Wry Documentation

Release 1.0

Ocado Technology

Oct 25, 2017

Contents

A pythonic AMT control library.

CHAPTER 1

Introduction

Wry is a library that facilitates interaction with, and configuration and control of, hardware devices that implement Intel AMT (vPro) technology.

It uses the openwsman python bindings.

Quickstart

Wry's functionality is exposed through the AMTDevice class. Initialize it as such:

```
>>> from wry import AMTDevice
>>> dev = AMTDevice(address, 'http', username, password)
```

You can then access different apects of device functionality, through aspect-specific namespaces. For example:

```
>>> dev.power.turn_on()
>>> dev.power.state
StateMap(state='on', sub_state=None)
```

Currently, the following namespaces are implemented:

- dev.power, via *wry.AMTPower*, provides access to:
  - Power state and control
- dev.boot, via *wry.AMTBoot*, provides access to:
  - Boot configuration
  - Boot medium selection
- dev.vnc, via *wry.AMTKVM*, provides access to:
  - Remote KVM (VNC) state and configuration
  - Setting of [additonal] user opt-in policy for KVM

- dev.opt_in, via *wry.AMTOptIn*, provides access to:
  - Setting of opt-in policies for KVM, Serial-over-LAN and media redirection
- dev.redirection, via *wry.AMTRedirection*, provides access to:
  - State and control of media redirection (IDER)
  - State and control of Serial-over-LAN (SOL)

You can click on a class name above, to see documentation for the available methods.

Status

Wry is in the early stages of development, and the interfaces it exposes may change as a result. Issues and pull requests are more than welcome.

Wry currently supports only Python 2.7. There are no philosophical reasons for this; it simply matches our target environment. Patches to support other platforms are welcome.

Compatibility

Wry relies on the wsman AMT protocol, and therefore supports AMT versions 7(?) onwards.

Tested on the following hardware/firmware:

- Intel NUC DC53427HYE (BIOS 0037, ME 8.1.40.1416)
- Intel NUC5i5MYBE

License

Apache. (C) 2015-2017 Ocado Innovation Ltd. Please see the `LICENSE` and `NOTICE` files.

CHAPTER 2

Usage

Quickstart

Wry's functionality is exposed through the AMTDevice class. Initialize it as such:

```
>>> from wry import AMTDevice
>>> dev = AMTDevice(address, 'http', username, password)
```

You can then access different apects of device functionality, through aspect-specific namespaces. For example:

```
>>> dev.power.turn_on()
>>> dev.power.state
StateMap(state='on', sub_state=None)
```

Currently, the following namespaces are implemented:

- dev.power, via *wry.AMTPower*, provides access to:
  - Power state and control
- dev.boot, via *wry.AMTBoot*, provides access to:
  - Boot configuration
  - Boot medium selection
- dev.vnc, via *wry.AMTKVM*, provides access to:
  - Remote KVM (VNC) state and configuration
  - Setting of [additonal] user opt-in policy for KVM
- dev.opt_in, via *wry.AMTOptIn*, provides access to:
  - Setting of opt-in policies for KVM, Serial-over-LAN and media redirection
- dev.redirection, via *wry.AMTRedirection*, provides access to:
  - State and control of media redirection (IDER)

  - State and control of Serial-over-LAN (SOL)

You can click on a class name above, to see documentation for the available methods.

In-Depth

As well as the above, the AMTDevice class provides more genearlized/low-level functionality.

class `wry.`**`AMTDevice`**(*target=None*, *is_ssl=True*, *username=None*, *password=None*, *debug=False*, *showxml=False*)

A wrapper class which packages AMT functionality into an accessible, device-centric format.

`dump`(*as_json=True*)

Print all of the known information about the device.

Returns WryDict or json.

class `wry.`**`AMTPower`**(*device*)

Control over a device's power state.

`available_states`()

Get a list of available power states given our current power state

`request_power_state_change`(*power_state*)

Change the NUC to the specified power state

`reset`()

Reboot the device.

`state`

A property which describes the machine's power state.

A `wry.device.StateMap` as described in `wry.device.AMT_POWER_STATE_MAP`.

`toggle`()

If the device is off, turn it on. If it is on, turn it off.

`turn_off`()

Turn off the device.

`turn_on`()

Turn on the device.

class `wry.`**`AMTKVM`**(*device*)

Control over a device's KVM (VNC) functionality.

`default_screen`

Default Screen. An integer.

`enabled`

Whether KVM functionality is enabled or disabled.

True/False

Note: This will return True even if KVM is enabled, but no ports for it are.

`enabled_ports`

Tells you (and/or allows you to set) the enabled ports for VNC.

opt_in_timeout
User opt-in timeout for KVM access, in seconds.

If set to 0, opt-in will be disabled.

password
This doesn't fail but always appears to return None

port_5900_enabled
Whether the standard VNC port (5900) is enabled. True/False.

session_timeout
Session timeout. In minutes.

setup(*password='', port5900Enabled=False, defaultScreen=0, optIn=True, optInTimeout=60, sessionTimeout=10*)
Set all basic KVM settings in one call

class wry.**AMTBoot**(*device*)
Control how the machine will boot next time.

config
Get configuration for the machine's next boot.

supported_media
Media the device can be configured to boot from.

wry.**AMTOptIn**
alias of *wry.AMTOptIn*

class wry.**AMTRedirection**(*device*)
Control over Serial-over-LAN and storage redirection.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index