**Middle-Tier RESTful API Documentation**

**Version** 1.0

04 February 2015

***The Veni VA Checkin System***

***Draft – Not for Construction***

Advanced Software Engineering Project

SE 6387.M01



*Anant Kambli*

*Brian MacKay*

*Raleigh Murráy*

*Shahed Shuman*

*Kathryn Whitmire*

# Change Log

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Changed By | Description |
| 1-Apr-2015 | 0.1 | BM | First Draft |
|  |  |  |  |
|  |  |  |  |

Contents

[Change Log 2](#_Toc415260959)

[Overview 4](#_Toc415260960)

[API Verbs 4](#_Toc415260961)

[HTTP Headers 4](#_Toc415260962)

[Payload Format 5](#_Toc415260963)

[Resources and Resource Operations 5](#_Toc415260964)

[Current Users Collection Resource 5](#_Toc415260965)

[NewUsers Resource Collection 6](#_Toc415260966)

[VistaMappings Resource Collection 6](#_Toc415260967)

[Facilities Resource Collection 6](#_Toc415260968)

[Appointments Resource Collection 6](#_Toc415260969)

[AppointmentStatuses Resource Collection 6](#_Toc415260970)

[Checkins Resource Collection 6](#_Toc415260971)

[Directions Resource Collections 6](#_Toc415260972)

# 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) | Has side effects, should only be called once |
| PUT | Update an element | 202 – Accepted (success)  401 - Unauthorized | 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

## 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-UserToken: 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

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. There are two possibilities for the payload format on either a request or response:

* **application/xml**The data is transferred using “Plain old XML” (or POX)
* **application/json**The data is transferred using the JavaScript Object Notation (or JSON)

In general, the expectation is that clients will be using JSON. Most testing will be done using JSON format for both requests and responses.

# Resources and Resource Operations

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

## 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 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)
* 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-UserToken entry in the header is required for currentUser collection operations.

### Operation: GET

* 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:
  + 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

NOTE: no X-UserToken entry in the header is required for currentUser collection operations.

#### Current User Object Format

To be determined.

## NewUsers Resource Collection

## VistaMappings Resource Collection

## Facilities Resource Collection

## Appointments Resource Collection

## AppointmentStatuses Resource Collection

## Checkins Resource Collection

## Directions Resource Collections