JOHNS HOPKINS
APPLIED PHYSICS LABORATORY

11100 Johns Hopkins Road • Laurel, Maryland 20723-6099

AOS Report No. AOS-17-0228

September 2017

VIRTUE TESTBED SOFTWARE APPLICATION PROGRAMMING INTERFACE (API)

Version 1.1

Prepared for: Intelligence Advanced Research Projects Activity (IARPA)

Prepared by: VirtUE Team

Task No.: CVI01
Contract No.: 2012-12050800010

This Page Intentionally Left Blank

Table of Contents

1. Introduction

JHU/APL is providing support to the Virtuous User Environment (VirtUE) program sponsored by the Intelligence Advanced Research Projects Activity (IARPA). JHU/APL's main efforts are focused along three technical areas comprising systems engineering, software engineering, and testing and evaluation (T&E). As part of JHU/APL's T&E support, JHU/APL is responsible for developing a testbed to support test and evaluation of performer solutions. This software application programming interface (API) document is intended to identify software interfaces that performer solutions are expected to support to enable their integration with JHU/APL's testbed.

2. Overview

2.1. Background

VirtUE seeks to leverage the federal government's impending migration to commercial cloud-based information Technology (IT) infrastructures and the current explosion of new virtualization and operating system (OS) concepts to create and demonstrate a more secure interactive user computing environment (UCE) than is currently available for government use or is expected to be available in the near future. [1]

- In Phase 1, VirtUE seeks to deliver an interactive UCE designed from the outset to be a more secure, capable sensor and defender in the cloud environment than the current government UCE solution. To be acceptable to potential government consumers, the new UCE must still offer functionality and performance characteristics comparable to the current government UCE.
- In Phase 2, performers shall take the technologies and/or concepts developed in Phase 1 and create novel external analytics and security controls that leverage them. The purpose of this analytics/control effort is to create dynamic detection and protection capabilities that make the Virtue user environment more resistant to attacks expected in the commercial cloud while minimizing the costs associated with these capabilities.

2.2. Scope

This document describes the interface requirements which must be met in order for performer solutions to be integrated into the JHU/APL Virtue Testbed. These requirements have been developed so that the functionality and performance of each solution can be tested in a consistent and automated manner. If necessary, approved exceptions to these requirements can be granted and JHU/APL will work with the performer to customize the test integration for their solution.

The following figure shows a notional architecture of this integration. The testbed and interfaces are deliberately agnostic to the design of the performer systems. Performers are expected to provide command-line interface (CLI) executables that conform to the defined interfaces. These executables can then be used by the testbed to control and interact with the performer system to execute test cases.

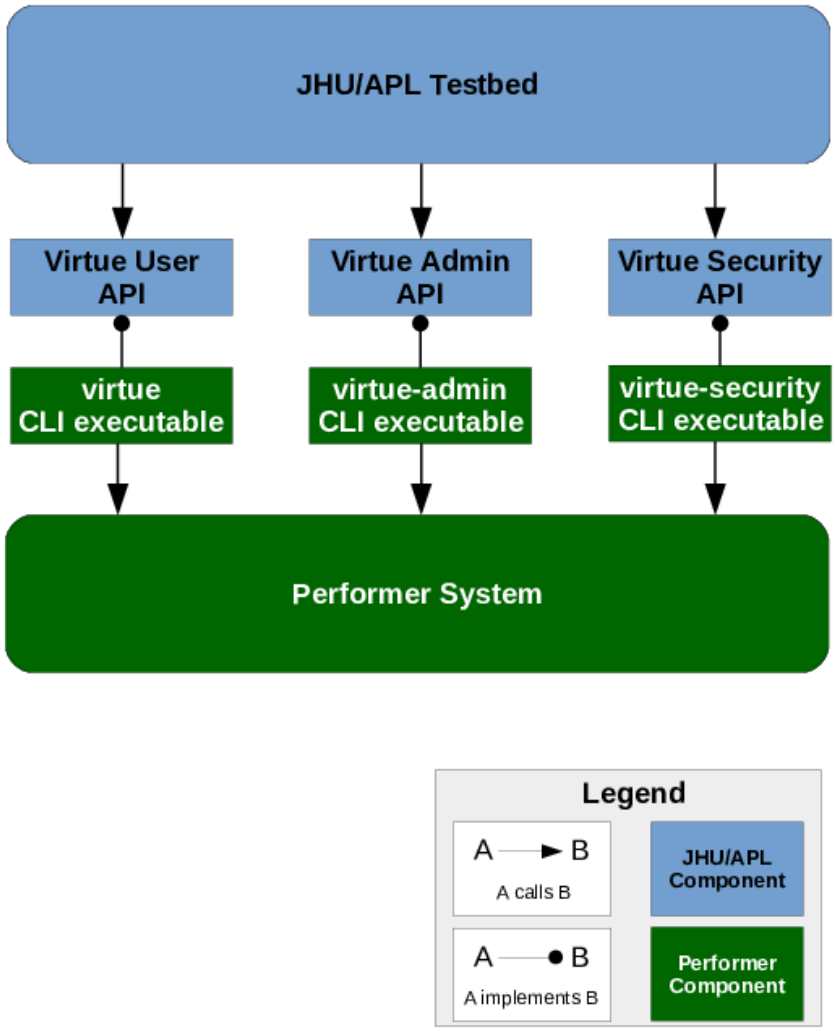

Figure 1. Notional Architecture

The APIs in this image are documented in later sections and describe the *minimum* interface for interacting with and controlling the Virtue system. Performers can, and should, implement additional functionality as needed for their solution.

2.3. Virtue APIs

This document describes three APIs:

- **Virtue API** or **Virtue User API**: this is the user-level interface for interacting with the Virtue system. It contains the functionality that a non-administrator user of Virtue would need, such as creating Virtues. It should be implemented in a `virtue` CLI executable.
- **Virtue Admin API**: this is the administrator-level interface for interacting with the Virtue system. It allows administrators to input the data and controls needed for users, such as Roles. It should be implemented in a `virtue-admin` CLI executable.
- **Virtue Security API**: this is the administrator-level interface for retrieving security

data and changing security profiles for Virtues. The API presented in this document is a reference example; performers are encouraged to develop their own API and work with JHU/APL on its integration. It should be implemented in a `virtue-security` CLI executable.

2.4. Virtue Eco-System Requirements

The **Virtue Eco-System** is the set of software and hardware components that work together to support a Virtue system instance. Requirements for this eco-system include:

- Solution must run on Amazon Web Services.
- Solution must have a command-line interface and a graphical user interface.
- The command-line interface must have a way to script calls and responses. Test cases will be developed that consist of scripts that call the APIs and measure the response, so there must be a way to input these scripts. One example is a bash shell accessible to the testbed. Another example would be a network-accessible socket interface.

3. Definition of Terms

- **CLUE**: the component that represents the CLient Ui (user interface) Endpoint. This system contains the CLI and GUI for users and administrators to interact with the system. It could be a hardware component like a pre-configured laptop, or a software component with the necessary installation and setup documentation.
- **Role**: a collection of applications and other resources needed for a user to perform a defined set of tasks.
- **Virtue**: the conceptual grouping of virtual instances (e.g., virtual machines, containers) and other components needed to instantiate a **Role** for a user. **Virtue** is also used to refer to the entire system, such as in the case of **Virtue API** or **Virtue Eco-System**.
- **VirtUE**: shorthand for Virtuous User Environment, VirtUE is used to describe the IARPA program. This is differentiated from **Virtue**, which is a concrete instantiation of the system as described above.

3.1. Object Schema Definitions

The following sections document objects that are used in the various APIs. Notionally, these objects are represented as JSON [2] when used as inputs and outputs for API calls.

3.1.1. Application

Description

Describes an application that can be run.

Type name: Application

Properties

Name	Required	Type	Description
id	false	string	The unique identifier for this Application. Format is implementation-specific. *Must be unique across all instances.*
name	true	string	The human-readable name for this Application. *Examples*: * `Firefox`
version	true	string	A version description. *Examples*: * `1.0` * `2017-05-04`

Name	Required	Type	Description
os	true	string (enum)	The operating system that the Application runs on. *Enum Values*: `[LINUX, WINDOWS]`

3.1.2. Resource

Description

Describes a resource that a Virtue can access.

Type name: Resource

Properties

Name	Required	Type	Description
id	false	string	The unique identifier for this Resource. Format is implementation-specific. *Must be unique across all instances.*
type	true	string (enum)	The type of the Resource. *Enum Values*: `[DRIVE, PRINTER]`
unc	true	string	The Universal Naming Convention (UNC) path to the resource.
credentials	false	object	Any credentials that are needed to access the resource.

3.1.3. Role

Description

Describes a role.

Type name: Role

Properties

Name	Required	Type	Description
id	false	string	The unique identifier for this Role. Format is implementation-specific. *Must be unique across all instances.*
name	true	string	The human-readable name for this Role. *Examples*: * `Database Administrator`

Name	Required	Type	Description
version	true	string	A version description. *Examples*: * `1.0` * `2017-05-04`
applicationIds	true	set of strings	The IDs of the Applications that are provided with this Role.
startingResourceIds	true	set of strings	The IDs of the Resources that Virtue instances from this Role can access when they are created. Format is implementation-specific.
startingTransducerIds	true	set of strings	The IDs of the Transducers that Virtue instances from this Role must have enabled when they are created. Format is implementation-specific.

3.1.4. Transducer

Description

Describes a security component that either passively monitors offline Virtue data (sensor) or actively monitors and can modify inline data (actuator).

Type name: Transducer

Properties

Name	Required	Type	Description
id	false	string	The unique identifier for this Transducer. Format is implementation-specific. *Must be unique across all instances.*
name	true	string	A human-readable name for this Transducer.
type	true	string (enum)	The type of Transducer. SENSORs passively monitor offline Virtue data; ACTUATORs actively monitor and can modify inline data. *Enum Values*: `[SENSOR, ACTUATOR]`
startEnabled	true	boolean	When a new Virtue is instantiated, this flag controls whether this Transducer should be enabled by default. This is in addition to the Transducers that are automatically enabled by the Virtue's Role specification.
startingConfiguration	true	object	When a new Virtue is instantiated, this is the starting configuration that should be applied. Format is Transducer-specific.

Name	Required	Type	Description
requiredAccess	false	set of strings (enum)	What access to underlying resources this Transducer needs. *Enum Values*: `[NETWORK, DISK, MEMORY]`

3.1.5. User

Description

Describes a user that can take actions in the system.

Type name: User

Properties

Name	Required	Type	Description
username	true	string	The username of this User. *Must be unique across all instances.*
authorizedRoleIds	true	set of strings	The IDs of all the Roles that this User is authorized to instantiate.

3.1.6. UserToken

Description

A unique token associated with a User login session.

Type name: UserToken

Properties

Name	Required	Type	Description
username	true	string	The username of the User for whom this token was generated.
token	true	string (uuid)	The unique, secret, generated token string. *Must be unique across all instances.*
expiration	true	string (date-time)	When this token expires.

3.1.7. Virtue

Description

Describes a single virtue.

Type name: Virtue

Properties

Name	Required	Type	Description
id	false	string	The unique identifier for this Virtue. Format is implementation-specific. *Must be unique across all instances.*
username	true	string	The username for the User to whom this Virtue belongs.
roleId	true	string	The identifier for the Role of which this Virtue is an instantiation. Format is implementation-specific.
applicationIds	true	set of strings	The IDs of the Applications that are currently contained in this Virtue. Format is implementation-specific.
resourceIds	true	set of strings	The IDs of the Resources that are currently attached to this Virtue. Format is implementation-specific.
transducerIds	true	set of strings	The IDs of the Transducers that are currently running in this Virtue. Format is implementation-specific.
state	true	string (enum)	The current state of the Virtue. *Enum Values*: `[CREATING, STOPPED, LAUNCHING, RUNNING, PAUSING, PAUSED, RESUMING, STOPPING, DELETING]`
ipAddress	false	string (ip)	An IP address for this Virtue. *Examples*: * `10.100.1.50`

4. Virtue (User) API

This section describes the commands in the Virtue API which must be implemented by the `virtue` CLI executable. Each command can be invoked by passing in the path as arguments to the `virtue` CLI executable. Each command has a number of input parameters, some of which are required, and may output some data to the console. Different exit conditions are enumerated with return codes.

4.1. Application Commands

4.1.1. Get Application

Path

```
application get
```

Description

Gets information about the indicated Application.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the requesting User.
applicationId	true	string	The ID of the Application to get. Format is implementation-specific.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidId*: The given ID is invalid.
254	**ERROR** *notImplemented*: This function has not been implemented.

Exit Code	Description
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

The Application with the given ID.
Type: Application

4.2. Role Commands

4.2.1. Get Role

Path

```
role get
```

Description

Gets information about the indicated Role.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the requesting User.
roleId	true	string	The ID of the Role to get. Format is implementation-specific.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidId*: The given ID is invalid.
254	**ERROR** *notImplemented*: This function has not been implemented.

Exit Code	Description
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

Information about the indicated Role.
Type: Role

4.3. User Commands

4.3.1. User Login

Path

```
user login
```

Description

Logs in a User with a username and password.

Input Parameters

Name	Required	Type	Description
username	true	string	Username of User to log in.
credentials	true	object	The credentials to authenticate the given User. The format is implementation-specific. *Examples*: * `{"password": "virtpassword"}`
forceLogoutOfOtherSessions	false	boolean	It is possible for the User to be logged into another session. If this flag is true, then any other sessions are forcibly logged out, leaving this call as the only open session. Default value is false if not provided.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.

Exit Code	Description
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidCredentialsFormat*: The given credentials are not in a valid format for this implementation.
11	**ERROR** *invalidCredentials*: Credentials are not valid for the indicated User.
12	**ERROR** *userAlreadyLoggedIn*: The given User is already logged into another session. This response is only given if the forceLogoutOfOtherSessions flag was false.
13	**ERROR** *userDoesntExist*: The given User does not exist in the system.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

The UserToken to be used for subsequent authorization calls.
Type: UserToken

4.3.2. User Logout

Path

```
user logout
```

Description

Logs out the User identified by the given UserToken.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the requesting User.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.

Exit Code	Description
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

4.3.3. List User Roles

Path

```
user role list
```

Description

Lists the Roles available to the given User.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the requesting User.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

A set of Roles available to the given User.
Type: set of Role

4.3.4. List User Virtues

Path

```
user virtue list
```

Description

Lists the current Virtue instantiations for the given User.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the requesting User.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

A set of Virtues for the given User.
Type: set of Virtue

4.4. Virtue Commands

4.4.1. Get Virtue Information

Path

```
virtue get
```

Description

Gets information about a specified Virtue by ID.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the requesting User.
virtueId	true	string	The ID of the Virtue to get. Format is implementation-specific.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidId*: The given ID is invalid.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

Information about the indicated Virtue.
Type: Virtue

4.4.2. Create Virtue

Path

```
virtue create
```

Description

Creates a new Virtue with the given properties. Also enables any Transducers on the Virtue that are supposed to be enabled on startup.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the requesting User.
roleId	true	string	The ID of the Role to instantiate for the Virtue. Format is implementation-specific.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *userNotAuthorizedForRole*: This User is not authorized to instantiate the given Role.
11	**ERROR** *virtueAlreadyExistsForRole*: A Virtue already exists for the given User and the given Role.
12	**ERROR** *invalidRoleId*: The given Role ID is not valid.
13	**ERROR** *cantEnableTransducers*: One of the configured Transducers can't be enabled on this Virtue.
100	**ERROR** *resourceCreationError*: There was an error creating the resources for the Virtue. Check user messages and server logs.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

Information about the created Virtue.
Type: Virtue

4.4.3. Launch a Virtue

Path

```
virtue launch
```

Description

Launches a Virtue

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the requesting User.
virtueId	true	string	The ID of the Virtue to launch. Format is implementation-specific.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidId*: The given ID is invalid.
11	**ERROR** *virtueAlreadyLaunched*: The indicated Virtue has already been launched.
12	**ERROR** *virtueStateCannotBeLaunched*: The indicated Virtue is in a state where it cannot be launched. Check the current Virtue state and take necessary actions.
13	**ERROR** *cantLaunchEnabledTransducers*: One or more of the enabled Transducers can't be launched.
100	**ERROR** *serverLaunchError*: There was an unanticipated server error launching the indicated Virtue. Check user messages for more information.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

Information about the launched Virtue.
Type: Virtue

4.4.4. Stop a Running Virtue

Path

```
virtue stop
```

Description

Stops a running Virtue.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the requesting User.
virtueId	true	string	The ID of the running Virtue to stop. Format is implementation-specific.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidId*: The given ID is invalid.
11	**ERROR** *virtueAlreadyStopped*: The indicated Virtue is already stopped.
12	**ERROR** *virtueStateCannotBeStopped*: The indicated Virtue is in a state where it cannot be stopped. Check the current Virtue state and take necessary actions.
100	**ERROR** *serverStopError*: There was an unanticipated server error stopping the indicated Virtue. Check user messages and server logs for more information.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

Information about the stopped Virtue.
Type: Virtue

4.4.5. Destroy a Stopped Virtue

Path

```
virtue destroy
```

Description

Destroys a Virtue. Releases all resources.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the requesting User.
virtueId	true	string	The ID of the running Virtue to destroy. Format is implementation-specific.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidId*: The given ID is invalid.
11	**ERROR** *virtueNotStopped*: The indicated Virtue is not stopped. Please stop it and try again.
100	**ERROR** *serverDestroyError*: There was an unanticipated server error destroying the indicated Virtue. Check user messages and server logs for more information.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

4.4.6. Launch a Virtue Application

Path

```
virtue application launch
```

Description

Launches an Application in a running Virtue.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the requesting User.
virtueId	true	string	The ID of the Virtue in which to launch the application. Format is implementation-specific.
applicationId	true	string	The ID of the Application to launch. Format is implementation-specific.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidVirtueId*: The given Virtue ID is invalid.
11	**ERROR** *invalidApplicationId*: The given Application ID is invalid.
12	**ERROR** *applicationNotInVirtue*: The indicated Application is not in this Virtue/Role.
13	**ERROR** *virtueNotRunning*: The Virtue holding this Application is not running. Launch it and try again.
14	**ERROR** *applicationAlreadyLaunched*: The indicated Application has already been launched.
100	**ERROR** *serverLaunchError*: There was an unanticipated server error launching the indicated Application. Check user messages for more information.

Exit Code	Description
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

Information about the launched Application. Format is implementation-specific.
Type: object

4.4.7. Stop a Running Virtue Application

Path

```
virtue application stop
```

Description

Stops a running Application in the indicated Virtue.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the requesting User.
virtueId	true	string	The ID of the Virtue in which to stop the application. Format is implementation-specific.
applicationId	true	string	The ID of the running Application to stop. Format is implementation-specific.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidVirtueId*: The given Virtue ID is invalid.
11	**ERROR** *invalidApplicationId*: The given Application ID is invalid.

Exit Code	Description
12	**ERROR** *applicationNotInVirtue*: The indicated Application is not in this Virtue/Role.
13	**ERROR** *virtueNotRunning*: The Virtue holding this Application is not running. Launch it and try again.
14	**ERROR** *applicationAlreadyStopped*: The indicated Application is already stopped.
100	**ERROR** *serverStopError*: There was an unanticipated server error stopping the indicated Application. Check user messages for more information.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

5. Virtue Administrative API

This section describes the commands in the Virtue-Admin API which must be implemented by the `virtue-admin` CLI executable. Each command can be invoked by passing in the path as arguments to the `virtue-admin` CLI executable. Each command has a number of input parameters, some of which are required, and may output some data to the console. Different exit conditions are enumerated with return codes.

5.1. Application Commands

5.1.1. List Applications

Path

```
application list
```

Description

Lists all Applications currently available in the system.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

All the Applications, with IDs.
Type: list of Application

5.2. Resource Commands

5.2.1. Get Resource

Path

```
resource get
```

Description

Gets information about the indicated Resource.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the requesting User.
resourceId	true	string	The ID of the Resource to get. Format is implementation-specific.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidId*: The given ID is invalid.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

Information about the indicated Resource.
Type: Resource

5.2.2. List Resources

Path

```
resource list
```

Description

Lists all Resources currently available in the system.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

All the Resources, with IDs.
Type: set of Resource

5.2.3. Attach Resource to Virtue

Path

```
resource attach
```

Description

Attaches the indicated Resource to the indicated Virtue. Does not change the underlying Role.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.
resourceId	true	string	The ID of the Resource to attach. Format is implementation-specific.
virtueId	true	string	The ID of the Virtue. Format is implementation-specific.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidResourceId*: The given Resource ID is invalid.
11	**ERROR** *invalidVirtueId*: The given Virtue ID is invalid.
12	**ERROR** *invalidVirtueState*: The given Virtue is not in a proper state to attach a Resource.
13	**ERROR** *invalidCredentials*: Don't have the correct credentials to attach this resource.
100	**ERROR** *cantAttach*: Can't attach to the Resource.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

The updated Virtue definition.
Type: Virtue

5.2.4. Detach Resource from Virtue

Path

```
resource detach
```

Description

Detaches the indicated Resource from the indicated Virtue. Does not change the underlying
Role.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.
resourceId	true	string	The ID of the Resource to detach. Format is implementation-specific.
virtueId	true	string	The ID of the Virtue. Format is implementation-specific.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidResourceId*: The given Resource ID is invalid.
11	**ERROR** *invalidVirtueId*: The given Virtue ID is invalid.
12	**ERROR** *invalidVirtueState*: The given Virtue is not in a proper state to detach a Resource.
100	**ERROR** *cantDetach*: Can't detach the Resource.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

The updated Virtue definition.
Type: Virtue

5.3. Role Commands

5.3.1. Create Role

Path

```
role create
```

Description

Creates a new Role with the given parameters.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.
role	true	Role	The Role parameters. ID will be ignored if present.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidFormat*: The implementation does not understand the format of this Role.
11	**ERROR** *invalidApplicationId*: One or more of the Application IDs is invalid.
12	**ERROR** *invalidResourceId*: One or more of the Resource IDs is invalid.
13	**ERROR** *invalidTransducerId*: One or more of the Transducer IDs is invalid.
14	**ERROR** *uniqueConstraintViolation*: This Role cannot be created because it violates a unique constraint.
15	**ERROR** *storageError*: There was an error trying to store this Role.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

The newly created Role, with ID.
Type: Role

5.3.2. List Roles

Path

```
role list
```

Description

Lists all Roles currently available in the system.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

All the Roles, with IDs.
Type: list of Role

5.4. System Commands

5.4.1. Export the Virtue system

Path

```
system export
```

Description

Export the Virtue system to a file.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

Exported Virtue system in the bytestream format.
Type: bytes

5.4.2. Import the Virtue system

Path

```
system import
```

Description

Import the Virtue system from the input bytestream data.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.
data	true	file	File byte data for the Virtue system.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

5.5. Test Commands

5.5.1. Import Testing User

Path

```
test import user
```

Description

Imports a pre-defined User that will be used for testing.
If called multiple times for the same User, the same username should be returned.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.

Name	Required	Type	Description
which	true	string (enum)	Which of the pre-defined Users to import. * ALICE: username "alice" * BOB: username "bob" *Enum Values*: `[ALICE, BOB]`

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *storageError*: There was an error trying to store the User.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

Credentials that can be used to log in as the user. Format is implementation-specific.
Type: object

5.5.2. Import Testing Application

Path

```
test import application
```

Description

Imports a pre-defined Application that will be used for testing.
If called multiple times for the same Application, the same ID should be returned.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.

Name	Required	Type	Description
which	true	string (enum)	Which of the pre-defined Applications to import. * FIREFOX: Mozilla Firefox, any recent version, either Windows or Linux. * INTERNET_EXPORER: Microsoft Internet Explorer, any recent version. * MICROSOFT_WORD: Microsoft Word, any recent version. * CALC: Windows calculator. * WINDOWS_CMD: Windows command-line application. * TERMINAL: A Linux terminal. * LIBREOFFICE_IMPRESS: LibreOffice Impress, either Windows or Linux. * LIBREOFFICE_WRITER: LibreOffice Writer, either Windows or Linux. * LIBREOFFICE_CALC: LibreOffice Calc, either Windows or Linux. *Enum Values*: `[FIREFOX, INTERNET_EXPLORER, MICROSOFT_WORD, CALC, WINDOWS_CMD, TERMINAL, LIBREOFFICE_IMPRESS, LIBREOFFICE_WRITER, LIBREOFFICE_CALC]`

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *storageError*: There was an error trying to store the Application.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

An ID that can be used for the indicated Application in future calls. Format is implementation-specific.
Type: string

5.5.3. Import Testing Role

Path

```
test import role
```

Description

Imports a pre-defined Role that will be used for testing.
If called multiple times for the same Role, the same ID should be returned.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.
which	true	string (enum)	Which of the pre-defined Roles to import. * OFFICE_USER: A Role with access to MICROSOFT_WORD, INTERNET_EXPLORER, LIBREOFFICE_IMPRESS, LIBREOFFICE_WRITER, and LIBREOFFICE_CALC. It can have any version. * POWER_USER: A Role with access to all applications. It can have any version. *Enum Values*: `[OFFICE_USER, POWER_USER]`

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *storageError*: There was an error trying to store the Role.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

An ID that can be used for the indicated Role in future calls. Format is implementation-

specific.
Type: string

5.6. User Commands

5.6.1. List Users

Path

```
user list
```

Description

Lists all Users currently present in the system.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

All the Users, with IDs.
Type: list of User

5.6.2. Get User

Path

```
user get
```

Description

Gets information about the indicated User.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.
username	true	string	The username of the User to get.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidUsername*: The given username is invalid, or doesn't exist.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

Information about the indicated User.
Type: User

5.6.3. List User Virtues

Path

```
user virtue list
```

Description

Lists the current Virtue instantiations for the given User.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.
username	true	string	The username of the User for whom to get Virtues.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidUsername*: The given username is invalid, or doesn't exist.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

A set of Virtues for the given User.
Type: set of Virtue

5.6.4. Force Logout User

Path

```
user logout
```

Description

Force logout of the indicated User and revoke their UserTokens. This should also post a message to the User to let them know what happened.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.

Name	Required	Type	Description
username	true	string	The username of the User to force logout.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *userNotLoggedIn*: The indicated User is not currently logged in.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

5.6.5. Authorize Role for User

Path

```
user role authorize
```

Description

Authorizes the indicated Role for the given User. This should also post a message to the User to let them know what happened.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.
username	true	string	The username of the User on which to operate.
roleId	true	string	The ID of the Role to authorize for the User. Format is implementation-specific.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidUsername*: The given username is invalid, or doesn't exist.
11	**ERROR** *invalidRoleId*: The given Role ID is invalid.
12	**ERROR** *userAlreadyAuthorized*: The User is already authorized for that Role.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

Information about the indicated User.
Type: User

5.6.6. Unauthorize Role for User

Path

```
user role unauthorize
```

Description

Unauthorizes the indicated Role for the given User.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.
username	true	string	The username of the User to operate on.
roleId	true	string	The ID of the Role to unauthorize for the User. Format is implementation-specific.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidUsername*: The given username is invalid, or doesn't exist.
11	**ERROR** *invalidRoleId*: The given Role ID is invalid.
12	**ERROR** *userNotAuthorized*: The User is not authorized for that Role.
13	*userUsingVirtue*: The indicated User is logged in and currently using a Virtue with the indicated Role. Force their logout and try again.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

Information about the indicated User.
Type: User

5.7. Usertoken Commands

5.7.1. List UserTokens

Path

```
usertoken list
```

Description

Lists all UserTokens currently present in the system.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the logged in admin User.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

All the UserTokens.
Type: set of UserToken

6. Virtue Security API

NOTE: The Virtue Security API is under active development. This section contains preliminary thoughts on the API being investigated.

This section describes the commands in the Virtue-Security API which must be implemented by the `virtue-security` CLI executable. Each command can be invoked by passing in the path as arguments to the `virtue-security` CLI executable. Each command has a number of input parameters, some of which are required, and may output some data to the console. Different exit conditions are enumerated with return codes.

6.1. Logging Commands

6.1.1. Get Available Log Components

Path

```
logging components get
```

Description

Gets all LogComponents that are currently registered.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the requesting User.

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
254	**ERROR** *notImplemented*: This function has not been implemented.

Exit Code	Description
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

All LogComponents currently defined in the system.
Type: set of LogComponent

6.1.2. Get Log Events

Path

```
logging events get
```

Description

Gets logging events that have been logged by a virtue.

Input Parameters

Name	Required	Type	Description
userToken	true	UserToken	The UserToken for the requesting User.
virtueId	false	string	If provided, get only LogEvents from this Virtue. Format is implementation-specific.
logComponentId	false	string	If provided, only get LogEvents from this LogComponent. Format is implementation-specific.
after	false	string (date-time)	If provided, only get LogEvents after the provided timestamp.
aboveLogLevel	false	string (enum)	If provided, only get LogEvents with a level above that in the provided LogLevel. Format is implementation-specific. *Enum Values*: `[NOTHING, ERROR, WARNING, INFO, DEBUG, EVERYTHING]`

Return Codes

Exit Code	Description
0	*success*: Successfully completed operation.
1	**ERROR** *invalidOrMissingParameters*: Invalid or missing parameters.

Exit Code	Description
2	**ERROR** *userNotAuthorized*: The User is not authorized for this operation.
3	**ERROR** *userTokenExpired*: The provided UserToken has expired. Refresh your session and try again.
10	**ERROR** *invalidVirtueId*: The given Virtue ID is invalid.
11	**ERROR** *invalidLogComponentId*: The given LogComponent ID is invalid.
12	**ERROR** *invalidAboveLogLevelId*: The given 'above LogLevel ID' is invalid.
254	**ERROR** *notImplemented*: This function has not been implemented.
255	**ERROR** *unspecifiedError*: An otherwise-unspecified error. Check user messages and/or error logs for more information.

Output

LogEvents that match the given query parameters, sorted by timestamp.
Type: list of LogEvent

7. Acronyms

API

Application Programming Interface.

APL or JHU/APL

The Johns Hopkins University Applied Physics Laboratory.

AWS

Amazon Web Services.

BAA

Broad Agency Announcement

CLI

Command-line interface.

CLUE

CLient Ui (user interface) Endpoint. This is the user interface for interacting with the Virtue system, consisting of a command-line interface and a graphical user interface.

GUI

Graphical user interface.

IARPA

Intelligence Advanced Research Projects Activity.

JSON

JavaScript Object Notation.

8. References

- [1] VirtUE Broad Agency Announcement (BAA): https://www.iarpa.gov/index.php/research-programs/virtue/virtue-baa
- [2] JSON specification: http://www.json.org/