Chris Long

[Email address]

SAVIOR System Interface Details

Version 1, April 2018

# Introduction

This document describes the external interfaces to the SAVIOR system.

# Client Interfaces

SAVIOR is designed to interact with three client applications: the Admin Workbench, Dashboard, and Desktop.

The Admin Workbench and Dashboard are web applications. They interact with rest of the SAVIOR system using the Virtue Management API (see §3).

The Desktop is a Java application running in a typical desktop environment (e.g., Windows, Linux). It also uses the Virtue Management API. In addition, it connects to Linux application VMs (running as Xen DomUs) via ssh and xpra[[1]](#footnote-1) (tunneled over ssh) and Windows application VMs (running as EC2 instances) via ssh and RDP.

# Virtue Management API

The Virtue management API implmented in SAVIOR is based on the API published by APL. The current version of SAVIOR does not implement the entire API as specified by APL.

The SAVIOR API also has a structural difference from the APL API, in the relationship between applications and Virtues. In the APL API, Virtues directly contain applications. However, in SAVIOR, Virtues contain virtual machines that contain applications.

The Virtue Manager (the “virtue-admin” module in the source code) provides a REST service implementation of the Virtue APIs. The following sections describe the supported calls. Unless otherwise indicated, parameters are passed as part of the URL and denoted by curly braces ({}).

In test mode, user authentication is not enforced. When authentication is enforced, all API calls require a user authentication token set as the cookie “JSESSIONID”, except for User Login, which returns the token when successful (see §3.1.3).

Currently supported operating systems are Linux and Windows, output as “LINUX” and “WINDOWS”.

The Date type is a string in ISO 8601 date and time format (e.g., “2018-04-16T01:22:07.455+0000”).

## Virtue (User) API

### Get Application

**URL**: get /user/application/{id}

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Description |
| id | true | string | Application ID |

Output

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| id | string | Application ID (same as the input) |
| launchCommand | string | Command to start the application |
| os | string | Operating system |
| name | string | Application name (human readable) |
| version | string | Application version (always “1.0” in current SAVIOR test data) |
| iconKey | String | Key to represent the icon to be displayed |

Error Conditions

http 204 “No content” returned if the id is not valid

### Get Role

**URL**: get /user/virtue/template/{templateId}

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Description |
| templateId | true | string | Virtue Template (aka Role) ID |

Output

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| applicationIDs | string[] | IDs of applications allowed for this Virtue Template |
| awsTemplateName | string | no longer used |
| color | string | Color used for the virtue in the desktop app and workbench |
| enabled | boolean | Whether the Virtue Template can be instantiated |
| fileSystemIds | string[] | IDs of file systems that should be attached to VMs in this Virtue Template |
| id | string | Virtue Template ID (same as templateId parameter) |
| lastEditor | string | The last person or entity that modified the Virtue Template |
| lastModification | Date | When the Virtue Template was last changed |
| name | string | Virtue Template name (human readable) |
| userCreatedBy | String | The person or entity that created the Virtue Template |
| timeCreatedAt | Date | When the Virtue Template was created |
| version | string | Virtue Template version (always “1.0” in current SAVIOR test data) |
| vmTemplateIds | string[] | IDs of virtual machine templates that are part of this Virtue Template |

Error Conditions

http 400 “Bad request” if the ID is not valid

### User Login

**URL**: post /login

Parameters

Passed as form data

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Description |
| username | true | string | User name |
| password | true | string | User password |
| forceLogoutOfOtherSessions | false | boolean | Not implemented |

Output

The output is as a cookie.

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| JSESSIONID | string | Hex string authentication token |

Error Conditions

None

### User Logout

**URL**: get /logout

Parameters

None

Output

None

Error Conditions

None

### List User Roles

**URL**: get /user/virtue/template

Parameters

None

Output

An array of Virtue Template objects. See §3.1.2 for the definition of the array elements.

Error Conditions

None

### List User Virtues

**URL**: get /user/virtue

Parameters

None

Output

An array of Virtue objects. See §3.1.7 for the definition of the array elements.

Error Conditions

None

### Get Virtue Information

**URL**: get /user/virtue/{virtueId}

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Description |
| virtueId | virtueId | string | ID of a Virtue |

Output

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| applicationIds | string[] | IDs of applications that can run in this Virtue |
| id | string | Virtue ID |
| name | string | Virtue name |
| state | string | One of: UNPROVISIONED, CREATING, STOPPED, LAUNCHING, RUNNING, PAUSING, PAUSED, RESUMING, STOPPING, DELETING |
| templateId | string | Virtue Template ID this Virtue was instantiated from |
| username | string | Name of current user |
| virtualMachineIds | string[] | IDs of virtual machines that can run in this Virtue |

Error Conditions

http 400 “Bad Request” returned if the id is not valid

### Create Virtue

**URL**: post /user/virtue/template/{virtueTemplateId}

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Description |
| virtueTemplateId | true | string | Virtue Template ID to instantiate |

Output

A Virtue. See §3.1.7 for the definition.

Error Conditions

http 400 “Bad request” if the ID is not valid

### Launch a Virtue

**URL**: post /user/virtue/{virtueId}

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Description |
| virtueId | true | string | Virtue ID to launch |

Output

None

Error Conditions

http 400 “Bad request” if the ID is not valid

### Stop a Running Virtue

**URL**: post /user/virtue/{virtueId}/stop

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Description |
| virtueId | true | string | Virtue ID to launch |

Output

None

Error Conditions

http 400 “Bad request” if the ID is not valid

### Destroy a Virtue

**URL**: delete /user/virtue/{virtueId}

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Description |
| virtueId | virtueId | string | ID of a Virtue |

Output

None

Error Conditions

http 400 “Bad Request” returned if the id is not valid

### Launch a Virtue Application

**URL**: post /user/virtue/{virtueId}/{applicationId}

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Description |
| virtueId | true | string | ID of the Virtue the application is to run in |
| applicationId | true | string | Application ID to start |

Output

None

Error Conditions

http 400 “Bad request” if either ID is not valid, or if the specified Virtue does not include the specified application

### Stop a Running Virtue Application

Not yet implemented.

## Virtue Administrative API

### List Applications

**URL**: get /admin/application

Parameters

None

Output

An array of Applications. See §3.1.1 for the definition of the array elements.

Error Conditions

None

### Get Applications

**URL**: Get /admin/application/{applicationId}

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Description |
| applicationId | true | string | ID of an application |

Output

A full object for the Applications. See §3.1.1 for the definition of the array elements.

### Create application

**URL**: Post /admin/application

Header ‘content-type’ must be set as application/json

Post Body:

JSON application. See §3.1.1 for the definition.

Output

The full object of the new Application. See §3.1.1 for the definition.

### Update application

**URL**: Post /admin/application/{ applicationId }

Header ‘content-type’ must be set as application/json

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Description |
| applicationId | true | string | ID of an application |

Post Body:

JSON application. See §3.1.1 for the definition.

Output

The full object of the new Application. See §3.1.1 for the definition.

### Delete application

**URL**: Post /admin/application/{ applicationId }

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Description |
| applicationId | true | string | ID of an application |

Output

None

### Get Resource

Not yet implemented

### List resources

Not yet implemented

### Attach resource to Virtue

Not yet implemented

### Detach resource from Virtue

Not yet implemented

### Create Role

**URL**: post /admin/virtue/template

Header ‘content-type’ must be set as application/json

Post Body:

JSON object with the following parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| applicationIDs | string[] | IDs of applications allowed for this Virtue Template |
| enabled | boolean | Whether the Virtue Template can be instantiated |
| name | string | Virtue Template name (human readable) |
| version | string | Virtue Template version (always “1.0” in current SAVIOR test data) |
| virtualMachineTemplateIds | string[] | IDs of virtual machine templates that are part of this Virtue Template |

Output

The full object of the new Virtue Template. See §3.1.2 for the definition of the element.

Error Conditions

http 400 “Bad request” if any of the IDs are not valid.

### List Role

**URL**: get /admin/virtue/template

Parameters

None

Output

An array of Virtue Templates. See §3.1.2 for the definition of the array elements.

Error Conditions

None

### Export Virtue system

**URL**: get /admin/export

Parameters

None

Output

System specific JSON document containing database information.

Error Conditions

None

### Import Virtue system

**URL**: POST /admin/import

Note: this is generally intended only for importing data into empty databases.

Parameters

Post body: System specific JSON from export.

Output

None

Error Conditions

None

### Import Testing User

**URL**: Get /admin/import/user/{name}

Output:

Returns a username as string.

**Valid Names:** admin, alice, bob

### Import Testing Application

**URL**: Get /admin/import/application/{name}

Output:

Returns the id of the application as string.

**Valid Names:**

* linux\_calc
* linux\_chrome
* linux\_firefox
* linux\_lo\_calc
* linux\_lo\_draw
* linux\_lo\_impress
* linux\_lo\_writer
* linux\_terminal
* linux\_thunderbird
* windows\_chrome
* windows\_cmd
* windows\_edge
* windows\_excel
* windows\_outlook
* windows\_powershell
* windows\_skype
* windows\_word

### Import Testing Virtual Machine

**URL**: Get /admin/import/ virtualmachines/{name}

Output:

Returns the id of the virtual machine as string.

**Valid Names:**

* external\_internet\_consumer
* linux\_all
* linux\_corporate\_email\_user
* linux\_power\_user
* linux\_router\_admin
* linux\_lo\_impress
* windows\_all
* windows\_corporate\_email\_user
* windows\_document\_editor
* windows\_power\_user

### Import Testing Role

**URL**: Get /admin/import/ virtue/{name}

Output:

Returns the id of the virtue template as string.

**Valid Names:**

* document\_editor
* external\_internet\_consumer
* linux\_corporate\_email\_user
* linux\_test\_devel
* linux\_text\_devel2
* power\_user
* router\_admin
* linux\_lo\_impress
* windows\_corporate\_email\_user
* windows\_test\_devel
* windows\_text\_devel2

### List Users

**URL**: /admin/user

Parameters

None

Output

An array of user information. See §3.2.14 for the definition of the array elements.

### Get User

**URL**: /admin/user/{username}

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Description |
| username | true | string | Name of an existing user |

Output

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| authorities | string[] | One or more of: ROLE\_USER, ROLE\_ADMIN |
| username | string | Name of the user |
| virtueTemplateIds | string[] | IDs of Virtue Templates the user is authorized to instantiate |

Error Conditions

http 400 “Bad request” returned if the username is not valid

### List User Virtues

Not yet implemented.

### Authorize Role for User

**URL**: Post /admin/user/{username}/assign/{templateId}

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Description |
| username | true | string | Name of an existing user |
| templateId | true | string | Virtue Template (aka Role) ID |

Output

None

Error Conditions

http 400 “Bad request” if username or templateId is not valid

### Unauthorize Role for User

**URL**: Post /admin/user/{username}/revoke/{templateId}

Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required | Type | Description |
| username | true | string | Name of an existing user |
| templateId | true | string | Virtue Template (aka Role) ID |

Output

None

Error Conditions

http 400 “Bad request” if username or templateId is not valid

### List UserTokens

**URL**: Get /admin/session

Output

Returns a map of usernames to lists of session IDs

### Force Logout User

**URL**: Get /admin/user/{username}/logout

Output

None

# Sense and Response API

The Sense and Response API is described in its own document.

1. https://xpra.org/ [↑](#footnote-ref-1)