python-device42api Documentation

Release v1.0

Michael Lang

April 13, 2014

Contents:

CHAPTER 1

Indices and tables

- *genindex*
- *modindex*
- *search*

class device42api.**Asset**(*json=None*, *parent=None*, *api=None*)

create Asset object

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> a = device42api.Asset(api=api)
>>> a.type = 'AC' # AC,Breaker Panel,Cable Modem,DMARC,Fabric Extender,Fax Machine,Filler Panel,
                  # Patch Panel Module,Projector,Scanner,Shredder,Software,Speaker Phone,TAP Mod
>>> a.serial_no = '1234567890'
>>> a.vendor = 'Test'
>>> a.building = 'TestBuilding'
>>> a.room = 'Test Room'
>>> a.rack_id = 80
>>> a.start_at = 1
>>> a.save()
{'msg': ['asset added/edited.', 1, ''], 'code': 0}
```

add_customField(*cf=None*)

add custom Fields to the object

```
>>> asset = api.get_asset('Rack with CustomFields')[0]
>>> cf = device42api.CustomField(api=api)
>>> cf.key      = 'used_since'
>>> cf.type     = 'date'
>>> cf.value    = '2014-04-02'
>>> asset.add_customField(cf)
{'msg': ['custom key pair values added or updated', 15, 'Asset with CustomFields - AC'], 'co
```

load()

get entries for asset from API

class device42api.**Building**(*json=None*, *parent=None*, *api=None*)

create Building object

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> b = device42api.Building(api=api)
>>> b.name    = 'Test Building'
>>> b.address = 'somewhere in the city'
>>> b.notes   = 'destruction ongoing, leave the building immediatley'
```

```
>>> b.save()
{'msg': ['Building added/updated successfully', 3, 'TestBuilding', True, True], 'code': 0}
```

add_customField(*cf=None*)
add custom Fields to the object

```
>>> b = api.get_building('Building with CustomFields')
>>> cf = device42api.CustomField(api=api)
>>> cf.key      = 'bldid'
>>> cf.value    = 23
>>> cf.value2   = 44
>>> cf.notes    = 'Building ids: 23,44'
>>> b.add_customField(cf)
{'msg': ['custom key pair values added or updated', 3, 'Building with CustomFields'], 'code'
```

class device42api.**CustomField**(*json=None*, *parent=None*, *api=None*)
create CustomField

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> b = device42api.Building(api=api)
>>> b.name    = 'Building with CustomFields'
>>> cf1 = device42api.CustomField(api=api)
>>> cf1.key  = 'created'
>>> cf1.type = 'date'
>>> cf1.value = '2014-04-02'
>>> cf1._api_path = 'building'
>>> cf1.name = b.name
>>> cf1.save()
{'msg': ['custom key pair values added or updated', 1, 'Building with CustomFields'], 'code': 0}
```

class device42api.**CustomFieldDevice**(*json=None*, *parent=None*, *api=None*)

Hint: special handling as API path changes for device custom fields

---

class device42api.**Customer**(*json=None*, *parent=None*, *api=None*)
create Customer

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> c = device42api.Customer(api=api)
>>> c.name = 'device42 Support'
>>> c.contact_info = 'device42 Support Team'
>>> c.save()
{'msg': ['Customer added or updated.', 1, 'device42 Support', True, True], 'code': 0}
```

Note: to get Contact details ready you need to switch to the GUI and create the appropriate type definitions ...

```
>>> c = device42api.Customer(api.get_customer('device42 Support'), api=api)
>>> c.customer  = c.name
>>> c.name      = 'Helpdesk'
>>> c.type      = 'Helpdesk' # this needs to be created in the GUI there's no API call for it
>>> c.email     = 'helpdesk@device42.com'
>>> c.phone     = '111-111-111'
>>> c.address   = 'Helpdesk Office Building 1'
>>> c.save()
{'msg': ['customer contact record added/updated successfully', 1, 'Helpdesk1'], 'code': 0}
```

class device42api.**Device**(*json=None*, *parent=None*, *api=None*)

create Device object

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> d = device42api.Device(api=api)
>>> d.name = 'TestDevice'
>>> d.serial_no = 'Ab123asd' # serial number must follow certain structure ???
>>> d.hardware = 'Generic Hardware 1U'
>>> d.in_service = 'yes'
>>> d.type = 'physical'
>>> d.service_level = 'production'
>>> d.os = 'RHEL Server'
>>> d.osver = 6.5
>>> d.memory = 16.000
>>> d.cpucount = 80
>>> d.cpucore = 8
>>> d.notes = 'my special device'
>>> d.save()
{'msg': ['device added or updated', 156, 'TestDevice', True, True], 'code': 0}
```

add_customField(*cf=None*)

add custom Fields to the object

```
>>> device = api.get_device(device_id=1)
>>> cf = device42api.CustomFieldDevice(api=api)
>>> cf.key      = 'used_since'
>>> cf.type     = 'date'
>>> cf.value    = '2014-04-02'
>>> device.add_customField(cf)
{'msg': ['custom key pair values added or updated', 1, 'Device with CustomFields'], 'code':
```

add_ip(*ipAddress=None*, *macAddress=None*)

adds an ipAddress to the device

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> racks = api.get_rack()
>>> racks[0].devices.keys()
[1.0]
>>> d = racks[0].devices[1.0]
>>> len(d.mac_addresses)
1
>>> d.add_ip('2.2.2.2') # if macAddress is ommited and only one macAddress is in device this
{'msg': ['mac address successfully added/updated', 2, '00:00:00:00:00:02', True, True], 'coc
>>> for ip in d.ip_addresses:
...     print ip.ipaddress
1.1.1.1
2.2.2.2
```

add_mac(*macAddress=None*, *port_name=None*)

adds a macAddress to the device (if new macAddress will be created)

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> d = device42api.Device(api=api)
>>> d.device_id = 1
>>> d.load()
>>> d.add_mac('00:00:00:00:00:02', 'eth1')
{'msg': ['mac address successfully added/updated', 2, '00:00:00:00:00:02', True, True], 'coc
```

load()

get entries for asset from API

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> d = device42api.Device(api=api)
>>> d.device_id = 156
>>> d.load()
>>> d.name, d.serial_no
('TestDevice', 'Ab123asd')
```

class device42api.**Device42API**(*host=None*, *port=443*, *username=None*, *password=None*, *noInit=False*)

API abstraction class this object deals with the https request to the device42 service

```
>>> api = device42api.Device42API(host='192.168.122.200', username='admin', password='changeme')
>>> for r in api.get_rack():
...     print r
...
TestRack1
TestRack2
```

low level methods:

- __get_api__(path='.../') # everythin after /api/1.0/
- __post_api__(path='.../', v='1.0', body=dict()) # v='1.0' or None
- __put_api__(path='.../', body=dict()) # currently not used

get_asset(*name=None*, *reload=False*)

return all assets from device42

```
>>> api.get_asset()
[<device42api.Asset object at 0x26a0b50>, <device42api.Asset object at 0x26a8450>]
>>> for a in api.get_assets():
...     print a, a.asset_id
...
<device42api.Asset object at 0x26aaa50> 1
<device42api.Asset object at 0x26a8590> 2
>>> api.get_asset('Asset with CustomFields')
<device42api.Asset object at 0x1c5e410>
```

get_building(*name=None*, *reload=False*)

return Building object from API if found otherwise False

```
>>> api.get_building('device42 Support')
<device42api.Building object at 0x1887d90>
>>> api.get_building('White House - Oval Office')
False
```

get_customer(*name=None*, *reload=False*)

return Customer object from API if found otherwise False

```
>>> api.get_customer('device42 Support')
<device42api.Customer object at 0x1887a10>
>>> api.get_customer('unknown')
False
>>> for con in api.get_customer('device42 Support').Contacts:
...     print con
{'phone': '111-111-111', 'address': 'Helpdesk Office 1', 'type': 'Helpdesk', 'email': 'helpc
```

get_device(*name=None*, *device_id=None*, *serial=None*)

return the Device from the API classified by

- name

- device_id
- serial

> **Attention:** return by name isn't working with device names including spaces (or anything which requires quoting) as the API always responses with 404 NOT FOUND. You're seeing this error because you have DEBUG Enabled in your settings

get_history()
return History records from API

```
>>> for h in api.get_history():
...     print h
2014-04-04T10:16:46.776Z Add/Change(API) admin building
```

get_macid_byAddress(*macAddress=None*, *reload=False*)
return IPAM_macaddress object from API if found otherwise False

```
>>> api.get_macid_byAddress('11:11:11:11:22:01')
<device42api.IPAM_macaddress object at 0x26a0a50>
>>> api.get_macid_byAddress('11:11:11:11:22:01').macaddress_id
3
```

get_patch_panel_modules()
return all patch panels from device42, use get_assets and validate patch_panel_model_id field

```
>>> api.get_patch_panels()
[<device42api.Asset object at 0x26b2450>]
>>> for p in api.get_patch_panels():
...     print p, p.asset_id
...
<device42api.Asset object at 0x26a8150> 2
```

get_patch_panels()
return all patch panels from device42, use get_assets and validate patch_panel_model_id field

```
>>> api.get_patch_panels()
[<device42api.Asset object at 0x26b2450>]
>>> for p in api.get_patch_panels():
...     print p, p.asset_id
...
<device42api.Asset object at 0x26a8150> 2
```

get_pdu_models()
return all PDU models from device42

```
>>> api.get_pdu_models()
[<device42api.PDU_Model object at 0x26a0bd0>]
>>> for m in api.get_pdu_models():
...     print m
...
pdu_model 1 ports 8 type NEMA 5-15R
```

get_rack(*name=None*, *building=None*, *room=None*)
return all racks from device42

```
>>> api.get_rack('TestRack1')
[<device42api.Rack object at 0x26a0dd0>]
>>> for r in api.get_rack():
...     for d in r.devices.values():
```

```
...             print u'device: %s id: %s' % (d.name, d.device_id)
...
device: Test Device id: 1
>>>
>>> api.get_rack(room='Test Room')
```

get_room(*name=None*, *reload=False*)
return Room object from API if found otherwise False

```
>>> api.get_room('device42 Support')
<device42api.Building object at 0x1887d90>
>>> api.get_room('Oval Office')
False
```

get_service_level(*name=None*, *reload=False*)
return ServiceLevel object from API if found otherwise False

```
>>> api.get_service_level('Production')
<device42api.ServiceLevel object at 0x1aa4310>
>>> print api.get_service_level('Production')
Production(1)
```

class device42api.**Device42APIObject**(*json=None*, *parent=None*, *api=None*)
basic Object representing a device42 API object, inherit from this one and implement at least:

- save()
- load()
- get_json()

class device42api.**Hardware**(*json=None*, *parent=None*, *api=None*)
create Hardware object

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> h = device42api.Hardware(api=api)
>>> h.name = 'TestHardware'
>>> h.type = 1  # 1=Regular,2=Blade,3=Other
>>> h.size = 1
>>> h.depth = 1 # 1=Full,2=Half
>>> h.notes = 'my test hardware'
>>> h.save()
{'msg': ['hardware model added or updated', 25, 'TestHardware', True, True], 'code': 0}
```

class device42api.**History**(*json=None*, *parent=None*, *api=None*)
only representing the History as object

Note: !!! can only be retrieved !!!

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> for h in api.get_history():
...     print h
2014-04-04T10:16:46.776Z Add/Change(API) admin building
```

class device42api.**IPAM_DNSRecord**(*json=None*, *parent=None*, *api=None*)
create IPAM DNSRecord

Note: the API and the device42 logic in the Backend don't provide sanity checking of DNS syntax/recomendations for data, in addition to the problem of validation, the DNS Zone needs to be created through the GUI

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> i = device42api.IPAM_DNSRecord(api=api)
>>> i.domain     = 'localdomain'
>>> i.type       = 'CNAME'
>>> i.nameserver = '127.0.0.1'
>>> i.name       = 'localhost'
>>> i.content    = '127.0.0.1'
>>> i.ttl        = 86400
>>> i.save()
{'msg': ['DNS record added/updated successfully', 1, 'localhost'], 'code': 0}
>>> i2 = device42api.IPAM_DNSRecord(api=api)
>>> i2.domain     = 'localdomain'
>>> i2.nameserver = '127.0.0.1'
>>> i2.name       = 'localhost'
>>> i2.content    = '127.0.0.2'
>>> i2.ttl        = 86400
>>> i2.type       = 'CNAME'
>>> i2.save()
{'msg': ['DNS record added/updated successfully', 2, 'localhost'], 'code': 0}
```

class device42api.**IPAM_ipaddress**(*json=None*, *parent=None*, *api=None*)

create IPAM ipaddress these objects are returned if you fetch devices with configured macAddresses, manual adding a mac Address to a device as follows ...

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> i = device42api.IPAM_ipaddress(api=api)
>>> i.ipaddress = '1.1.1.1' # 127.0.0.1 is not supported and will lead to {'msg': 'list index ou
>>> i.macaddress = '00:11:22:33:44:55'
>>> i.device = 'Test Device'
>>> i.type = 'static' # static or dhcp
>>> i.get_json()
{'device': 'Test Device', 'macaddress': '00:11:22:33:44:55', 'ipaddress': '1.1.1.1', 'type': 'st
>>> i.save()
{'msg': ['ip added or updated', 1, '1.1.1.1', True, True], 'code': 0}
```

load()

there's nothing to be loaded for now

save_dnsRecord(*nameserver=None*, *ttl=86400*)

saves the A DNS record for the device and the IP

Note: I've not added any DNS logic as the API and GUI suffer support and I'm anoyed if implementing this over and over. Since I didn't want to introduce dependencies to other python Modules there's no logic in here

---

> **Attention:** the DNS Zones must exist and need to be created through the GUI

```
>>> d = api.get_device('TestDevice')
>>> # since the device doesn't carry a valid FQDN set it accordingly !
>>> d.name = 'testdevice.localdomain'
testdevice.localdomain
>>> i = d.ip_address[0]
>>> i.ipaddress
1.1.1.1
>>> i.save_dnsRecord()
```

```
[{'msg': ['DNS record added/updated successfully', 1, 'testdevice.localdomain'], 'code': 0},
 {'msg': ['DNS record added/updated successfully', 2, '1.1.1.1.in-addr.arpa'], 'code': 0}]
```

Attention: when changing the parent device in any way remember that there's no logic which removes the old entries, so you end up with multiple entries for the address

```
>>> d = api.get_device('TestDevice')
>>> d.name = 'testdevice2.localdomain'
>>> i = d.ip_address[0]
>>> i.save_dnsRecord()
[{'msg': ['DNS record added/updated successfully', 3, 'testdevice2.localdomain'], 'code': 0}
 {'msg': ['DNS record added/updated successfully', 4, '1.1.1.1.in-addr.arpa'], 'code': 0}]
```

class device42api.**IPAM_macaddress**(*json=None*, *parent=None*, *api=None*)

create IPAM macaddress these objects are returned if you fetch devices with configured macAddresses, manual adding a mac Address to a device as follows ...

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> i = device42api.IPAM_macaddress(api=api)
>>> i.macaddress = '00:11:22:33:44:55'
>>> i.port_name = 'eth0'
>>> i.device = 'Test Device'
>>> i.get_json()
{'device': 'Test Device', 'macaddress': '00:11:22:33:44:55', 'port_name': 'eth0'}
>>> i.save()
{'msg': ['mac address successfully added/updated', 1, '00:11:22:33:44:55', True, True], 'code':
```

class device42api.**IPAM_subnet**(*json=None*, *parent=None*, *api=None*)

create IPAM subnet

```
>>> api = device42api.Device42API(host='192.168.122.102', username='admin', password='admin')
>>> sub = device42api.IPAM_subnet(api=api)
>>> sub.network = '1.1.1.0'
>>> sub.mask_bits = 24
>>> sub.name    = 'Home Servers'
>>> sub.gateway = '1.1.1.254'
>>> sub.save()
{'msg': ['subnet successfully added/updated', 1, 'Home Servers-1.1.1.0/24'], 'code': 0}
```

class device42api.**IPAM_switch**(*json=None*, *parent=None*, *api=None*)

create IPAM switch postponed

class device42api.**IPAM_switchport**(*json=None*, *parent=None*, *api=None*)

create IPAM switchport

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> sp = device42api.IPAM_switchport(api=api)
>>> sp.port = 1
>>> sp.vlan_ids = '1'
>>> sp.description = 'Test Port'
>>> sp.up = 'yes'
>>> sp.up_admin = 'no'
>>> sp.count = 'yes'
>>> sp.save()
{'msg': ['switchport successfully added/updated', 7, '1'], 'code': 0}
```

Attention: !!! API Bug !!! even with the switchport_id given the API adds a new port

```
>>> sp.get_json()
{'switchport_id': 7, 'port': 7}
>>> sp.save()
{'msg': ['switchport successfully added/updated', 9, '7'], 'code': 0}
```

class device42api.**IPAM_vlan**(*json=None*, *parent=None*, *api=None*)

create IPAM subnet

```
>>> v = device42api.IPAM_vlan(api=api)
>>> v.number = 1
>>> v.name = 'Default VLAN'
>>> v.save()
{'msg': ['vlan successfully added', 1, 'Default VLAN', True], 'code': 0}
```

class device42api.**PDU**(*json=None*, *parent=None*, *api=None*)

create Rack object

Note: !!! the PDU Model needs to exist an unfortunatley there's no API call to create it !!!

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> pdu_models = api.get_pdu_models()
>>> p = device42api.PDU(api=api)
>>> p.name = 'Test PDU'
>>> p.pdu_id = 1
>>> p.rack_id = 80
>>> p.device = 156
>>> p.notes = 'Test PDU Test Device'
>>> p.where = 'left'
>>> p.start_at = 1
>>> p.save()
{'msg': '{'pdu': [u"Model PDU with pk u'1' does not exist."]}', 'code': 1}
```

Note: !!! that might be an API bug ? updating after adding it in the GUI works

---

```
>>> p.save()
{'msg': ['PDU Rack Info successfully added/edited.', 1, 'PDU Test'], 'code': 0}
>>> r = api.get_rack()
>>> r[0].pdus
[{'start_at': 1.0, 'name': 'PDU Test', 'orientation': 'Front', 'pdu_id': 1, 'depth': 'Full Depth
```

class device42api.**PDU_Model**(*json=None*, *parent=None*, *api=None*)

only representing the PDU Models as object

Note: !!! since there's no API call to create/update these can only be retrieved !!!

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> models = api.get_pdu_models()
>>> for m in models:
...     print m
pdu_model 1 ports 8 type NEMA 5-15R
```

class device42api.**PatchPanel**(*json=None*, *parent=None*, *api=None*)

create PatchPanel

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> cp = device42api.PatchPanel(api=api)
```

```
>>> p = api.get_patch_panels()[0]
>>> mac = api.get_macid_byAddress('00:00:00:00:00:01')
>>> cp.patch_panel_id = p.asset_id
>>> cp.number = 1
>>> cp.mac_id = mac.address_id
>>> cp.get_json()
{'patch_panel_id': 2, 'mac_id': 1, 'number': 1}
>>> cp.save()
{'msg': ['patch port details edited successfully.', 1, 'Test Panel : 1'], 'code': 0}
```

class device42api.**PatchPanelModule**(*json=None*, *parent=None*, *api=None*)

only representing the Patch Panel Modules as object

Note: !!! since there's no API call to create/update these can only be retrieved !!!

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> modules = api.get_patch_panel_modules()
>>> for m in modules:
...     print m
Test Panel Singular port_type=RJ45 ports_in_row=12 ports=24
```

class device42api.**Rack**(*json=None*, *parent=None*, *api=None*)

create Rack object

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> r = device42api.Rack(api=api)
>>> r.name = 'TestRack1'
>>> r.size = 42
>>> r.room = 'Test Room'
>>> r.building = 'TestBuilding'
>>> r.room_id = 2
>>> r.numbering_start_from_botton = 'no'
>>> r.notes = 'my personal rack'
>>> r.save()
{'msg': ['rack added/updated.', 80, 'TestRack1', True, True], 'code': 0}
```

add_customField(*cf=None*)

add custom Fields to the object

```
>>> rack = api.get_rack('Rack with CustomFields')[0]
>>> cf = device42api.CustomField(api=api)
>>> cf.key      = 'used_since'
>>> cf.type     = 'date'
>>> cf.value    = '2014-04-02'
>>> rack.add_customField(cf)
{'msg': ['custom key pair values added or updated', 3, 'Rack with CustomFields (in Room with
```

add_device(*device=None*, *start_at='auto'*)

add's a device to the rack starting at given position "start_at=xxx" or auto for next possible free slot

```
>>> # if created you need to set the rack_id first
>>> hw  = device42api.Hardware(api=api)
>>> hw.name, hw.type, hw.size, hw.depth = 'Generic Hardware 1U', 1, 1, 1
>>> h.save()
>>> dev = device42api.Device(api=api)
>>> dev.name = 'Test Device'
>>> dev.hardware = 'Generic Hardware 1U'
>>> dev.save()
```

```
>>> r.rack_id = 80
>>> r.add_device(dev, start_at=1)
{'msg': ['device added or updated in the rack', 1, '[1.0] - TestRack1 -Test Room'], 'code':
```

load()
get entries for rack from API

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> r = device42api.Rack(api=api)
>>> r.rack_id = 80
>>> r.load()
>>> r.name, r.notes
('TestRack1', 'my personal rack')
>>> r.devices
{32.0: <device42api.Device object at 0x991cd0>, 36.0: <device42api.Device object at 0x991b50
```

class device42api.**Room**(*json=None*, *parent=None*, *api=None*)
create Room object

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> r = device42api.Room(api=api)
>>> r.name      = 'Test Room'
>>> r.building  = 'Test Building'
>>> r.building_id = 3
>>> r.notes     = 'coffee corner for sysadmins'
>>> r.save()
{'msg': ['Room added/updated successfully', 2, 'Test Room', True, True], 'code': 0}
```

add_customField(*cf=None*)
add custom Fields to the object

```
>>> room = api.get_room('Room with CustomFields')
>>> cf = device42api.CustomField(api=api)
>>> cf.key      = 'used_since'
>>> cf.value    = '2014-04-02'
>>> cf.type     = 'date'
>>> room.add_customField(cf)
{'msg': ['custom key pair values added or updated', 3, 'Room with CustomFields @ Building wi
```

load()
get entries for room from API

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> r = device42api.Room(api=api)
>>> r.room_id = 2
>>> r.load()
>>> r.notes
'coffee corner for sysadmins'
>>> r.building
'TestBuilding'
```

class device42api.**ServiceLevel**(*json=None*, *parent=None*, *api=None*)
only representing the ServiceLevels as object

Note: !!! since there's no API call to create/update these can only be retrieved !!!

```
>>> api = device42api.Device42API(host='127.0.0.1', username='admin', password='changeme')
>>> sl = api.get_service_level('Production')
```

```
>>> print sl
Production(1)
>>> for sl in api.get_service_level():
...     print sl
QA
Development
Production
```

CHAPTER 2

Example usage

following steps are necessary (in order) to get a device deployed in a Rack/Room/Building.

- create Hardware representing our device(s)
- create a Building
- create a Room in our Building
- create a Rack in our Room
- create a Device
- add macAddress and ipAddress for the Device
- create a switch
- create a patch panel
- connect switch and panel and device

```
>>> import device42api
>>> api             = device42api.Device42API(host='localhost', username='admin', password='changeme'
>>> size            = 2
>>> # create Hardware for device
>>> hw              = device42api.Hardware(api=api)
>>> hw.name         = 'Generic Hardware %sU' % size
>>> hw.type         = 1
>>> hw.size         = size
>>> hw.depth        = 1
>>> hw.manufacturer = 'Test'
>>> hw.save()
{'msg': ['hardware model added or updated', 1, 'Generic Hardware 2U', True, True], 'code': 0}
>>> # create Building
>>> b               = device42api.Building(api=api)
>>> b.name          = 'Test Building'
>>> b.address       = 'somewhere in the city'
>>> b.notes         = 'destruction ongoing, leave the building immediatley'
>>> b.save()
{'msg': ['Building added/updated successfully', 1, 'Test Building', True, True], 'code': 0}
>>> # create a Room in our Building
>>> room                = device42api.Room(api=api)
>>> room.name           = 'Test Room'
>>> room.building       = b.name
>>> room.building_id    = b.building_id
>>> room.notes          = 'coffee corner for sysadmins'
>>> room.save()
```

```
{'msg': ['Room added/updated successfully', 1, 'Test Room', True, True], 'code': 0}
>>> # create a Rack in our Room
>>> rack            = device42api.Rack(api=api)
>>> rack.name       = 'Test Rack'
>>> rack.size       = 42
>>> rack.room       = room.name
>>> rack.building   = room.building
>>> rack.room_id    = room.room_id
>>> rack.numbering_start_from_botton = 'no'
>>> rack.notes         = 'First Rack'
>>> rack.save()
{'msg': ['rack added/updated.', 1, 'Test Rack', True, True], 'code': 0}
>>> # create a Device
>>> device                  = device42api.Device(api=api)
>>> device.name             = 'Test Device'
>>> device.serial_no        = '0123456789'
>>> device.hardware         = 'Generic Hardware 2U'
>>> device.in_service       = 'yes'
>>> device.type             = 'physical'
>>> device.service_level    = 'production'
>>> device.os               = 'RHEL Server'
>>> device.osver            = 6.5
>>> device.memory           = 16.000
>>> device.cpucount         = 80
>>> device.cpucore          = 8
>>> device.notes            = 'Test Device'
>>> device.save()
{'msg': ['device added or updated', 1, 'Test Device', True, True], 'code': 0}
>>> # add macAddress and ipAddress for the Device
>>> for mA, iA, dN in (('00:11:22:33:44:55', '1.1.1.1', 'eth0'),
...                    ('00:11:22:33:44:66', '1.1.1.2', 'eth1')):
...     device.add_mac(macAddress=mA, port_name=dN)
...     device.add_ip(ipAddress=iA, macAddress=mA)
True
True
True
True
>>> # validate mac and ip addresses
>>> for m in device.mac_addresses:      print m, m.macaddress, m.mac_id
...
<device42api.IPAM_macaddress object at 0x2272ad0> 00:11:22:33:44:55 1
<device42api.IPAM_macaddress object at 0x22728d0> 00:11:22:33:44:66 2
>>> for i in device.ip_addresses:       print i, i.ipaddress, i.ip_id
...
<device42api.IPAM_ipaddress object at 0x2272950> 1.1.1.1 1
<device42api.IPAM_ipaddress object at 0x22772d0> 1.1.1.2 2
>>> rack.add_device(device, start_at=1)
{'msg': ['device added or updated in the rack', 1, '[1.0] - Test Rack -Test Room'], 'code': 0}
>>> # create a switch
>>> switch                  = device42api.Device(api=api)
>>> switch.name             = 'switch1'
>>> switch.serial_no        = '9876543210'
>>> switch.hardware         = 'Generic Hardware 2U'
>>> switch.in_service       = 'yes'
>>> switch.type             = 'physical'
>>> switch.service_level    = 'production'
>>> switch.is_it_switch     = 'yes'
>>> switch.save()
```

```
{'msg': ['device added or updated', 2, 'switch1', True, True], 'code': 0}
>>> # since Hardware isn't fetch able in the API you need to remember the Hardware size if you want t
>>> rack.add_device(switch, start_at=rack.size - size)
{'msg': ['device added or updated in the rack', 1, '[40.0] - Test Rack -Test Room'], 'code': 0}
```

d