyyy hh:mm:ss' (for statistics),

date_of_birth: ‘mm-dd-yyyy’,

id_number: string (*series of alnum, length and format state specific),*
email_address: string, *(RFC syntax)*
first_registration: boolean,
home_zip_code: ‘zzzzz’ *(5 digit)*

us_citizen: boolean,
has_state_license: boolean,
is_eighteen_or_older: 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 *(optional; no validation b/c allowable 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,
opt_in_volunteer: boolean,
partner_opt_in_email: boolean,
partner_opt_in_sms; boolean,
partner_opt_in_volunteer: boolean,
survey_question_1: string, (required if survey_answer_1 is not blank)
survey_answer_1: string,
survey_question_2: string, (required if survey_answer_2 is not blank)
survey_answer_2: string,
callback: string (optional),
custom_stop_reminders_url: string (optional, URL format),
async: boolean (optional, default is “true”)
}}

Success: return 200
{ pdfurl: URL *(for the generated PDF), uid: UID string*}

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 }

Survey Question Error: return 400
{ message: string (“Question N required when Answer N provided”) }

*bulk_registrations*

Creates 1-n incomplete registrant records. Used for creating data in the core rocky system for statistics, but does not cause emails to be sent or PDFs to be created. Validations are not

performed

**POST /api/v3/registrations/bulk**

POST { registrations: [ {
status: string (indicates a step number that the user stopped at)
ineligible_non_participating_state: boolean (whether the registrant is ineligible due to the state not participating),
ineligible_age: boolean (whether the registrant is ineligible due to being too young),
ineligible_non_citizen: boolean (whether the registrant is ineligible due to not being a US citizen),
under_18_ok: boolean (whether the registrant decided to proceed despite being under 18),
remind_when_18: boolean (whether the registrant requested a reminder when 18),
age: integer (age of registrant),
javascript_disabled: boolean (whether the user had javascript disabled),
using_state_online_registration: boolean (whether the user selected to use the state online reg system),
finish_with_state: boolean (whether the user is in a “finish_with_state” flow)
*See registrations interface definition for remaining fields of “registration” input record*
}]
}

Success: return 200
{ abandoned_registrations: integer (number of records written)}

Error: return 400
[{ response_hash }]
Return an array of responses for each record that is either

{ } (empty hash)
or
{ type: string (ValidationError, SyntaxError, UnsupportedLanguageError),
field_name: string (*field where error occurred)*,
message: string (*determined by Rocky, for display by caller, in specified lang)* }]

*gregistrations*

Creates a new registrant record using the input parameter data, for the use case where the user was eligible to finish their registration on a government operated state-specific web site, chose to be re-directed there, and did not return -- presumably because they finished registration on the state's site.

Differs from *registrations* only in the following ways:

- Only the following parameters are required: lang, partner_id, send_confirmation_reminder_emails, date_of_birth, email_address, home_zip_code, us_citizen, name_title, last_name
- If send_confirmation_reminder_emails is true, then the emails sent are the the emails that are pertinent to this use case, rather than the standard use case of completing a registration using the Rocky UI
- The PDF is not generated, and therefore the async parameter has no effect
- Success has no output parameters
- There is an additional error return code

**POST /api/v3/gregistrations.json**

POST { registration: {
*See registrations interface definition for fields of "registration" input record*
}}

Success: return 200

Unsupported state: return 400
{ message: string }

*See registrations interface definition for other return data definitions.*

*bulk_gregistrations*

Same as gregistrations but expects an array of registrant records and returns the number of records created.

**POST /api/v3/bulk_gregistrations.json**

POST { bulk_gregistrations: [{
status: string (indicates a step number that the user stopped at)
created_at: (*UTC datetime format),*
updated_at: (*UTC datetime format),*
*See registrations interface definition for fields of "registration" input record*
}]}

Success: return 200
{ bulk_gregistrations: integer (number of records written)}

Error: return 400
[{ response_hash }]
Return an array of responses for each record that is either

{ } (empty hash)
or
{ type: string (ValidationError, SyntaxError, UnsupportedLanguageError),
field_name: string (*field where error occurred)*,
message: string (*determined by Rocky, for display by caller, in specified lang)* }]

*gregistrationstates*

Returns a list of state for which Rocky current supports per-state integration with states. There are no input parameters. State return data is the 2 letter code for the state, and the URL to use to send registrant data to the state's web site.

**GET /api/v3/gregistrationstates.json**

Success: return 200
{ states: [name: string *(2 letter state code)*, url: string *(URL format)] }

*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
- callback is an optional string parameter, exactly as described above

**GET /api/v3/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),*
callback: string (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”)* }

*pdf_ready*

For a given registration UID, returns true or false to indicate whether the PDF for that registrant exists.

**GET /api/v3/registrations/pdf_ready** *returns status of PDF generation for the given registrant*

GET: { UID: string *(no specific length),*
callback: string *(optional)*
}

Invalid Partner or API key: return 400
{ message: string }

Not Found UID: return 400
{ field_name: “UID”, message: string *(“Registrant not found” }*

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
{ pdf_ready: boolean, UID: uid}

*stop_reminders*

**POST /api/v3/registrations/stop_reminders** *For a given registration UID sets reminders_left to*

*0 to prevent further reminder emails.*

POST: { partner_id: string (*series of digits, no specific length),*
partner_API_key: string*, (no specific length),*
UID: string *(no specific length),*
callback: string *(optional)*
}

Invalid Partner or API key: return 400
{ message: string }

Not Found UID: return 400
{ field_name: “UID”, message: string *(“Registrant not found” }*

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 {
UID: string (no specific length),
first_name: string,
last_name: string,
email_address: string,
reminders_stopped: boolean
}

*registrations*

For a given partner, checks partner_id (ID in the “partners” table) and corresponding API key, and returns partner-specific registration records. Partner account was created using Rocky web UI; API key was set then, and can be reset later via admin UI. Optional “email” parameter filters the partner’s registration records, to return only those with an email address that matches the address provided in the parameter. 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. The callback parameter is an optional string parameter, exactly as described above.

**GET /api/v3/registrations.json** *returns registration records associated with the given partner*

GET: { partner_id: string (*series of digits, no specific length),*
partner_API_key: string*, (no specific length),*
since: string *(optional, UTC datetime format),*
email: string *(optional, email name at domain format),*
callback: string *(optional)*
}

Invalid Partner or API key: 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,
opt_in_volunteer: boolean,
partner_opt_in_email: boolean,
partner_opt_in_sms; boolean,
partner_opt_in_volunteer: boolean
survey_question_1: string,
survey_answer_1: string,
survey_question_2: string,
survey_answer_2: string.
finish_with_state: boolean,
created_via_api: boolean ]
}

*gregistrations*

For a given gpartner -- a government partner such as a local elections office -- checks partner_id (ID in the “partners” table) and corresponding API key, and returns partner-specific registration records. Partner account was created by the Rocky admin; API key was set then, and can be reset later by the Rocky admin. The record returned are those records for which the registrant’s ZIP code is one of the ZIP codes associated with the partner_id -- ZIP codes that were set by Rocky admin during creation, and can be updated.

Optional “email” parameter filters the partner’s registration records, to return only those with an email address that matches the address provided in the parameter. 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. The callback parameter is an optional string parameter, exactly as described above.

**GET /api/v3/gregistrations.json** *returns registration records associated with the given government partner*

GET: { gpartner_id: string (*series of digits, no specific length),*
gpartner_API_key: string*, (no specific length),*
since: string *(optional, UTC datetime format),*
email: string *(optional, email name at domain format),*
callback: string *(optional)*
}

Invalid Partner or API key: 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,
source_tracking_id: string,
partner_tracking_id: string ]
}

*partners*

Creates a new partner, very similar to partner creation in the Web UI of the partner portal.

**POST /api/v3/partners.json**

POST { partner: {
org_name: string,
org_URL: string (*URL format)*,
org_privacy_url: string *(optional)*,
contact_name: string,
contact_email: string *(email address format),*
contact_phone: string *(nnn-nnn-nnnn format),*
contact_address: string,
contact_city: string,
contact_state: string *(2 letter state code),*
contact_ZIP: *(nnnnn format)*
logo_image_URL: string *(URL format)*,
survey_question_1_[locale]: string, (where [locale] may be any of the enabled locale codes)
survey_question_2_[locale]: string, (where [locale] may be any of the enabled locale codes)
partner_ask_volunteer: boolean
}
}

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
{ partner_id: string (*series of digits) }*
*partner*

For a given partner, checks partner_id (ID in the "partners" table) and corresponding API key, and returns partner-specific profile records. Partner profile was created using Rocky web UI or API call POST partners. The callback parameter is an optional string parameter, exactly as described above.

**GET /api/v3/partners/[partner_id].json**

GET: { partner_API_key: string, *(no specific length),*
callback: string *(optional)*
}

Invalid Partner or API key: return 400

{ message: string }

Success: return 200

{

org_name: string

org_URL: string (URL format),

org_privacy_url: string (URL format),

contact_name: string,

contact_email: string *(email address format),*

contact_phone: string *(nnn-nnn-nnnn format),*

contact_address: string,

contact_city: string,

contact_state: string *(2 letter state code),*

contact_ZIP: *(nnnnn format)*

logo_image_URL: string *(URL format)*,

application_css_URL: string (URL format),

registration_css_URL: string (URL format),

parnter_css_URL: string (URL format),

finish_iframe_url: string (URL format),

survey_question_1_<loc>: string (where loc may be any enabled locale),

survey_question_2_<loc>: string (where loc may be any enabled locale),

whitelabeled: boolean,

rtv_email_opt_in: boolean,

partner_email_opt_in: boolean,

rtv_sms_opt_in: bollean,

partner_sms_opt_in: boolean,

rtv_ask_email_opt_in: boolean,

partner_ask_email_opt_in: boolean,

rtv_ask_sms_opt_in: bollean,

partner_ask_sms_opt_in: boolean,

ask_for_volunteers: boolean,

partner_ask_for_volunteers: boolean,

external_tracking_snippet: string,

registration_instructions_url: string (URL format),

application_css_present: boolean,

application_css_url: string (URL format)

registration_css_present: boolean,

registration_css_url: string (URL format),

partner_css_present: boolean,

partner_css_url:string (URL format),
primary: boolean
}

*partner*

For a given partner, checks partner_id (ID in the “partners” table) and corresponding API key, **without** an API key, and returns the portion of partner-specific profile records that are not private. Intended for unauthenticated access to public information.

Partner profile was created using Rocky web UI or API call POST partners. The callback parameter is an optional string parameter, exactly as described above.

In addition to the return field names above, there are return fields that are aliases for the above, provided for internal use with Rocky. Though present in return data, the alias fields can be ignored by most callers. They are:

- organization = org_name
- url = org_URL
- privacy_url = org_privacy_url
- name = contact_name
- email = contact_email
- phone = contact_phone
- address = contact_address
- city = contact_city
- state_abbrev = contact_state
- zip_code = contact_ZIP
- rtv_email_opt_in = rtv_ask_email_opt_in
- partner_email_opt_in = partner_ask_email_opt_in
- rtv_sms_opt_in = rtv_ask_email_opt_in
- partner_sms_opt_in = partner_ask_sms_opt_in

**GET /api/v3/partnerpublicprofiles/[partner_id].json**

GET: { partner_id: string (*series of digits),*
callback: string *(optional)*
}

Invalid Partner or API key: return 400
{ message: string }

Success: return 200
{ org_name: string (*URL format)*,
org_URL: string,

org_privacy_url: string,
logo_image_URL: string *(URL format)*,
survey_question_1: {hash},
survey_question_2: {hash},
whitelabeled: boolean,
rtv_ask_email_opt_in: boolean,
partner_ask_email_opt_in: boolean,
rtv_ask_sms_opt_in: bollean,
partner_ask_sms_opt_in: boolean,
rtv_ask_volunteer: boolean,
partner_ask_volunteer: boolean
}

In addition to the return field names above, there are return fields that are aliases for the above, provided for internal use with Rocky. Though present in return data, the alias fields can be ignored by most callers. They are:

- organization = org_name
- url = org_URL
- privacy_url = org_privacy_url
- rtv_email_opt_in = rtv_ask_email_opt_in
- partner_email_opt_in = partner_ask_email_opt_in
- rtv_sms_opt_in = rtv_ask_email_opt_in
- partner_sms_opt_in = partner_ask_sms_opt_in