Avocent® ACS800/8000 Advanced Console System

Application Programming Interface (API)

User Guide

Technical Support Site

If you encounter any installation or operational issues with your product, check the pertinent section of this manual to see if the issue can be resolved by following outlined procedures. Visit https://www.VertivCo.com/en-us/support/ for additional assistance.

TABLE OF CONTENTS

1 OVERVIEW

This document explains the external (remote web) Application Programming Interface (API) services and schema for the Avocent® ACS800/8000 Advanced Console System.

Remote web API services allow any third-party software system to integrate with the console system software. This integration enables your applications, tools and systems to manage information in the console system software and provides support for accessing information, performing unit control and adding events to the event log.

1.1 Base URL

The base URL format is: https://<IP_ADDRESS>:<PORT_NUMBER>/api/v1/

NOTE: The version number is mandatory to allow for future enhancements and expansion.

The HTTPS port number is 48048 by default. HTTP may also be used with a default port number of 8080 but this is completely insecure. By default, HTTP access is disabled and only HTTPS access is enabled. An admin can enable or disable HTTP and HTTPS access as well as change their respective port numbers.

NOTE: Changes to these settings will restart the RESTful API server and disconnect any existing RESTful sessions.

1.2 Methods

The following basic HTTP methods are supported.

Table 1.1 Supported HTTP Methods

METHOD	DESCRIPTION
POST	Used to create a new resource (specified in the JSON body) underneath the resource specified by the URL. The resource created is returned in the body or enough information is returned to find the new resource (an ID or URL). Also used to initiate actions.
GET	Requests a representation of the specified resource. This has no other effects other than reading and returning the data.
PUT	This method is used to modify an existing resource (specified by the URL) with the data present in the JSON body. Only the items present in the body are modified and the rest are left unchanged. The response code is typically 204 (Status No Content) with no content in the body unless otherwise requested by a specific parameter.
PATCH	This method is similar to the PUT method, but is typically intended only for modifying a portion of the specified resource. PUT in this API is also allowed to only modify a portion of a resource, so PATCH is included just for those applications that already use it.
DELETE	This method deletes the resource specified by the URL.

1.3 Body

The body for GET/POST/PUT/PATCH requests uses JSON syntax. This means that the "Content-Type" and "Accept" headers should be set to **application/json**. All parameters and values are case sensitive. Parameters that are string types must have their value enclosed in quotes even if the content is numerical, as is the case with some of the parameters that have a dropdown menu in the WebUI. All parameters are string type unless otherwise specified to be integer, array or something else.

1.4 Query Parameters

The fields query parameter is supported for many resources to enable the user to limit the fields that are returned. For example: **GET /serialPorts?fields=pinout,speed**.

This example above would return an array of all the serial ports with only the speed and pinout fields of each.

Sub-fields such as speed which is under the "physical" portion of a serialPort are unique and may be specified without any reference to the parent (physical in this example).

1.5 Response Codes

The API utilizes standard HTTP response codes where appropriate. The following table lists the response codes supported and typical usage.

Table 1.2 Response Code Descriptions

RESPONSE CODE	MEANING	DESCRIPTION
200 OK	Success	Returned for successful request. Response may include a JSON body with results.
201 Created	Created	Returned for successful request that has resulted in the creation of a new resource.
204 No Content	Success	Returned for successful request. Response does NOT include a JSON body with results.
400 Bad Request	Failure	Returned on failed request. Response includes a JSON encoded list of errors for one or more of the problematic parameters. May also indicate other system errors.
401 Unauthorized	Authorization Failure	Returned for request without the proper authentication.
404 Not Found	API not active	Returned for request where the resource is not found. Response typically includes a JSON encoded error structure with more error detail.

1.5.1 Error information

In addition to returning a failing response code, error information is returned in the response body providing more detail to the user. This response is in JSON format as follows:

```
{
"error": {
"code": "AE003",
"message": "invalid parameter",
"detail": "bob"
}
}
```

NOTE: Not all error information responses will include the detail field.

1.5.2 Ignored keys

If unknown keys are sent as part of the JSON body of a PUT or PATCH request, then they will be ignored by the API. This is part of the RESTful way of supporting different devices and different versions of API implementation. If a device doesn't understand or support something, it is permitted to ignore it.

Rather than ignoring mis-spellings and leaving the caller to wonder why a parameter didn't get set, the API will return a response body along with the 200 response code that contains a successful response message along with a list of keys that were ignored. This permits the caller to look for this information if so desired.

```
{
"status": "success", "ignoredKeys": [
```

```
"datea", "timee"
]
}
```

If a parent key is ignored, then all children below it are ignored but are not processed and listed as ignored keys.

Even if all keys are ignored, success will still be returned because the command didn't fail to write anything that it attempted.

1.6 Authentication

The ACS RESTful API supports two different methods of authentication: JSON Web Token (JWT) and basic authentication.

1.6.1 JSON Web Token (JWT)

The JWT method allows for the user to login and authenticate using the /sessions/login resource, passing it a valid appliance username and password in the JSON body of the request. If successful, the appliance returns a JWT that must be included in the header of all subsequent requests as the "Authorization" key with a value of "Bearer <JWT>".

This minimizes some of the appliance authentication overhead on each individual RESTful API call. The JWT remains valid for 60 minutes. A GET on the /sessions/refresh resource can be done before the token expires to refresh and provide a new token.

NOTE: The username and password are transmitted unencrypted as plain text in the original /sessions/login request body, so it is recommended to use HTTPS for RESTful communications.

1.6.2 Basic authentication

Basic authentication takes a username/password pair and encodes it using base64. This resulting base64 value must then be included in every request header as the "Authorization" key with a value of "Basic <BASE64_VALUE>". Each such request is then authenticated by the appliance, the request is executed, and the session terminated.

NOTE: The username/password pair are transmitted unencrypted (base64 is NOT secure encryption) in every request, so it is recommended to use HTTPS for RESTful communications.

1.7 Document Conventions

1.7.1 Abbreviated URLs

Throughout this document, the URLs in the examples will generally be abbreviated to just show the portion of the URL after the /v1. This is only to make the document more readable. The full URL is necessary when using the API. For example:

/system/info is shown instead of https://10.20.30.40:48048/api/v1/system/info

1.7.2 Examples

In the RESTful examples throughout the document, what is sent in the request is shown in bold type including both URL components and message body. The response body is shown in normal type. For example:

POST /sessions/login {"username": "admin", "password": "avocent"}

```
{
"token":"eyJhbGciOiJIUzI1NiIsInR…fVzCM"
}
```

2 API RESOURCES AND METHODS

The following table is an outline of the URLs that the API provides in this release.

Table 2.1 URL Descriptions

URL	METHOD	DESCRIPTION
Sessions		
/sessions/login	POST	Login with username and password, creating an API session, and returning a token in JSON body.
/sessions/logout	POST	Exit and cleanup the current session.
/sessions/refresh	GET	Using the token provided, obtain a new refreshed token which is returned in the JSON body response.
System		
/system/info	GET	Read basic system info such as serial number and type.
/system/reboot	POST	Reboots the appliance. Returns immediately.
/system/shutdown	POST	Shutdown the appliance. Returns immediately.
/system/factoryDefault	POST	Reset the appliance to the factory defaults and reboot.
/system/firmware/version	GET	Returns version information of firmware, bootcode, and date of build.
/system/firmware/download	POST	Download a firmware image file to the appliance using ftp, sftp or scp.
/system/firmware/install	POST	Installs a previously downloaded firmware image file.
/system/firmware/downloaded	GET	Returns the version information of a previously downloaded firmware file.
/system/config/save	POST	Save the appliance configuration to a file.
/system/config/restore	POST	Restore the appliance configuration from a saved file.
/system/dateAndTime	GET PUT PATCH	Read and configure system time parameters.
/system/dateAndTime/timezones	GET	Return a list of all recognized timezones.
/system/general	GET PUT PATCH	Read and configure general system parameters including onlineHelp, language, banner, etc.
Security		
/security	GET PUT PATCH	Read and configure various security profile parameters.
Network		
/network/settings	GET PUT PATCH	Read and configure various appliance specific network parameters.
/network/devices /network/devices/<ETH#>	GET PUT PATCH	Read and configure network device specific parameters: method (dhcp or static), IP address, netmask, gateway, etc.
Ports		
/serialPorts	GET	Read and configure various serial port parameters: status, pinout, parity, profile, etc.
/serialPorts/<PORT#>	GET PUT PATCH	
Pluggable		
/pluggableDevices /pluggableDevices/<NAME>	GET	Return a list of attached pluggable devices and their details or return details on the named pluggable device.
/pluggableDevices/<NAME>/eject	POST	Eject the specified device so that it is safe to remove.

URL	METHOD	DESCRIPTION
/pluggableDevices/<NAME>/delete	POST	Delete the specified device after having unplugged it.
/pluggableDevices/<NAME>/setConsole	POST	Setup the specified pluggable device as a console port.
Authentication		
/authentication/	GET PUT PATCH	Read and configure general appliance authentication parameters.
/authentication/dsview /authentication/kerberos /authentication/ldap /authentication/radius /authentication/tacacs	GET PUT PATCH	Read and configure authentication parameters specific to each type of authentication server.
Users		
/users /users/<NAME>	GET PUT PATCH POST DELETE	Read and configure user specific parameters, including listing all users, adding a user, and editing existing users.
Resources		
/resources	GET	Return a list of available API resources.

2.1 Sessions

2.1.1 /sessions/login

This action establishes a connection given username and password provided in the JSON body using the configured authentication of the appliance.

A web token is returned that is to be sent in the header of ALL subsequent requests as the "Authorization" key with a value of "Bearer <TOKEN>".

An alternative to this login session is to send a base64 encoded username/password pair in every API request using an "Authorization" key with a value of "Basic <BASE64_VALUE>" as detailed in section 1.6.

Methods

POST

Parameters

PARAMETER	DESCRIPTION
username	Valid username of an account on the appliance. (root, admin, etc.)
password	Valid password for the specified username.

Query

None

Response Body

JSON object

Response Codes

200	OK
401	Not Authorized

40x	Failure

Examples

```
POST /sessions/login {"username": "admin", "password": "avocent"}
{
"token":"eyJhbGciOiJIUzI1NiIsInR…fVzCM"
}
POST /sessions/login {"username": "bad", "password": "bad"}
{
"error": {
"code": "AE017",
"message": "user authentication failed"
}
}
```

2.1.2 /sessions/logout

This action invalidates the web token accompanied with it in the header and removes the associated API session from the appliance. Subsequent use of the web token will be unsuccessful.

Methods

POST

Parameters

None

Query

None

Response Body

JSON object

Response Codes

200	OK
40x	Failure

Examples

```
POST /sessions/logout
{
 "logout": "OK",
 "username": "root"
}
```

2.2 System

2.2.1 /system/info

This resource provides access to read only system information about the appliance's identity, versions, power and CPU information.

Methods

GET

Parameters

PARAMETER	DESCRIPTION
serialNumber	Serial number assigned to the appliance at the factory.
type	Description of the type of unit including model number with port count, power supplies, and modem presence. Example: ACS8048 with single power supply
bootcode	Version number of the installed bootcode. Example: 1.17
firmware	Full version number of the installed firmware. Example: 1.3.75.2779+551+28+11
firmwareDate	Date of the installed firmware. Example: Sep 1 2016 – 04:07:14
bootedFrom	Identifies whether the appliance is currently booted from hardware (internal Flash) or network.
powerSupply1	Status of power supply 1: on/off
powerSupply2	Status of power supply 2, if present: on/off
cpu	Description of the cpu: ARMv7 Processor rev 0 (v71)
cores	Number of cores in the cpu: integer 2

Query

Fields are supported for all parameters

Response Body

JSON object

Response Codes

200	OK
400	Bad request

Examples

GET /system/info


```
{
"serialNumber": "1234567890",
"type": "ACS8048 with single power supply",
"bootcode": "1.17",
"firmware": "1.3.75.2779+551+28+11",
"firmwareDate": "Sep 1 2016 – 04:07:14",
"bootedFrom": "hardware",
"powerSupply1": "on",
"cpu": "ARMv7 Processor rev 0 (v7l)",
"cores": 2
}
```


2.2.2 /system/reboot

This action causes the appliance to reboot.

Methods

POST

Parameters

None

Query

None

Response Body

JSON object

Response Codes

200	OK
40x	Failure

Examples

POST /system/reboot


```
{
"status": "initiated reboot"
}
```


2.2.3 /system/shutdown

This action causes the appliance to shut down.

Methods

POST

Parameters

None

Query

None

Response Body

JSON object

Response Codes

200	OK
40x	Failure

Examples

```
POST /system/shutdown
{
"status": "initiated shutdown"
}
```

2.2.4 /system/factoryDefault

This action restores the appliance to the factory default and reboots the appliance.

Methods

POST

Parameters

None

Query

None

Response Body

JSON object

Response Codes

200	OK
40x	Failure

Examples

```
POST /system/factoryDefault
{
"status": "initiated factoryDefault"
}
```

2.2.5 /system/firmware/version

This resource provides information about the currently installed and running firmware, including build date and version numbers of various components.

Methods

GET

Parameters

PARAMETER	DESCRIPTION
version	Full version number of the installed firmware. Example: 1.3.75.2779+551+28+11
bootVersion	Version number of the installed bootcode. Example: 1.17
date	Date of the firmware build. Example: Aug 12 2017 – 09:12:24

Query

None

Response Body

JSON object

Response Codes

200	OK
400	Bad Request

Examples

```
GET /system/firmware/version
{
"version": "1.3.75.2779+551+28+11",
"bootVersion": "1.17",
"date": "Aug 12 2017 – 09:12:24"
}
```

2.2.6 /system/firmware/download

This action causes the appliance to download the firmware file specified in preparation for subsequent firmware updating. The action does not return until the file download has completed or fails. Depending

on the network speed, this could take a couple minutes.

Methods

POST

Parameters

PARAMETER	DESCRIPTION
protocol	Specifies the protocol to use to download file: ftp/scp/sftp.
ipAddress	The IP address of the remote server from which to download the file.
username	The username to access the remote server.
password	The password to access the remote server.
directory	The directory path on the remote server, typically relative to the ftp root directory.
filename	The filename of the firmware file on the remote server.

Query

None

Response Body

JSON object

Response Codes

200	OK
400	Bad Request

Examples

```
POST /system/firmware/download
{
"protocol": "ftp",
"ipAddress": "10.20.30.80",
"username": "anonymous",
"password": "anonymous",
"directory": "pub/firmware/",
"filename": "firmware_acs8_1_2_9.fl"
}
Response is:
{
"status": "download successful",
"firmware": {
"version": "1.2.9.2449+540+23+11",
"date": "03/01/17"
}
}
```

2.2.7 /system/firmware/install

This action causes the appliance to install a previously downloaded firmware image into flash memory. This request does not return until the installation is complete, which may take up to two minutes.

Methods

POST

Parameters

None

Query

None

Response Body

JSON object

Response Codes

200	OK
400	Bad Request

Examples

```
POST /system/firmware/install
{
"status": "install successful",
"firmware": {
"version": "1.2.9.2449+540+23+11",
"date": "03/01/17"
}
}
```

2.2.8 /system/firmware/downloaded

This resource provides information about a firmware image that has previously been downloaded to the appliance.

Methods

GET

Parameters

PARAMETER	DESCRIPTION
version	Full version number of the firmware file. Example: 1.2.9.2449+540+23+11
date	Date of the build of the firmware file. Example: 03/01/17

Query

None

Response Body

JSON object

Response Codes

200	OK
400	Bad Request

Examples

```
GET /system/firmware/downloaded
{
"version": "1.2.9.2449+540+23+11",
"date": "03/01/17"
}
```

2.2.9 /system/config/save

This action saves the system configuration of the appliance.

This command does not return until the save is complete, which may take several minutes depending upon the format.

NOTE: XML format is not supported in the API.

Methods

POST

Parameters

PARAMETER	DESCRIPTION
format	Format to save the configuration in. Default is cli if nothing is specified. Otherwise choose from: cli/compressed
where	Where to save the file, either to local appliance file system or a remote server: local/remote.
protocol	Protocol used to transfer the file to a remote server: ftp/scp/sftp.
ipAddress	IP Address of the remote server.
username	Username of the account to use on the remote server.

PARAMETER	DESCRIPTION
password	Password for the specified username on the remote server. Defaults to "anonymous" if none is provided.
directory	Directory where the file is to be written. If the directory starts with "/", then it is considered an absolute path. Otherwise the directory is relative to /mnt/hdUser/backup for local, which is the default location of local configuration files, or relative to the specified protocol's configured base directory on the remote server.
filename	Filename to use for the saved configuration file.

Query

None

Response Body

JSON object

Response Codes

200	OK
400	Bad Request

Examples

```
POST /system/config/save
{
"format": "cli",
"where": "remote",
"protocol": "ftp",
"ipAddress": "10.20.30.70",
"username": "anonymous",
"password": "anonymous",
"directory": "pub",
"filename": "myconfig.cli"
}
Response is:
{
"status": "backup configuration saved as cli config filename:
pub/myconfig.cli"
}
```

2.2.10 /system/config/restore

Restores the appliance configuration from a specified file.

NOTE: XML format is not supported in the API.

Methods

POST

Parameters

PARAMETER	DESCRIPTION
where	Where the configuration file is located: local/remote.
protocol	Protocol used to retrieve remote file: ftp/scp/sftp.
ipAddress	IP Address of the remote server.
username	Username of the account to use on the remote server.
password	Password for the specified username on the remote server. Defaults to "anonymous" if none is provided.
directory	Directory where the file is located. If the directory starts with "/", then it is considered an absolute path. Otherwise the directory is relative to /mnt/hdUser/backup for local, which is the default location of local configuration files, or relative to the ftp/scp/sftp configured base directory on the remote server.
filename	Name of the configuration file to restore.

Query

None

Response Body

JSON object

Response Codes

200	OK
400	Bad Request

Examples

```
POST /system/config/restore
{
"where": "remote",
"protocol": "ftp",
"ipAddress": "10.20.30.70",
"username": "anonymous",
"password": "anonymous",
"directory": "pub",
"filename": "myconfig.cli"
}
Response is:
{
"status": "backup configuration restored from cli config filename:
pub/myconfig.cli"
}
```

The following command restores from the local file /mnt/hdUser/backup/myconfig.cli, which is sitting in the default local directory for backup configuration files. The directory parameter is not actually needed in this case, as it is assumed to be blank if not provided.

```
POST /system/config/restore
{
"where": "local",
"directory": "",
"filename": "myconfig.cli"
```

```
}
Response is:
{
"status": "backup configuration restored from cli config filename:
myconfig.cli"
}
```

2.2.11 /system/dateAndTime

Gets and sets parameters related to the system's date and time.

NOTE: Setting the time and date is not permitted when NTP is enabled and will result in an error.

Methods

GET, PUT, PATCH

Parameters

PARAMETER	DESCRIPTION
time	Current time.
date	Current date.
ntp	Network Time Protocol. enabled/disabled.
ntpSettings (only valid when ntp is enabled)	
ntpServer	IP address or name.
ntpServer2	IP address or name.
timezone	Name of the timezone. UTC is the default.

Query

Fields are supported for all parameters.

Response Body

JSON object

Response Codes

200 OK

204 OK No Content (for a PUT or PATCH)

400 Bad Request

Examples

```
GET /system/dateAndTime
{
"time": "18:00:50",
"date": "07/12/2017",
"ntp": "enabled",
"ntpSettings": {
```

```
"ntpServer": "10.20.30.40",
"ntpServer2": "ntp.pool.org"
},
"timezone": "UTC"
}
PUT /system/dateAndTime
{
"ntp": "disabled"
}
PUT /system/dateAndTime
{
"time": "05:00:00",
"date": "08/01/2017"
}
PUT /system/dateAndTime
{
"ntp": "enabled",
"ntpSettings": {
"ntpServer": "ntp.pool.org",
"ntpServer2": ""
}
}
PUT /system/dateAndTime {"timezone":"US/Central"}
```

2.2.12 /system/dateAndTime/timezones

Gets a list of all the recognized timezones. These values can be used when setting the timezone via the /system/dateAndTime resource.

Methods

GET

Parameters

None

Query

None

Response Body

JSON object

Response Codes

200	OK
400	Bad Request

Examples

```
GET /system/dateAndTime/timezones
{
```

```
"timezones": [
"Africa/Abidjan",
"Africa/Accra",
….
"Zulu"
]
}
```


2.2.13 /system/general

Gets and sets general appliance level parameters.

Methods

GET, PUT, PATCH

Parameters

PARAMETER	DESCRIPTION
language	Language used for ssh, telnet and console port sessions to the appliance. english/chinese/french/german/japanese/spanish.
onlineHelp	URL for the online help product documentation. Default: http://global.avocent.com/us/olh/acs8x/en/index.html.
banner	Controls whether a login banner is displayed. enabled/disabled.
bannerText	Text of the login banner to display. It may include special characters for formatting including \t for tabs and \n for newlines. Only shown in the full return body when banner is enabled.
viewer	The type of viewer to use when opening serial or appliance sessions. html5/jnlp.

Query

Fields are supported for all parameters.

Response Body

JSON object

Response Codes

200 OK

204 OK No Content (for a PUT or PATCH)

400 Bad Request

Examples


```
GET /system/general
{
"language": "english",
"onlineHelp": "http://global.avocent.com/us/olh/acs8x/en/index.html",
"banner": "enabled",
"bannerText":
"=================================================================
=======\n WARNING! The use of this system is restricted to
```

```
authorized users. \n \n All information and communications on
this system are subject \n to review, monitoring and recording at
any time, without notice\n or permission. Users should have no
expectation of
privacy.\n==========================================================
==========\n",
"viewer": "html5"
}
PUT /system/general {"bannerText":"==========\n Multiple\n Line\n
Banner\n==========\n"}

PUT /system/general {"language":"german","banner":"disabled","viewer":"jnlp"}
```

2.3 Security Profile

2.3.1 /security

Gets and sets parameters in the security profile of the appliance.

Methods

GET, PUT, PATCH

Parameters

PARAMETER	DESCRIPTION
idleTimeout	Session idle timeout in seconds: integer. 0 disables the timeout entirely, otherwise 90 seconds is the minimum accepted value.
rpc	RPC service: enabled/disabled
pluggableDevices	Pluggable device detection: enabled/disabled
pluggableStorage	Pluggable storage devices: enabled/disabled
access	Serial port access can be configured to allow access for all users, or allow the configuration of group and user-specific authorizations to restrict access: all/user_group
allAccessSettings (only valid when access is all)	
session	Sessions can be set to allow single or multiple session read/write: single/multiple
killMultiSession	Enables a user to kill other sharers of a session: enabled/disabled
sendMessageMultiSession	Enables a user to send a message to other users sharing a session: enabled/disabled
powerControl	Enables a user to control power of a multi session: enabled/disabled
dataBufferManagement	Enables a user for data buffer management of a session: enabled/disabled
restfulClientMenu	Enables the restfulClient menu: enabled/disabled
eraseFlash	Control whether entire flash config is erased on a factory default:

PARAMETER			DESCRIPTION
			enabled/disabled
bootp			Controls bootp configuration retrieval: enabled/disabled
bootpSettings (only valid when bootp is enabled)			
	bootpInterface		Specifies network interface used by bootp: eth#
	liveConfigurationRetrieval		Enables live configuration retrieval any time DHCP renews: enabled/disabled
sshUserPass			Controls whether SSH allows authentication via username/password: enabled/disabled
profile			Sets the appliance security profile: open/moderate/secure/custom
customProfile (only valid when profile is custom)			
	telnet		Telnet service: enabled/disabled
	ftp		FTP service: enabled/disabled
	snmp		SNMP service: enabled/disabled
	ipsec		IPSec: enabled/disabled
	answerIcmp		Answer ICMP message: enabled/disabled
	sshVersion		SSH Version: 1 / 2 / 1|2 / 2|1
	sshPort		SSH TCP port number (default 22): integer
	sshRootAccess		SSH allow root access: enabled/disabled
	sshCipherLevel		SSH minimum cipher and mac suite level: low/high
	httpSession		HTTP sessions: enabled/disabled
	httpSettings (only valid when httpSession is enabled)		
		httpPort	HTTP Port number (default 80): integer
	httpsSession		HTTPS sessions: enabled/disabled
	httpsSettings (only valid when httpsSession is enabled)		
		httpsTlsVersion	HTTPS TLS version: 1.1 / 1.1|1.2 / 1.2 / 1.1+1.0 / 1.1|1.2+1.0 / 1.2+1.0
		httpsCipherLevel	HTTPS minimum cipher suite level: low/medium/high
		httpsPort	HTTPS Port number (default 443): integer
		redirectHttp	Redirect HTTP/HTTPS: enabled/disabled
consolePort			Appliance console port: enabled/disabled
apiHttpAccess			Allow RESTful API access via HTTP: enabled/disabled
apiHttpSettings (only valid when apiHttpAccess is enabled)			

PARAMETER	DESCRIPTION
apiHttpPort	HTTP Port number for RESTful API access: integer
apiHttpsAccess	Allow RESTful API access via HTTPS: enabled/disabled
apiHttpsSettings (only valid when apiHttpsAccess is enabled)	
apiHttpsPort	HTTPS Port number for RESTful API access: integer
fips	FIPS 140-2 module: enabled/disabled NOTE: Changing the fips value will cause the appliance to reboot.
dsview	Allow appliance to be managed by DSView: enabled/disabled

Query

Fields are supported for all parameters.

Response Body

JSON object

Response Codes

200	OK
204	OK No Content (for a PUT or PATCH)
400	Bad Request

NOTE: Pluggable Storage Device changes will be effective after reboot only.

NOTE: Disabling Pluggable Device Detection will be effective after reboot only.

NOTE: Profile changes affecting HTTP and HTTPS will terminate all http sessions.

NOTE: Disabling the Console Port can make the appliance inaccessible and should only be done in the most extreme cases.

NOTE: API changes made while using the API may terminate the API session.

Examples

GET /security


```
{
"idleTimeout": 63955,
"rpc": "disabled",
"pluggableDevices": "enabled",
"pluggableStorage": "disabled",
"access": "all",
"allAccessSettings": {
"session": "multiple",
"killMultiSession": "enabled",
"sendMessageMultiSession": "enabled",
"powerControl": "enabled",
"dataBufferManagement": "enabled",
"restfulClientMenu": "enabled"
```

},
"eraseFlash": "disabled",
"bootp": "enabled",
"bootpSettings": {
"bootpInterface": "eth0",
"liveConfigurationRetrieval": "enabled"
},
"sshUserPass": "enabled",
"profile": "custom",
"customProfile": {
"telnet": "enabled",
"ftp": "disabled",
"snmp": "disabled",
"ipsec": "disabled",
"answerIcmp": "enabled",
"sshVersion": "1|2",
"sshPort": 22,
"sshRootAccess": "enabled",
"sshCipherLevel": "low",
"httpSession": "enabled",
"httpSettings": {
"httpPort": 80
},
"httpsSession": "enabled",
"httpsSettings": {
"httpsTlsVersion": "1.1|1.2+1.0",
"httpsCipherLevel": "low",
"httpsPort": 443,
"redirectHttp": "disabled"
}
},
"consolePort": "enabled",
"apiHttpAccess": "enabled",
"apiHttpSettings": {
"apiHttpPort": 8080
},
"apiHttpsAccess": "enabled",
"apiHttpsSettings": {
"apiHttpsPort": 48048
},
"fips": "disabled",
"dsview": "enabled"
}

GET /security

{
"idleTimeout": 63955,
"rpc": "disabled",
"pluggableDevices": "disabled",
"access": "user_group",
"eraseFlash": "disabled",
"bootp": "disabled",
"sshUserPass": "enabled",
"profile": "open",
"consolePort": "enabled",
"apiHttpAccess": "disabled",
"apiHttpsAccess": "disabled",
"fips": "disabled",
"dsview": "disabled"
}

```
GET /security?fields=idleTimeout
{
"idleTimeout": 300
}
GET /security?fields=rpc,pluggableDevices
{
"rpc": "disabled",
"pluggableDevices": "enabled"
}
PUT /security {"customProfile": {"httpsSettings": {"httpsPort": 444,
"redirectHttp": "disabled"}}}
```

2.4 Network

2.4.1 /network/settings

Get and set various network parameter settings of the console system.

NOTE: When configuring failoverSettings, the primary and secondary interface cannot be set to the same value, so when swapping them the changes need to be made in one PUT command rather than separately.

Methods

GET, PUT, PATCH

Parameters

PARAMETER		DESCRIPTION
hostname		User defined name of the appliance on the network. Defaults to format of "ACS80xx-<serialNumber>"
primaryDns		IP address of the primary DNS server.
secondaryDns		IP address of the secondary DNS server.
domain		Domain address. Default: corp.avocent.com
search		Search address. Default: corp.avocent.com
lldp		Link Layer Discovery Protocol: enabled/disabled
ipv6		IPv6 support: enabled/disabled
ipv6Settings (only valid when ipv6 is enabled)		
	dhcpv6Dns	Get the IPv6 DNS server address from DHCPv6: enabled/disabled
	dhcpv6Domain	Get the IPv6 Domain name from DHCPv6: enabled/disabled
multipleRouting		Enable network failover or IPv4 Multiple Routing Tables: none/enable_network_failover/enable_multiple_routing
failoverSettings (only valid when multipleRouting is enable_network_failover)		
	primaryInterface	Primary network interface: eth#
	secondaryInterface	Secondary network interface: eth#
	trigger	Failover trigger: primary_interface_down/unreachable_primary_default_gateway/ unreachable_ip_address/unreachable_dsview
	unreachableIp	IP Address to probe when trigger is unreachable_ip_address.
	dhcpv6Domain	Get the IPv6 Domain name from DHCPv6: enabled/disabled

Query

Fields are supported for all parameters.

Response Body

JSON object

Response Codes

200	OK
204	OK No Content (for a PUT or PATCH)
400	Bad Request

Examples

```
GET /network/settings
{
"hostname": "ACS8048-123456789",
"primaryDns": "10.20.30.40",
"secondaryDns": "10.20.30.50",
"domain": "corp.avocent.com",
"search": "corp.avocent.com",
"lldp": "disabled",
"ipv6": "enabled",
"ipv6Settings": {
"dhcpv6Dns": "disabled",
"dhcpv6Domain": "disabled"
},
"multipleRouting": "enable_network_failover",
"failoverSettings": {
"primaryInterface": "eth0",
"secondaryInterface": "eth1",
"trigger": "unreachable_ip_address",
"unreachableIp": "10.20.30.50"
}
}
PUT /network/settings {"hostname": "myhostname"}
```

2.4.2 /network/devices[/<INT>]

Get and set various network device parameters for individual network interfaces.

Methods

GET, PUT, PATCH

Parameters

PARAMETER	DESCRIPTION
interface	Read-only name of the network interface: eth0, eth1, etc.
isPrimary	Read-only. Identifies if this is the primary interface: enabled/disabled
status	Status of the interface: enabled/disabled
ipv4Method	Method used to configure IPv4: dhcp/static/unconfigured
ipv6Method	Method used to configure IPv6: stateless/dhcpv6/static/unconfigured

PARAMETER		DESCRIPTION
mac		Read only hardware mac address of this interface. Example: 00:e0:86:01:02:03
ipv4Static (only valid when ipv4Method is static)		
	ipv4Address	IPv4 Address for static configuration. Example: 10.20.30.40
	ipv4Mask	Subnet mask for static configuration. Example: 255.255.255.0
	ipv4Gateway	Gateway to use for static configuration. Example: 10.20.30.1
ipv6Static (only valid when ipv6Method is static)		
	ipv6Address	IPv6 Address for static configuration. Example: fd00:0024:0000:0000:02e0:86ff:fe10:2c3b
	ipv6PrefixLength	IPv6 Prefix Length for static configuration. Example: 112

Query

Fields are supported for all parameters.

Response Body

JSON object

Response Codes

200 OK

204 OK No Content (for a PUT or PATCH)

400 Bad Request

Examples

```
GET /network/devices
{
"devices": [
{
"interface": "eth0",
"isPrimary": "enabled",
"status": "enabled",
"ipv4Method": "dhcp",
"ipv6Method": "stateless",
"mac": "00:11:22:33:44:55",
},
{
"interface": "eth1",
…
}
]
}
GET /network/devices/eth1
{
"interface": "eth1",
"isPrimary": "disabled",
"status": "enabled",
"ipv4Method": "static",
"ipv6Method": "stateless"
"mac": "00:11:22:33:44:56",
"ipv4Static": {
"ipv4Address": "10.20.30.40",
"ipv4Mask": "255.255.255.0",
"ipv4Gateway": ""
}
}
```

2.5 Serial Ports

2.5.1 /serialPorts[/<PORT#>]

Get parameters for one or all serial ports. Set various serial port parameters for one serial port.

Methods

GET, PUT, PATCH

Parameters

PARAMETER		DESCRIPTION
port		Read-only port number
profile		Port profile: cas/power/dial_in/dial_out/socket_client/unconfigured Note: dial_in and dial_out are not supported at this time, and unconfigured is a readonly state that cannot be set.
deviceName		Read-only device name assigned by operating system. Examples: ttyS1, ttyUSB0, ttyACM0
status		Port status: enabled/disabled
physical (only valid when profile is cas, power or socket_client		
	pinout	Serial port pinout: auto/cisco/cyclades/rs422/rs485
	speed	Serial port speed: integer 1200/2400/4800/9600/19200/38400/57600/115200/230400

PARAMETER		DESCRIPTION
	parity	Parity: even/odd/none
	dataBits	Number of data bits: integer 5/6/7/8
	stopBits	Number of stop bits: integer 1/2
	flowControl	Flow control mechanism: none / hardware / software / rxon_software / txon_software
cas (only valid when profile is cas)		
	name	User assigned or default name for port. Default is the last half of the appliance mac address followed by "-p-" and the port number. Example: 10-2c-3d-p-1
	autoDiscovery	Enable auto discovery. The target name will be discovered and will be associated with this serial port. If it fails, the default port name will be used. enabled/disabled
	speedAutoDetection	Enable speed auto detection to try to discover the speed of the serial port: enabled/disabled
	protocol	The protocol that will be used by authorized users to access the serial port or target: ssh / telnet / raw_mode / telnet|ssh / telnet|raw_mode / ssh|raw_mode / telnet|ssh|raw_mode
	authentication	Authentication type that will be used to authenticate the user during target session: none / dsview_down_local / dsview / dsview|local / ldap_down_local / ldap / ldap|local / local / local|radius / local|tacacs+ / otp / otp|local / radius_down_local / radius / radius|local / tacacs+_down_local / tacacs+ / tacacs+|local
	textHotKey	Hotkey to suspend the target session and return to the cli prompt. Not available for Raw. Default: "^Z" which is Ctrl-Z
	powerHotKey	Hotkey to suspend the target session and display the Power Management Menu to control the outlets merged to the target. Not available for Raw. Default: "^P" which is Ctrl-P
	restfulHotKey	Hotkey to suspend the target session and display the RESTful Client Menu, which is used to send user-defined RESTful actions to a RESTful server. Default: "" (not configured)
	telnetAliasPort	TCP port used to connect directly to a serial port using Telnet protocol. Default: 7000+port number
	sshAliasPort	TCP port used to connect directly to a serial port using SSH protocol. Default: "" (not configured)
	rawModeAliasPort	TCP port used to connect directly to a serial port using raw socket for connection. Default: "" (not configured)
	ipv4AliasAddress	IPv4 address used to connect directly to a serial port.
	ipv4AliasInterface	Interface associated with the ipv4AliasAddress. eth#
	ipv6AliasAddress	IPv6 address used to connect directly to a serial port.
	ipv6AliasInterface	Interface associated with the ipv6AliasAddress. eth#
	dcdSensitivity	Allow session only if DCD is on. enabled/disabled
	autoAnswer	Enables processing of input data so that when the input data matches one input string configured in Auto Answer, the configured output string will be transmitted to the serial port. enabled/disabled
	dtrMode	DTR Mode can be set to the following: always_on – DTR status will always be on. normal – This is the default. The DTR status will depend on the existence of a CAS session. off_interval - when the a CAS session is closed, the DTR will stay down during this interval.
	dtrOffInterval	Interval used by DTR Mode off_interval in milliseconds: integer, Default: 100.
	linefeedSuppression	Enables the suppression of the linefeed character after the carriage return character. enabled/disabled
	nullAfterCrSuppression	Enables the suppression of the NULL character after the carriage return character. enabled/disabled
	transmissionInterval	The interval the port waits to send data to a remote client in milliseconds: integer
	breakSequence	An administrator can configure the control key as the break sequence, entering ^ before the letter. Default: ~break
	breakInterval	Interval for the break signal in milliseconds: integer, Default: 500
	multiSessionMenu	Enables the multi-session menu when connecting to a port that is already being accessed by another user. enabled/disabled
	loginNotification	Enables the notification to multi-session users when a new user logs in or a user logs out. enabled/disabled
	infoNotification	Displays an information message when a target session is opened. enabled/disabled
cas/dataBuffering (only valid when profile is cas)		
	bufferingStatus	Enables or disables data buffering. enabled/disabled
	bufferingType	Controls the type of data buffering:

PARAMETER		DESCRIPTION
		local – stores the data buffering file on the appliances local file system nfs – stores the data buffering file on an NFS server syslog – sends the data to the syslog server dsview – sends the data to the DSView server.
	localDevice	When the bufferingType is set to local, this field specifies where on the local system the data buffering files are stored. Options are the built-in memory (mmcblk0) or a connected USB storage or SD card location.
	timeStamp	When enabled, adds the time stamp to the data buffering line for a local or NFS server. enabled/disabled
	loginMessage	Includes special notification for logins and logouts in data buffering. enabled/disabled
	sessionLogging	Controls when data is stored. enabled/disabled
power (only valid when profile is power)		
	speedAutoDetect	Enable speed auto detect for power device. Tries to discover the speed of the attached power device. enabled/disabled
	pollingRate	The interval in seconds to update information from the PDU: integer, Default: 20
	deviceType	The type of power device connected to the serial port. pdu/ups
	pduType	Defines the type or vendor of the PDU connected to the serial port. auto/cyclades/enp/spc/servertech/raritan/apc/eaton
	powerCycleInterval	The interval in seconds between Off and On actions for the power cycle command: integer, Default: 15
	syslog	When enabled, the PDU will send syslog messages to the appliance. enabled/disabled
	buzzer	Enables or disables the PDU's buzzer. enabled/disabled
	overcurrentProtection	When enabled, the software's overcurrent protection is on. enabled/disabled
	upsType	Defines the type or vendor of the UPS connected to the serial port. gxt4
socketClient (only valid when profile is socket_client)		
	remoteServer	IPv4 or IPv6 address of the remote server.
	remoteTcpPort	TCP port to be used to establish a connection with a remote server.
	establishConnection	Configure the event that will trigger the establishment of the connection. dcd/always

Query

Fields are supported for all parameters.

Response Body

JSON object

Response Codes

200 OK

204 OK No Content (for a PUT or PATCH)

400 Bad Request

Examples

GET /serialPorts/1

```
{
"port": "1",
"profile": "cas",
"deviceName": "ttyS1",
"status": "enabled",
"physical": {
"pinout": "cisco",
"speed": 115200,
"parity": "none",
"dataBits": 8,
"stopBits": 1,
"flowControl": "none"
},
"cas": {
"name": "MyServer",
"autoDiscovery": "disabled",
…
}
}
```

GET /serialPorts

```
{
"serialPorts": [
{
"port": "1",
"profile": "cas",
"deviceName": "ttyS1",
"status": "enabled",
…
},
…
{
"port": "48",
…
}
]
}
```

PUT /serialPorts/1 '{"cas":{"sshAliasPort":"8001"}}

}

GET /serialPorts/75

```
{"error": {
"code": "AE002",
"message": "resource id not found",
"detail": "75 is not a valid port id"
}
}
```

2.6 Pluggable Devices

2.6.1 /pluggableDevices[/<NAME>]

Read information about the attached pluggable devices (USB, SD card).

Methods

GET

Parameters

PARAMETER	DESCRIPTION
deviceName	Linux assigned name of the USB device. (ie ttyACM0, ttyUSB0)
deviceType	Device's type: console/ethernet/modem/storage/wirelessModem
card	Physical device type: "mmc SD" / "usb usbslot"
devicePath	USB device path to uniquely identify where the device is in the usb tree.
deviceInfo	Device info available via usb device descriptors.
status	Current state of the device: inserted/unmounted/ejected
port	Serial port number assigned to this device if it is enabled as a console.

Query

Fields are supported for all parameters.

Response Body

JSON object

Response Codes

200	OK
400	Bad Request

Examples

GET /pluggableDevices

```
{
"pluggableDevices": [
{
"deviceName": "ttyACM0",
"deviceType": "Console",
"card": "usb usbslot",
"devicePath": "1-1.3",
|"deviceInfo": "…",
"status": "ejected",
"port": "49"
},
…
]
}
```

GET /pluggableDevices/ttyUSB0

```
{
"deviceName": "ttyUSB0",
"deviceType": "console",
"card": "usb usbslot",
"devicePath": "1-1.4",
"deviceInfo": "…",
"status": "inserted",
"port": "49"
}
```

GET /pluggableDevices/mmcblk1p1

```
{
"deviceName": "mmcblk1p1",
"deviceType": "stroage",
"card": "mmc SD",
"devicePath": "",
"deviceInfo": " ",
"status": "inserted"
}
```

2.6.2 /pluggableDevices/<NAME>/setConsole

Setup the specified pluggable device as a console port. This adds the device to the list of serial ports by adding a new serial port number. The serial port must then be configured appropriately. This can only be done for devices which show a deviceType of "console".

Methods

POST

Parameters

None

Query

None

Response Body

JSON object

Response Codes

200	OK
400	Bad Request

Examples

```
POST /pluggableDevices/ttyUSB0/setConsole
{
"status": "success. ttyUSB0 set to console, port 49"
}
```

2.6.3 /pluggableDevices/<NAME>/eject

Eject the specified pluggable device so that it can be physically removed without causing loss of data. For a storage device, this makes sure the device is not busy.

Methods

POST

Parameters

None

Query

None

Response Body

JSON object

Response Codes

200	OK
400	Bad Request

Examples

```
POST /pluggableDevices/sda1/eject
{
"status": "It is now safe to physically unplug the sda1 device"
}
```

2.6.4 /pluggableDevices/<NAME>/delete

Delete the specified pluggable device after it has been safely ejected and physically unplugged.

Methods

POST

Parameters

None

Query

None

Response Body

JSON object

Response Codes

200	OK
400	Bad Request

Examples

```
POST /pluggableDevices/ttyUSB1/delete
{
"status": "success. ttyUSB1 deleted"
}
```

2.7 Authentication

2.7.1 /authentication

Get and set various appliance authentication parameters.

Methods

GET, PUT, PATCH

Parameters

PARAMETER	DESCRIPTION
applianceAuthType	Type of authentication to use for granting access to the appliance: dsview_down_local / dsview / dsview|local / ldap_down_local / ldap / ldap|local / local / local|radius / local|tacacs+ / otp / otp|local / radius_down_local / radius / radius|local / tacacs+_down_local / tacacs+ / tacacs+|local
singleSignOn	Enabling single sign-on uses the specified single sign-on authentication and no further authentication is needed when accessing a port: enabled/disabled
singleSignOnAuthType	Authentication to use for single sign-on: unconfigured / dsview_down_local / dsview / dsview|local / ldap_down_local / ldap / ldap|local / local / local|radius / local|tacacs+ / otp / otp|local / radius_down_local / radius / radius|local / tacacs+_down_local / tacacs+ / tacacs+|local

Query

Fields are supported for all parameters.

Response Body

JSON object

Response Codes

200 OK

204 OK No Content (for a PUT or PATCH)

400 Bad Request

Examples

```
GET /authentication/kerberos
{
"server": "10.20.30.41",
"realmDomainName": "avocent.com",
"domainName": "avocent.com"
}
PUT /authentication/kerberos { "server": "10.20.30.41" }
```

2.7.2 /authentication/ldap

Get and set LDAP authentication parameters.

Methods

GET, PUT, PATCH

Parameters

PARAMETER	DESCRIPTION
server	IP Address of the LDAP server.
base	Base.
secure	Secure mode: on/off/start_tls
userName	The database username.
password	The database password for the username.
attributes	Login attributes.

Query

Fields are supported for all parameters.

Response Body

JSON object

Response Codes

200 OK

204 OK No Content (for a PUT or PATCH)

400 Bad Request

Examples

```
GET /authentication/ldap
{
"server": "10.20.30.42",
"base": "",
"secure": "off",
"userName": "myuser",
"password": "mypassword",
"attributes": ""
}
PUT /authentication/ldap { "server":"10.20.30.42", "secure": "on" }
```

2.7.3 /authentication/radius

Get and set RADIUS authentication parameters.

Methods

GET, PUT, PATCH

Parameters

PARAMETER		DESCRIPTION
firstAuthenticationServer		IP Address of the first authentication server.
firstAccountingServer		IP Address of the first accounting server.
secondAuthenticationServer		IP Address of the second authentication server.
secondAccountingServer		IP Address of the second accounting server.
secret		Secret word or passphrase, applies to both sets of servers.
timeout		Desired number of seconds for server timeout: integer
retries		Desired number of retries: integer
serviceType		Enable Service-Type attribute to specify the authorization group: enabled/disabled.
serviceTypeGroups: (only valid if serviceType is enabled)		
	login	Authorization group name for Login.
	framed	Authorization group name for Framed.
	callbackLogin	Authorization group name for Callback Login.
	callbackFramed	Authorization group name for Callback Framed.
	outbound	Authorization group name for Outbound.
	administrative	Authorization group name for Administrative.

Query

Fields are supported for all parameters.

Response Body

JSON object

Response Codes

200 OK

204 OK No Content (for a PUT or PATCH)

400 Bad Request

Examples

GET /authentication/radius

```
{
"firstAuthenticationServer": "127.0.0.1",
"firstAccountingServer": "127.0.0.1",
"secondAuthenticationServer": "",
"secondAccountingServer": "",
"secret": "******",
"timeout": 3,
"retries": 2,
"serviceType": "enabled",
"serviceTypeGroups": {
"login": "",
"framed": "",
"callbackLogin": "",
"callbackFramed": "",
"outbound": "",
"administrative": ""
}
}
```

PUT /authentication/radius {"firstAuthenticationServer": "10.20.30.45","timeout":120}

2.7.4 /authentication/tacacs

Get and set TACACS+ authentication parameters.

Methods

GET, PUT, PATCH

Parameters

PARAMETER	DESCRIPTION
firstAuthenticationServer	IP Address of the first authentication server.
firstAccountingServer	IP Address of the first accounting server.
secondAuthenticationServer	IP Address of the second authentication server.
secondAccountingServer	IP Address of the second accounting server.
service	Service: ppp/raccess/shell
secret	Secret word or passphrase, applies to both sets of servers.
timeout	Desired number of seconds for server timeout.
retries	Desired number of retries.
version	Version: v0 / v0_v1 / v1 / v1_v0

PARAMETER		DESCRIPTION
userLevel		Enable User-Level attribute to specify the authorization group: enabled/disabled
userLevelGroups: (only valid if userLevel is enabled)		
	userLevel1	Authorization group name for User-Level 1.
	...	...
	userLevel15	Authorization group name for User-Level 15.

Query

Fields are supported for all parameters.

Response Body

JSON object

Response Codes

200 OK

204 OK No Content (for a PUT or PATCH)

400 Bad Request

Examples

GET /authentication/tacacs

```
{
"firstAuthenticationServer": "10.20.30.46",
"firstAccountingServer": "10.20.30.47",
"secondAuthenticationServer": "",
"secondAccountingServer": "",
"service": "ppp",
"secret": "",
"timeout": 10,
"retries": 2,
"version": "v1",
"userLevel": "enabled",
"userLevelGroups": {
"userLevel1": "",
…
"userLevel15": ""
}
}
```

PUT /authentication/tacacs {"firstAuthenticationServer": "10.20.30.46", "firstAccountingServer": "10.20.30.47","version": "v0_v1"}

2.8 Users

2.8.1 /users[/<NAME>]

This resource provides the ability to view and edit user settings as well as add new users and delete existing users.

Methods

GET, PUT, PATCH, POST, DELETE

NOTE: For methods PUT, PATCH, POST, and DELETE, the Name is required and not optional. For example: DELETE /users/testuser

Parameters

PARAMETER		DESCRIPTION
name		User name. admin and root exist by default.
settings		
	password	Password for the user.
	changePasswordNextLogin	Set to force the user to change the password the next time they log in. enabled/disabled.
	userGroups	List of groups to which this user belongs: string array.
passwordExpiration		
	minimumDays	The minimum number of days allowed between password changes: integer
	maximumDays	The maximum number of days a password is valid: integer
	inactiveDays	The number of inactive days after which a password is considered expired: integer
	warningDays	The number of days a warning is issued to the user prior to expiration: integer
accountExpiration		Account expiration date.

Query

Fields are supported for all parameters.

Response Body

JSON object

Response Codes

200 OK

201 Created

204 OK No Content (for a PUT or PATCH)

400 Bad Request

Examples

GET /users

```
{
"users": [
{
"name": "admin",
"settings": {
"password": "",
…
},
{
"name": "root",
…
}
]
}
```

GET /users/admin

```
{
"name": "admin",
"settings": {
"password": "",
"changePasswordNextLogin": "disabled",
"userGroups": [
"admin"
],
"passwordExpiration": {
"minimumDays": 0,
"maximumDays": 99999,
"inactiveDays": "",
"warningDays": 7
},
"accountExp": ""
}
}
POST /users
{
"name": "bob",
"settings": {
"password": "password1234!",
"changePasswordNextLogin": "enabled",
…
}
Response is:
{
"name": "bob",
"settings": {
"password": "password1234!",
…
}
```

2.9 Resources

2.9.1 /resources

This resource provides a list of the available resources and methods.

Methods

GET

Parameters

None

Query

None

Response Body

JSON object

Response Codes

200	OK

40x	Failure

Examples


```
{
"resources": [
"GET /authentication",
"PATCH /authentication",
"PUT /authentication",
"GET /authentication/dsview",
"PATCH /authentication/dsview",
"PUT /authentication/dsview",
"GET /authentication/kerberos",
"PATCH /authentication/kerberos",
"PUT /authentication/kerberos",
"GET /authentication/ldap",
"PATCH /authentication/ldap",
"PUT /authentication/ldap",
"GET /authentication/radius",
"PATCH /authentication/radius",
"PUT /authentication/radius",
"GET /authentication/tacacs",
"PATCH /authentication/tacacs",
"PUT /authentication/tacacs",
"GET /network/devices",
"GET /network/devices/:INT",
"PATCH /network/devices/:INT",
"PUT /network/devices/:INT",
"GET /network/settings",
"PATCH /network/settings",
"PUT /network/settings",
"GET /pluggableDevices",
"GET /pluggableDevices/:NAME",
"POST /pluggableDevices/:NAME/delete",
"POST /pluggableDevices/:NAME/eject",
"POST /pluggableDevices/:NAME/setConsole",
"GET /resources",
"GET /security",
"PATCH /security",
"PUT /security",
"GET /serialPorts",
"GET /serialPorts/:PORT",
"PATCH /serialPorts/:PORT",
"PUT /serialPorts/:PORT",
"POST /sessions/login",
"POST /sessions/logout",
"GET /sessions/refresh",
"POST /system/config/restore",
"POST /system/config/save",
"GET /system/dateAndTime",
"PATCH /system/dateAndTime",
"PUT /system/dateAndTime",
"GET /system/dateAndTime/timezones",
"POST /system/factoryDefault",
"POST /system/firmware/download",
"GET /system/firmware/downloaded",
"POST /system/firmware/install",
"GET /system/firmware/version",
"GET /system/general",
"PATCH /system/general",
"PUT /system/general",
```

```
"GET /system/info",
"POST /system/reboot",
"POST /system/shutdown",
"GET /users",
"POST /users",
"DELETE /users/:NAME",
"GET /users/:NAME",
"PATCH /users/:NAME",
"PUT /users/:NAME"
]
}
```

This page intentionally left blank.

APPENDICES

Appendix A: cURL

The cURL command line utility is one method that can be used to communicate with the RESTful API.

Examples:

```
$ curl -H "Content-Type: application/json" -H "Accept:application/json"
http://10.20.30.40:8080/api/v1/sessions/login -d
'{"username":"admin","password":"avocent"}'

{
"token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1MDQxMTU2NzUsImlkIjoiYWRt
aW4iLCJvcmlnX2lhdCI6MTUwNDExMjA3NSwic2lkIjo4fQ.UYGXje5It2hAJryruP3etUaabSh5pfiPP_sXXZF37og"
}

$ curl -H "Content-Type: application/json" -H "Accept:application/json" -H
"Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1MDQxMTU2NzUsImlkIjoiYWRta
W4iLCJvcmlnX2lhdCI6MTUwNDExMjA3NSwic2lkIjo4fQ.UYGXje5It2hAJryruP3etUaabSh5
pfiPP_sXXZF37og" http://10.20.30.40:8080/api/v1/system/info
{
"serialNumber": "0012345678",
"type": "ACS8048 with single power supply",
"bootcode": "1.17",
"firmware": "1.3.77.2909+551+28+11",
"bootedFrom": "hardware",
"powerSupply1": "on",
"cpu": "ARMv7 Processor rev 0 (v7l)",
"cores": 2
}
```

Appendix B: Python

Python examples are included for both http and https that demonstrate how python can be used to communicate with the RESTful API. These examples run on Python versions 2.7 and 3.5. They require the python "requests" module be installed on the system running python.

An example can be run as follows:

```
$ python acsapi_example_http.py
```

Appendix C: Helper Script

A bash shell helper script is provided in the ACS8000 root filesystem under /usr/share/restapi/restapi-helper.sh.

This helper script modifies the environment of the running shell to add GET/PUT/POST/PATCH/DELETE shell functions. These functions provide a simple command line interface to demonstrate the API. The shell functions utilize the 'curl' program, which must be installed on the system.

From a bash shell, the script must be "sourced" in order to make the necessary changes to the shell environment which are shown as SETUP parameters and functions.

When sourcing the script, the last parameter is the IPv4 address of the ACS8000. In the example below, 10.20.30.40 is the ACS IP address.

```
$ . restapi-helper.sh 10.20.30.40

Token for Basic Authentication saved in /home/root/.acsrestapi/Basictoken-127.0.0.1

SETUP parameters:

ACSHOST                                  127.0.0.1

ACSPROTOCOL http                         http

NOTE: A special means for using the helper script is provided when logged into the appliance as a user in
the admin group. The command useapi allows subsequent GET and PUT commands to the logged in
appliance.

ACSPORT                                  8180

ACSTOKENDIR                              /home/root/.acsrestapi

ACSTOKENTYP                              Basic (Basic Authentication)

ACSURL                                   /api/v1

ACSDEBUG                                 no

Change your ACSHOST parameter to the ACS IP Address. For example:ACSHOST=192.168.161.10

GET /resource

                                         Example:

                                         GET /serialPorts/2

PUT /resource '{...json parameters...}'

                                         Example:

                                         PUT /serialPorts/2 '{"physical":{"speed":38400}}'

PATCH /resource '{...json parameters...}'

                                         Example:

                                         PATCH /security '{"idleTimeout":0}'
```

```
POST /resource '{...json parameters...}'

                                                    Example:
                                                    POST /sessions/login
                                                    '{"username":"admin","password":"avocent"}'
DELETE /resource

  Note: These functions automatically prepend
  '/acs/v1' to the /resource
```

After running the helper script, the current shell environment can now execute POST, GET and the other commands mentioned above:

```
$ GET /system/info
{
"serialNumber": "001234567",
"type": "ACS8048 with single power supply",
"bootcode": "1.17",
"firmware": "1.3.77.2909+551+28+11",
"firmwareDate": "Sep 20 2017 – 08:03:39",
"bootedFrom": "Hardware",
"powerSupply1": "On",
"cpu": "ARMv7 Processor rev 0 (v7l)",
"cores": 2
}
```

Appendix D: Certificate Verification

When using HTTPS, in order to avoid certificate verification warnings, the client should be set to ignore SSL certificate verification.

With the python requests package, this is done by adding "verify=False" to the requests command:

```
>>> requests.get(URL, verify=False)
```

With cURL the "-k" option is used to disable certificate verification:

```
$ curl -k -H "Content-Type: application/json" -H "Accept:application/json"
https://10.20.30.40:48048/api/v1/sessions/login -d
'{"username":"admin","password":"avocent"}'
```

The restapi-helper script automatically includes the -k option on the underlying curl commands when using https.

VertivCo.com | Vertiv Headquarters, 1050 Dearborn Drive, Columbus, OH, 43085, USA

590-1814-501B