**Middle-Tier RESTful API Documentation**

**Version** 1.0

16 April 2015

***The Veni VA Checkin System***

Advanced Software Engineering Project

SE 6387.M01



*Anant Kambli*

*Brian MacKay*

*Raleigh Murráy*

*Shahed Shuman*

*Kathryn Whitmire*

# Change Log

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Changed By | Description |
| 2-Apr-2015 | 0.9 | BM | First Draft |
| 11-Apr-2015 | 1.0 | BM | First Release |
| 22-Apr-2015 | 1.1 | BM | Tweaks, corrections, etc. |

Contents

[Change Log 2](#_Toc416121704)

[Overview 4](#_Toc416121705)

[API Verbs 4](#_Toc416121706)

[Query Strings 4](#_Toc416121707)

[HTTP Headers 5](#_Toc416121708)

[Payload Format 5](#_Toc416121709)

[Resources and Resource Operations 5](#_Toc416121710)

[Current Users Collection Resource 5](#_Toc416121711)

[Operation: POST 5](#_Toc416121712)

[Operation: GET 6](#_Toc416121713)

[Operation: DELETE 6](#_Toc416121714)

[NewUsers Resource Collection 7](#_Toc416121715)

[Operation: POST 7](#_Toc416121716)

[Users Resource Collection 7](#_Toc416121717)

[Operation: PUT 7](#_Toc416121718)

[VaFacilities Resource Collection 8](#_Toc416121719)

[Operation: GET (collection) 8](#_Toc416121720)

[Operation: GET (element) 8](#_Toc416121721)

[FacilityStates Resource Collection 9](#_Toc416121722)

[Operation: GET (collection) 9](#_Toc416121723)

[Appointments Resource Collection 9](#_Toc416121724)

[Operation: GET (collection) 9](#_Toc416121725)

[Operation: GET (element) 10](#_Toc416121726)

[Checkins Resource Collection 11](#_Toc416121727)

[Operation: POST 11](#_Toc416121728)

[Directions Resource Collections 11](#_Toc416121729)

[Operation: GET 11](#_Toc416121730)

# Overview

The Veni Middle Tier API uses REST Architectural Style. The information on the middle tier of the Veni System is represented as a set of *collection resources* that contain *resource elements*. Each of the collection resources is represented by a world-wide web-style URI that uses the same domain and the same first level URI part following the domain name. For example, the currentUsers resource is represented by <http://www.somedomain.com/veniapi/currentUsers>.

Element resources within a collection resource are identified by an “index” in the URI, following the collection resource name. The index may be a simple integer or a string. For example, to get information about the current user, do a GET, passing the “user token” returned from the logon operation to the currentUsers resource like this: [http://www.somedomain.com/veniapi/currentUsers/*[usertoken*](http://www.somedomain.com/veniapi/currentUsers/%5busertoken)*]*.

## API Verbs

Consistent with the REST Architectural Style, the Veni Middle Tier API uses the standard HTTP verbs and status codes for its operations. In particular:

|  |  |  |  |
| --- | --- | --- | --- |
| Verb | Action | Returned Status Codes | Notes |
| GET | Retrieve the requested collection or element | 200 – Success  404 – Not found  401 – Unauthorized | Safe – no side effects (nullipotent) |
| POST | Insert an element | 201 – Created (success)  401 – Unauthorized  403 – Forbidden (authentication failure)  409 – Conflict (duplicate key) | Has side effects, should only be called once |
| PUT | Update an element | 202 – Accepted (success)  401 – Unauthorized  404 – Not Found (the element to update does not exist) | Idempotent – no side effects on subsequent calls |
| DELETE | Delete an element | 204 – No Content (success)  401 – Unauthorized | Idempotent – will succeed even if the element has already been deleted |

In addition to the status codes listed above, all calls to the Veni Middle Tier API may return one of the following status codes at any time:

* 400 – Bad Request: the request could not be properly recognized (format, syntax, etc.)
* 408 – Request Timeout: the userToken has expired and the client is required to re-authenticate (not applicable to POSTs to the currentUsers collection)
* 500 – Internal Server Error: an unexpected server-side error occurred

## Query Strings

Some GET operations allow query strings. A query string may be used to filter the response from a GET operation or to “shape” the response payload (for example, describing that the get should not just return a summary of the collection, but the details of the members of the collection).

## HTTP Headers

The Veni Middle Tier API uses the HTTP headers to transfer “metadata” during requests and responses. In particular:

* Location: On POSTs and PUTs, returns the URI to the inserted or updated data in the response
* X-Token: On most requests, a User Token is required to authenticate the user
* X-[xxxxx]: On some calls, hyperlinks are provided to the client on responses (as documented below). These come as customized headers (e.g. X-Directions)

## Payload Format – Header Specification

Veni Middle Tier clients can control the payload format for both requests and responses by using the ACCEPTS: and CONTENT-TYPE: header fields on a request. All data passed back and forth should use the *JavaScript Object Notation* (or JSON). In headers this is identified as:

* **application/json**The data is transferred using the JavaScript Object Notation (or JSON)

### GET Request Headers:

On GET Requests, the request header should include and ACCEPTS: header specifying the format of the response (i.e., application/json):

* Request
  + Accepts: application/json

The Response to a GET will include a CONTENT-TYPE: header specifying a json format:

* Response
  + Content-Type: application/json

### POST and PUT Request Headers:

POST and PUT requests should include both ACCEPTS: and CONTENT-TYPE: headers specifying the JSON format. The response will include a CONTENT-TYPE: header where appropriate.

* Request
  + Accepts: application/json
  + Content-Type: application/json
* Response
  + Content-Type: application/json

## Authentication Tokens – X-Token Headers

Most requests require an authentication token (known as a “UserToken” or, in HTTP headers, an “X-Token”). Authentication tokens are generated and returned from POSTs to either the NewUsers or CurrentUsers collections. The following operations do not require an X-Token in the request header:

* And of the operations related to the CurrentUsers collection resource (POST, GET, DELETE).   
  A POST operation creates an X-Token. The other operations require a UserToken in the URI.
* Any of the GET operation from the VaFacilities collection resource  
  These operations return benign information. In addition, that information is required \*before\* a user can be created.
* A GET from the FacilityStates resource collection  
  Also benign, also needed before a user can be created.

## Returning Error Information

In addition to error statuses returned with an HTTP response, some calls return additional “ApiError” information on specific errors. This information is returned as a JSON packet, as an array of strings:

[errorString1, errorString2, … errorStingN]

# Resources and Resource Operations

The next sections document the operations on the various Collection Resources and their elements.

## NewUsers Resource Collection

The only operation available for the newUsers resource collection is a POST (insert). In addition, the POST to newUsers is how a new user is created. As a result, no authentication token is required for this operation.

### Operation: POST

* URI: <http://www.domainname.com/veniapi/newUsers>
* Response statuses:
  + 201 – Created (success)
  + 400 – Bad Request: there is mal-formed data in the request (for example the VetId).   
     An ApiErrors collection is returned
  + 404 – Not Found: The VA does not recognize the user information as a veteran
  + 409 – Conflict: likely the result of trying to create a user with an existing VetId   
     An ApiErrors collection is returned
* Request Payload:  
  The request payload is very similar to the normal user’s payload. The only difference is the presence of the user’s chosen password.

{ VetId: string,  
 FirstName: string,  
 LastName: string,  
 EmailAddress: string,  
 Password: string,  
 FacilityIds: Array [ int ]   
}

* Response Payload:  
  The response payload is the same as that returned from a POST or a GET to currentUsers. The effect of this post is to create the new user and to log him/her on. The response payload will include the full definition of the User object for the authenticated user (see the GET documentation for the currentUsers collection (above)). One of the fields will include the UserToken which must be included in the header of all requests to resources other than currentUsers and newUsers
* Headers:  
  The response header will include:
  + Location: a URI (including the UserToken) to get the information for the current user (a GET to the currentUsers collection).
  + X-Token: the logged on user’s UserToken

NOTE: no X-Token entry in the header is required for the request message. The UserToken in the returned payload suffices.

## Current Users Collection Resource

The Current Users collection represents the logged on users. A user cannot see information about other logged on users, and can only see information about him/herself. A POST (insert into) the Current User collection is the mechanism a client uses to authenticate to the system.

The result of an authentication (a POST to currentUsers) is a UserToken. The client should

### Operation: POST

* URI: <http://www.domainname.com/veniapi/currentUsers>
* Response statuses:
  + 201 – Created (success)
  + 403 – Forbidden – authentication failure (username and/or password incorrect)
* Request Payload:  
  The request payload contains the user’s credentials. Currently, the only credentials accepted are the username and the password (both strings)

{ EmailAddress: string,

Password: string

}

* Response Payload:  
  The response payload will include the full definition of the User object for the authenticated user (see the GET documentation below). One of the fields will include the UserToken which must be included in the header of all requests to resources other than currentUsers
* Headers:  
  The response header will include:
  + Location: a URI (including the UserToken) to get the information for the current user

NOTE: no X-Token entry in the header is required for currentUsers collection operations.

### Operation: GET

* URI: [http://www.domainname.com/veniapi/currentUsers/*[UserToken]*](http://www.domainname.com/veniapi/currentUsers/%5bUserToken%5d)  
  where [UserToken] is an authentication token string returned from a POST operation to either the NewUsers or CurrentUsers resource collection. The client should treat that token as opaque information – there is no client usable information in the token.
* Response statuses:
  + 200 – Success
  + 404 – Not found – indicates that the user token is incorrect
* Request Payload: None
* Response Payload: An object with the full definition of the current user (see below). One of the fields is the UserToken
* Query Strings: None

NOTE: no X-Token entry in the header is required for currentUsers collection operations. The UserToken in the returned payload suffices.

#### Response Payload:

{ VetID: string, //10 characters, numeric only  
 FirstName: string,  
 LastName: string,  
 EmailAddress: string,  
 UserToken: string,  
 FacilityIds: Array [ int ]   
}

### Operation: DELETE

A delete of a currentUsers element is the equivalent of a “log-off” operation

* URI: [http://www.domainname.com/veniapi/currentUsers/*[UserToken]*](http://www.domainname.com/veniapi/currentUsers/%5bUserToken%5d)  
  where [UserToken] is a string returned from a POST operation. The client should treat that token as opaque information – there is no client usable information in the token.
* Response statuses:
  + 204 – No Content (success)  
    Note: because the DELETE operation is idempotent, deleting an already deleted currentUser (effectively, a user logging off more than once) or passing an unknown UserToken will always silently succeed.
* Request Payload: None
* Response Payload: None

NOTE: no X-Token entry in the header is required for currentUsers collection operations. The UserToken in the returned payload suffices.

### Operation: PUT

A PUT operation allows a user to update his information. Only information about the current user can be changed.

During an update, every field in the object definition below must be included in the payload, however fields that have not changed can be set to NULL.

* URI: [http://www.domainname.com/veniapi/currentUsers/*[UserToken]*](http://www.domainname.com/veniapi/currentUsers/%5bUserToken%5d)
* Response statuses:
  + 202 – Accepted (success)
  + 401 – Not authorized – the user token is missing or invalid
* Header (request):
  + X-Token: a UserToken that represents the currently logged on user
* Header (response):
  + Location: a URI for a GET of the CurrentUsers collection for the current user (after the updates have been applied
* Request Payload: Information about the user. Not every field is changeable (for example, the VetId cannot be changed, and changing passwords is a separate operation) Any empty fields in the request payload will be ignored during the update:

{ FirstName: string,  
 LastName: string,  
 EmailAddress: string,  
 FacilityIds: Array [ int ]   
}

* Response Payload:   
  an object representing the current user after the changes have been applied (see above)

## Passwords Resource Collection

A veteran can send a PUT operation to the Passwords collection to change his/her password.

### Operation: PUT

* URI: [http://www.domainname.com/veniapi/passwords/*[UserToken]*](http://www.domainname.com/veniapi/passwords/%5bUserToken%5d)
* Response statuses:
  + 202 – Accepted (success)
  + 400 – Bad Request – the username or (old) password is incorrect
  + 401 – Not authorized – the user token is missing or invalid
* Header (request):
  + X-Token: a UserToken that represents the currently logged on user
* Header (response): no special response headers
* Request Payload: None
* Response Payload:   
  an object representing the current user (see above)

## VaFacilities Resource Collection

Only GET operations are available from the vaFacilities collection. Inserts (POSTs) and updates (PUTs) have no meaning. Two get operations are possible, one for the collection (which returns a list of facilities (summary info only)) and one for a particular element.

### Operation: GET (collection)

* URI: <http://www.domainname.com/veniapi/vaFacilities>
* Response statuses:
  + 200 – Success
  + 401 – Not authorized – the user token is missing or invalid
  + 404 – Not Found – only applicable if State query string is used and either state is incorrect or there are no VA facilities in the state
* Header (request):
  + X-Token: a UserToken that represents the currently logged on user
* Request Payload: None
* Response Payload: An array of facility summary objects as described below.
* Query Strings:
  + ?State=TwoCharacterStateCode  
    Optional, if present, only those facilities in the queried state are included   
    example: ?State=TX

#### Response Payload:

Array [ { FacilityId: int,  
 FacilityName: string,  
 City: string,  
 State: string  
 } ]

### Operation: GET (element)

* URI: [http://www.domainname.com/veniapi/vaFacilities/*[facilityId]*](http://www.domainname.com/veniapi/vaFacilities/%5bfacilityId%5d)   
  where [facilityId] is an integer returned from a GET from the vaFacilities collection or from a GET for an Appointment (or any other mention of a facility). The client should treat that token as opaque information – there is no client usable information in the token.
* Response statuses:
  + 200 – Success
  + 401 – Not authorized – the user token is missing or invalid
  + 404 – Not found – indicates that the facility code is incorrect
* Header (request):
  + X-Token: a UserToken that represents the currently logged on user
* Request Payload: None
* Response Payload: A facility detail object as described below.
* Query Strings: None

#### Response Payload:

{ FacilityId: int,  
 FacilityName: string,  
 Address: string,  
 PhoneNumber1: string,  
 PhoneNumber2: string,  
 City: string,  
 State: string,  
 Latitude: number (floating point),  
 Longitude: number (floating point)   
 }

NOTE: if the latitude and longitude values are not known, they are returned set equal to 0.0.

## FacilityStates Resource Collection

Only a general GET operation is available from the FacilityStates collection. The result is an array of two character strings, each a state code (like “TX”). Note that US territories like Puerto Rico (State Code = “PR”) and Guam (State Code = “GU”) are included.

### Operation: GET (collection)

* URI: http://www.domainname.com/veniapi/facilityStates
* Response statuses:
  + 200 – Success
* Header (request):
  + X-Token: a UserToken that represents the currently logged on user
* Request Payload: None
* Response Payload: An array of two-character state codes
* Query Strings: None

#### Response Payload:

Array [ Two-Character-State-Code-strings ]

## Appointments Resource Collection

Only GET operations are available from the appointments collection. Inserts (POSTs) and updates (PUTs) have no meaning. Two get operations are possible, one for the collection and one for a particular element. Both operations return objects of the same structure, one is an array of objects, the other is a single object. Only appointments for the authenticated user are available.

Every time a veteran performs a GET for an appointment (or set of appointments), the VistA system is checked to see if there are any new appointments. Newly downloaded appointments are marked “New” with a Boolean flag.

### Operation: GET (collection)

* URI: http://www.domainname.com/veniapi/appointments
* Response statuses:
  + 200 – Success
  + 401 – Not authorized – the user token is missing or invalid
* Header (request):
  + X-Token: a UserToken that represents the currently logged on user
* Request Payload: None
* Response Payload: An array of appointment objects as described below.
* Query Strings:
  + ?Date=*[Date in RFC-3339 format]*If present, only appointments for that date are returned. Cannot be combined with StartDate or EndDate
  + ?BeginDate=*[Date in RFC-3339 format]*If present, only appoints on or after that date are returned. May be combined with EndDate, cannot be combined with Date
  + ?EndDate=*[Date in RFC-3339 format]*If present, only appoints on or before that date are returned. May be combined with StartDate, cannot be combined with Date

#### Response Payload:

Array [ { *[appointment structure defined below]*  
 } ]

### Operation: GET (element)

* URI: [http://www.domainname.com/veniapi/vaFacilities/*[facilityInt]*](http://www.domainname.com/veniapi/vaFacilities/%5bfacilityInt%5d)   
  where [facilityInt] is an integer returned from a GET from the vaFacilities collection or from a GET for an Appointment (or any other mention of a facility). The client should treat that token as opaque information – there is no client usable information in the token.
* Response statuses:
  + 200 – Success
  + 401 – Not authorized – the user token is missing or invalid
  + 404 – Not found – indicates that the facility code is incorrect
* Header (request):
  + X-Token: a UserToken that represents the currently logged on user
* Request Payload: None
* Response Payload: A facility detail object as described below.
* Query Strings: None

#### Response Payload:

{ AppointmentCode: string,  
 FacilityId: int,  
 FacilityName: string,  
 AppointmentTime: [DateTime in RFC-3339 format],  
 Practitioner: string,  
 Specialty: string,  
 BuildingName: string,  
 RoomNumber: string,  
 Status: [one of “Confirmed”, “Cancelled”, “Onhold”, “Checkedin”] ,  
 New: Boolean //true if this was downloaded from VistA as a result of this GET  
 }

## CalendarItems Resource Collection

The CalendarItems resource collection differs from all of the others. Rather than returning JSON data, it returns an iCalendar entry (with a *text/calendar* mime-type).

Only a singleton of a CalendarItem can be done. The item specifier on the GET operation is an AppointmentCode.

### Operation: GET (element)

* URI: [http://www.domainname.com/veniapi/calendarItems/*[appointmentCode]*](http://www.domainname.com/veniapi/calendarItems/%5bappointmentCode%5d)   
  where [appointmentCode] is a string returned from a GET from the Appointments collection. The client should treat that token as opaque information – there is no client usable information in the token.
* Response statuses:
  + 200 – Success
  + 401 – Not authorized – the user token is missing or invalid
  + 404 – Not found – indicates that the appointment code is incorrect
* Header (request):
  + X-Token: a UserToken that represents the currently logged on user
* Request Payload: None
* Response Payload: An iCalendar formatted response that represents the appointment.
* Query Strings: None

#### Sample iCalendar Response:

BEGIN:VCALENDAR

VERSION:2.0

PRODID:-//Veni Checkin//Appointment//EN

BEGIN:VEVENT

UID:4dbe780363a622fc161e0bd79ff95320

DTSTAMP:20150414T182830Z

DTSTART:20150502T131500Z

DTEND: 20150502T140000Z

SUMMARY:Appointment with Jay Smith, MD at Dallas Veterans Administration Hospital

LOCATION:Franklin Building, Room: 528

END:VEVENT

END:VCALENDAR

## Checkins Resource Collection

The Checkins collection only supports a POST operations. Checkins are tightly tied to appointments. Information about a checkin can be retrieve via a GET to the Appointments collection.

### Operation: POST

* URI: [http://www.domainname.com/veniapi/checkins/*[appointmentCode]*](http://www.domainname.com/veniapi/checkins/%5bappointmentCode%5d)The appointment code is the code returned from a GET of the appointments collection
* Response statuses:
  + 201 – Created (success)
  + 403 – Forbidden – authentication failure (username or password incorrect)
  + 404 – Not found – indicates that the appointment code is incorrect
* Header (request):
  + X-Token: a UserToken that represents the currently logged on user
* Request Payload:  
  An appointment checkin object as described below.
* Response Payload: None
* Headers:  
  The response header will include:
  + Location: a URI that specifies a GET from the appointments collection for the appointment referred to in this POST. The status will have been updated
  + X-Directions: a URI to a GET from the Directions collection that will return directions (within the facility) for this appointment

#### Checkins POST Payload:

Checkin: { VeteranId: string,  
 AppointmentCode: string,  
 Latitude: number,  
 Longitude: number  
 }

## Directions Resource Collections

The directions collection only supports a GET operation. Neither an insert (POST) nor an update (PUT) make any sense. A directions element represents the directions to an appointment *within* the facility

### Operation: GET

* URI: [http://www.domainname.com/veniapi/directions/*[appointmentCode]*](http://www.domainname.com/veniapi/directions/%5bappointmentCode%5d)
* Response statuses:
  + 200 – Success
  + 401 – Not authorized – the user token is missing or invalid
  + 404 – Not found – appointment code not found
* Header (request):
  + X-Token: a UserToken that represents the currently logged on user
* Request Payload: None
* Response Payload: An (ordered) array of strings, each one part of the directions
* Query Strings: None