Ironic UI Documentation

Release 6.0.1.dev2

OpenStack Foundation

Sep 09, 2022

CONTENTS

CHAPTER

ONE

INTRODUCTION

The ironic UI is an OpenStack Horizon plugin that will allow users to view and manage their ironic bare metal nodes, ports and drivers.

The documentation provided here is continually kept up-to-date based on the latest code that has been committed, and may not represent the state of the project at any specific prior release.

For information on any current or prior version of Ironic, see the release notes.

For more information on ironic, see the ironic documentation.

1.1 ironic-ui installation guide

1.1.1 Ironic-UI Installation

Manual Installation

Please note that the following instructions assume that you have an existing installation of the OpenStack Horizon dashboard application. For Horizon installation please see https://docs.openstack.org/horizon/latest/contributor/quickstart.html.

1. Clone the Ironic UI repository:

```
git clone https://opendev.org/openstack/ironic-ui
```

2. Change into the root directory of your horizon installation and activate the python virtualenv. Example:

```
source .venv/bin/activate
```

Note: The `.venv` folder is pre-installed when horizon is setup with `./run_tests.sh`. Do not attempt to reinstall the virtual environment.

---

3. Copy the `_2200_ironic.py` file from `ironic_ui/enabled/_2200_ironic.py` file to `horizon/openstack_dashboard/local/enabled` directory. Example, set as if being executed from the root of the ironic-ui repository:

```
cp ./ironic_ui/enabled/_2200_ironic.py ../horizon/openstack_dashboard/
↪local/enabled
```

4. Change into the ironic-ui repository and package the plugin:

```
pip install -r requirements.txt -e .
```

   This will build and install the ironic-ui plugin into the active virtual environment associated with your horizon installation. The plugin is installed in editable mode as a link back to your ironic-ui plugin directory.

5. Change back into the horizon repository and bring up your environment:

```
./run_tests.sh --runserver
```

   The Bare Metal service should now be visible in the Horizon navigation.

6. Start the server in test mode with the `npm run test` command.

7. Access the test page in order to initate tests.

   http://localhost:8000/jasmine/?spec=horizon.dashboard.admin.ironic

Installation with DevStack

In order to use the Ironic UI with devstack, you will need to enable the UI plugin separately in your installation local.conf file.

This is done in a similar fashion to enabling Ironic for devstack.

Make sure you have horizon enabled, which is the default in devstack.

Then, enable the Ironic UI plugin appending the following line to the end of the local.conf file, just after Ironic plugin enablement:

```
enable_plugin ironic-ui https://github.com/openstack/ironic-ui
```

After this, you can run ./stack.sh from the devstack directory.

The Bare Metal Provisioning plugin should now be visible in the Horizon navigation.

1.1.2 Uninstallation

To uninstall, use `pip uninstall ironic-ui` from with-in the horizon virtual environment. You will also need to remove the `openstack_dashboard/enabled/_2200_ironic.py` file from the horizon installation.

1.2 Contributing to Ironic UI

If youre interested in contributing to the Ironic UI project, the following will help get you started.

1.2.1 How to Contribute

Contributor License Agreement

In order to contribute to the Ironic UI project, you need to have signed OpenStacks contributors agreement.

See also:

- https://docs.openstack.org/infra/manual/developers.html
- https://wiki.openstack.org/CLA

Project Hosting Details

Bug tracker https://storyboard.openstack.org/#!/project/952

Mailing list (prefix subjects with `[ironic]` for faster responses) http://lists.openstack.org/cgi-bin/mailman/listinfo/openstack-discuss

Code Hosting https://opendev.org/openstack/ironic-ui

Code Review https://review.opendev.org/#/q/status:open+project:openstack/ironic-ui,n,z

1.2.2 Autogenerated API Documentation

Source Code Index

Ironic_ui

The `ironic_ui` Module

The `ironic_ui.api.ironic` Module

`ironic_ui.api.ironic.`**`driver_details`**`(`*request*, *driver_name*`)`

Retrieve the details of a specified driver

Parameters

- **`request`** HTTP request
- **`driver_name`** Name of the driver

Returns dictionary of driver details

https://docs.openstack.org/python-ironicclient/latest/cli/osc/v1/index.html#baremetal-driver-show

ironic_ui.api.ironic.**driver_list**(*request*)

Retrieve a list of drivers.

Parameters **request** HTTP request.

Returns A list of drivers.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.driver.html#ironicclient.v1.driver.DriverManager.list

ironic_ui.api.ironic.**driver_properties**(*request*, *driver_name*)

Retrieve the properties of a specified driver

Parameters

- **request** HTTP request
- **driver_name** Name of the driver

Returns Property list

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.driver.html#ironicclient.v1.driver.DriverManager.properties

ironic_ui.api.ironic.**ironicclient**(*request*)

Returns a client connected to the Ironic backend.

Parameters **request** HTTP request.

Returns Ironic client.

ironic_ui.api.ironic.**node_create**(*request*, *params*)

Create a node

Parameters

- **request** HTTP request.
- **params** Dictionary of node parameters

ironic_ui.api.ironic.**node_delete**(*request*, *node_id*)

Delete a node from inventory.

Parameters

- **request** HTTP request.
- **node_id** The UUID or name of the node.

Returns node.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.node.html#ironicclient.v1.node.NodeManager.delete

ironic_ui.api.ironic.**node_get**(*request*, *node_id*)

Retrieve a node.

Parameters

- **request** HTTP request.
- **node_id** The UUID or name of the node.

Returns node.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.node.html#ironicclient.v1.node.NodeManager.get

ironic_ui.api.ironic.**node_get_boot_device**(*request*, *node_id*)

Get the boot device for a specified node.

Parameters

- **request** HTTP request.
- **node_id** The UUID or name of the node.

Returns Dictionary with keys boot_device and persistent

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.node.html#ironicclient.v1.node.NodeManager.get_boot_device

ironic_ui.api.ironic.**node_get_console**(*request*, *node_id*)

Get connection information for a nodes console.

Parameters

- **request** HTTP request.
- **node_id** The UUID or name of the node.

Returns Console connection information

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.node.html#ironicclient.v1.node.NodeManager.get_console

ironic_ui.api.ironic.**node_get_supported_boot_devices**(*request*, *node_id*)

Get the list of supported boot devices for a specified node.

Parameters

- **request** HTTP request.
- **node_id** The UUID or name of the node.

Returns List of supported boot devices (strings)

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.node.html#ironicclient.v1.node.NodeManager.get_boot_device

ironic_ui.api.ironic.**node_inject_nmi**(*request*, *node_id*)

Inject Non-Masking Interrupts into a specified node.

Parameters

- **request** HTTP request.
- **node_id** The UUID or name of the node.

Returns Empty response.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.node.html#ironicclient.v1.node.NodeManager.inject_nmi

ironic_ui.api.ironic.**node_list**(*request*)

Retrieve a list of nodes.

Parameters **request** HTTP request.

Returns A list of nodes.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.node.html#ironicclient.v1.node.NodeManager.list

ironic_ui.api.ironic.**node_list_ports**(*request*, *node_id*)

List all the ports on a given node.

Parameters

- **request** HTTP request.
- **node_id** The UUID or name of the node.

Returns A full list of ports. (limit=0)

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.node.html#ironicclient.v1.node.NodeManager.list_ports

ironic_ui.api.ironic.**node_set_boot_device**(*request*, *node_id*, *device*, *persistent*)

Set the boot device for a specified node.

Parameters

- **request** HTTP request.
- **node_id** The UUID or name of the node.
- **device** boot device.
- **persistent** True or False.

Returns null.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.node.html#ironicclient.v1.node.NodeManager.set_boot_device

ironic_ui.api.ironic.**node_set_console_mode**(*request*, *node_id*, *enabled*)

Start or stop the serial console for a given node.

Parameters

- **request** HTTP request.
- **node_id** The UUID or name of the node.
- **enabled** True to start the console, False to stop it

Returns node.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.node.html#ironicclient.v1.node.NodeManager.set_console_mode

ironic_ui.api.ironic.**node_set_maintenance**(*request*, *node_id*, *state*, *maint_reason=None*)

Set the maintenance mode on a given node.

Parameters

- **request** HTTP request.
- **node_id** The UUID or name of the node.
- **state** The maintenance state to set.

Returns node.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.node.html#ironicclient.v1.node.NodeManager.set_maintenance

ironic_ui.api.ironic.**node_set_power_state**(*request*, *node_id*, *state*, *soft=False*)

Set power state for a given node.

Parameters

- **request** HTTP request.
- **node_id** The UUID or name of the node.
- **state** the power state to set [on, off, reboot].
- **soft** flag for graceful power off or reboot

Returns node.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.node.html#ironicclient.v1.node.NodeManager.set_power_state

ironic_ui.api.ironic.**node_set_provision_state**(*request*, *node_id*, *state*, *cleansteps=None*)

Set the target provision state for a given node.

Parameters

- **request** HTTP request.
- **node_id** The UUID or name of the node.
- **state** the target provision state to set.
- **cleansteps** Optional list of cleaning steps

Returns node.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.node.html#ironicclient.v1.node.NodeManager.set_provision_state

ironic_ui.api.ironic.**node_set_raid_config**(*request*, *node_id*, *target_raid_config*)

Set target raid configuration for a given node.

Parameters

- **request** HTTP request.
- **node_id** The UUID or name of the node.
- **target_raid_config** Target raid configuration.

Returns Node.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.node.html#ironicclient.v1.node.NodeManager.set_target_raid_config

ironic_ui.api.ironic.**node_update**(*request*, *node_id*, *patch*)

Update a specified node.

Parameters

- **request** HTTP request.

- **node_id** The UUID or name of the node.
- **patch** Sequence of update operations

Returns node.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.node.html#ironicclient.v1.node.NodeManager.update

ironic_ui.api.ironic.**node_validate**(*request*, *node_id*)

Validate a specified node.

Parameters

- **request** HTTP request.
- **node_id** The UUID or name of the node.

Returns List of dictionaries, each containing an interface status

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.node.html#ironicclient.v1.node.NodeManager.validate

ironic_ui.api.ironic.**port_create**(*request*, *params*)

Create network port

Parameters

- **request** HTTP request
- **params** Port creation parameters

Returns Port

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.port.html#ironicclient.v1.port.PortManager.create

ironic_ui.api.ironic.**port_delete**(*request*, *port_uuid*)

Delete a network port

Parameters

- **request** HTTP request
- **port_uuid** Port uuid

Returns Port

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.port.html#ironicclient.v1.port.PortManager.delete

ironic_ui.api.ironic.**port_update**(*request*, *port_uuid*, *patch*)

Update a specified port.

Parameters

- **request** HTTP request.
- **port_id** The UUID of the port.
- **patch** Sequence of update operations

Returns Port.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.port.html#ironicclient.v1.port.PortManager.update

ironic_ui.api.ironic.**portgroup_create**(*request*, *params*)

Create a portgroup.

Parameters

- **request** HTTP request.
- **params** Portgroup creation parameters.

Returns Portgroup.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.portgroup.html#ironicclient.v1.portgroup.PortgroupManager.create

ironic_ui.api.ironic.**portgroup_delete**(*request*, *portgroup_id*)

Delete a portgroup from the DB.

Parameters

- **request** HTTP request.
- **portgroup_id** The UUID or name of the portgroup.

Returns Portgroup.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.portgroup.html#ironicclient.v1.portgroup.PortgroupManager.delete

ironic_ui.api.ironic.**portgroup_get_ports**(*request*, *portgroup_id*)

Get the ports associated with a specified portgroup.

Parameters

- **request** HTTP request.
- **portgroup_id** The UUID or name of the portgroup.

Returns List of ports.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.portgroup.html#ironicclient.v1.portgroup.PortgroupManager.list_ports

ironic_ui.api.ironic.**portgroup_list**(*request*, *node_id*)

List the portgroups associated with a given node.

Parameters

- **request** HTTP request.
- **node_id** The UUID or name of the node.

Returns A full list of portgroups. (limit=0)

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.portgroup.html#ironicclient.v1.portgroup.PortgroupManager.list_portgroups

ironic_ui.api.ironic.**portgroup_update**(*request*, *portgroup_id*, *patch*)

Update a specified portgroup.

Parameters

- **request** HTTP request.
- **portgroup_id** The UUID or name of the portgroup.
- **patch** Sequence of update operations

Returns Portgroup.

http://docs.openstack.org/developer/python-ironicclient/api/ironicclient.v1.port.html#ironicclient.v1.portgroup.PortgroupManager.update

The ironic_ui.api Module

The ironic_ui.api.ironic_rest_api Module

class ironic_ui.api.ironic_rest_api.**BootDevice**(***kwargs*)

Bases: django.views.generic.base.View

get(*request*, *node_id*)

Get the boot device for a specified node

Parameters

- **request** HTTP request.
- **node_id** Node name or uuid.

Returns Dictionary with keys boot_device and persistent.

put(*request*, *node_id*)

Set the boot device for a specific node

Parameters

- **request** HTTP request.
- **node_id** Node name or uuid.

Returns null.

url_regex = 'ironic/nodes/(?P<node_id>[a-zA-Z0-9-._~]+)/boot_device$'

class ironic_ui.api.ironic_rest_api.**DriverDetails**(***kwargs*)

Bases: django.views.generic.base.View

get(*request*, *driver_name*)

Get the details of a specified driver

Parameters

- **request** HTTP request
- **driver_name** Driver name

Returns Dictionary of details

url_regex = 'ironic/drivers/(?P<driver_name>[0-9a-zA-Z_-]+)$'

class ironic_ui.api.ironic_rest_api.**DriverProperties**(***kwargs*)

Bases: django.views.generic.base.View

get(*request*, *driver_name*)

Get the properties associated with a specified driver

Parameters

- **request** HTTP request.
- **driver_name** Driver name.

Returns Dictionary of properties.

url_regex = 'ironic/drivers/(?P<driver_name>[0-9a-zA-Z_-]+)/properties$'

class ironic_ui.api.ironic_rest_api.**Drivers**(***kwargs*)

Bases: django.views.generic.base.View

get(*request*)

Get the list of drivers.

Parameters **request** HTTP request.

Returns List of drivers.

url_regex = 'ironic/drivers/$'

class ironic_ui.api.ironic_rest_api.**InjectNmi**(***kwargs*)

Bases: django.views.generic.base.View

put(*request*, *node_id*)

Inject Non-Masking Interrupts into a specified node.

Parameters

- **request** HTTP request.
- **node_id** Node name or uuid.

Returns Empty response.

url_regex =
'ironic/nodes/(?P<node_id>[a-zA-Z0-9-._~]+)/management/inject_nmi$'

class ironic_ui.api.ironic_rest_api.**Maintenance**(***kwargs*)

Bases: django.views.generic.base.View

delete(*request*, *node_id*)

Take a specified node out of the maintenance state

Parameters

- **request** HTTP request.
- **node_id** Node name or uuid.

Returns Return code.

patch(*request*, *node_id*)

Put a specified node into maintenance state

Parameters

- **request** HTTP request.
- **node_id** Node name or uuid.

Returns Return code.

url_regex = 'ironic/nodes/(?P<node_id>[a-zA-Z0-9-._~]+)/maintenance$'

class ironic_ui.api.ironic_rest_api.**Node**(***kwargs*)

Bases: django.views.generic.base.View

delete(*request*, *node_id*)

Delete an Ironic node from inventory

Parameters

- **request** HTTP request.
- **node_id** Node name or uuid.

get(*request*, *node_id*)

Get information on a specified node.

Parameters

- **request** HTTP request.
- **node_id** Node name or uuid.

Returns node.

patch(*request*, *node_id*)

Update an Ironic node.

Parameters

- **request** HTTP request.
- **node_id** Node name or uuid.

url_regex = 'ironic/nodes/(?P<node_id>[a-zA-Z0-9-._~]+)$'

class ironic_ui.api.ironic_rest_api.**NodePortgroups**(***kwargs*)

Bases: django.views.generic.base.View

get(*request*, *node_id*)

Get the list of portgroups associated with a specified node.

Parameters

- **request** HTTP request.
- **node_id** Node name or uuid.

Returns List of portgroups.

url_regex = 'ironic/nodes/(?P<node_id>[a-zA-Z0-9-._~]+)/portgroups$'

class ironic_ui.api.ironic_rest_api.**NodePorts**(***kwargs*)

Bases: django.views.generic.base.View

get(*request*, *node_id*)

Get the list of ports associated with a specified node.

Parameters

- **request** HTTP request.
- **node_id** Node name or uuid.

Returns List of ports.

url_regex = 'ironic/nodes/(?P<node_id>[a-zA-Z0-9-._~]+)/ports/detail$'

class ironic_ui.api.ironic_rest_api.**Nodes**(***kwargs*)

Bases: django.views.generic.base.View

get(*request*)

Get the list of nodes.

Parameters **request** HTTP request.

Returns List of nodes.

post(*request*)

Create an Ironic node.

Parameters **request** HTTP request.

url_regex = 'ironic/nodes/$'

class ironic_ui.api.ironic_rest_api.**Port**(***kwargs*)

Bases: django.views.generic.base.View

delete(*request*, *port_uuid*)

Delete a network port.

Parameters

- **request** HTTP request
- **port_uuid** Port uuid.

patch(*request*, *port_uuid*)

Update an Ironic port.

Parameters

- **request** HTTP request.
- **port_uuid** Port uuid.

url_regex = 'ironic/ports/(?P<port_uuid>[0-9a-f-]+)$'

class ironic_ui.api.ironic_rest_api.**Portgroup**(***kwargs*)

Bases: django.views.generic.base.View

delete(*request*, *portgroup_id*)

Delete a portgroup.

Parameters

- **request** HTTP request.
- **portgroup_id** UUID or name of portgroup.

patch(*request*, *portgroup_id*)

Update an Ironic portgroup.

Parameters

- **request** HTTP request.
- **portgroup_id** UUID or name of portgroup.

url_regex = 'ironic/portgroups/(?P<portgroup_id>[a-zA-Z0-9-._~]+)$'

class ironic_ui.api.ironic_rest_api.**PortgroupPorts**(***kwargs*)

Bases: django.views.generic.base.View

get(*request*, *portgroup_id*)

Get the ports for a specified portgroup.

Parameters

- **request** HTTP request.
- **portgroup_id** UUID or name of portgroup.

Returns List of port objects.

url_regex = 'ironic/portgroups/(?P<portgroup_id>[a-zA-Z0-9-._~]+)/ports$'

class ironic_ui.api.ironic_rest_api.**Portgroups**(***kwargs*)

Bases: django.views.generic.base.View

post(*request*)

Create a portgroup.

Parameters **request** HTTP request.

Returns Portgroup.

url_regex = 'ironic/portgroups$'

class ironic_ui.api.ironic_rest_api.**Ports**(***kwargs*)

Bases: django.views.generic.base.View

post(*request*)

Create a network port.

Parameters **request** HTTP request.

Returns Port

url_regex = 'ironic/ports/$'

class ironic_ui.api.ironic_rest_api.**RaidConfig**(***kwargs*)

Bases: django.views.generic.base.View

put(*request*, *node_id*)

Set the RAID configuration for a specified node.

Parameters

- **request** HTTP request.
- **node_id** Node name or node uuid

Returns None

url_regex = 'ironic/nodes/(?P<node_id>[a-zA-Z0-9-._~]+)/states/raid$'

class ironic_ui.api.ironic_rest_api.**StatesConsole**(***kwargs*)

Bases: django.views.generic.base.View

get(*request*, *node_id*)

Get connection information for the nodes console

Parameters

- **request** HTTP request.
- **node_id** Node name or uuid.

Returns Connection information.

put(*request*, *node_id*)

Start or stop the serial console.

Parameters

- **request** HTTP request.
- **node_id** Node name or uuid.

Returns Return code.

url_regex = 'ironic/nodes/(?P<node_id>[a-zA-Z0-9-._~]+)/states/console$'

class ironic_ui.api.ironic_rest_api.**StatesPower**(***kwargs*)

Bases: django.views.generic.base.View

patch(*request*, *node_id*)

Set the power state for a specified node.

Parameters

- **request** HTTP request.
- **node_id** Node name or uuid.

Returns Return code.

url_regex = 'ironic/nodes/(?P<node_id>[a-zA-Z0-9-._~]+)/states/power$'

class ironic_ui.api.ironic_rest_api.**StatesProvision**(***kwargs*)

Bases: django.views.generic.base.View

put(*request*, *node_id*)

Set the provision state for a specified node.

Parameters

- **request** HTTP request.
- **node_id** Node name or uuid.

Returns Return code.

url_regex = 'ironic/nodes/(?P<node_id>[a-zA-Z0-9-._~]+)/states/provision$'

class ironic_ui.api.ironic_rest_api.**SupportedBootDevices**(***kwargs*)

Bases: django.views.generic.base.View

get(*request*, *node_id*)

Get the list of supported boot devices for a specified node.

Parameters

- **request** HTTP request.
- **node_id** Node name or uuid.

Returns List of supported boot devices.

url_regex = 'ironic/nodes/(?P<node_id>[a-zA-Z0-9-._~]+)/boot_device/supported$'

class ironic_ui.api.ironic_rest_api.**Validate**(***kwargs*)

Bases: django.views.generic.base.View

get(*request*, *node_id*)

Validate a specified node

Parameters

- **request** HTTP request.
- **node_id** Node name or uuid.

Returns List of dictionaries of interface statuses.

url_regex = 'ironic/nodes/(?P<node_id>[a-zA-Z0-9-._~]+)/validate$'

The ironic_ui.content Module

The ironic_ui.content.ironic.urls Module

The ironic_ui.content.ironic.views Module

class ironic_ui.content.ironic.views.**DetailView**(***kwargs*)

Bases: django.views.generic.base.TemplateView

template_name = 'admin/ironic/node_detail.html'

class ironic_ui.content.ironic.views.**IndexView**(***kwargs*)

Bases: django.views.generic.base.TemplateView

template_name = 'admin/ironic/index.html'

The ironic_ui.content.ironic Module

The ironic_ui.content.ironic.panel Module

class ironic_ui.content.ironic.panel.**Ironic**

Bases: horizon.base.Panel

name = 'Ironic Bare Metal Provisioning'

permissions = ('openstack.roles.admin', 'openstack.services.baremetal')

slug = 'ironic'

The ironic_ui.enabled._2200_ironic Module

The ironic_ui.enabled Module

The ironic_ui.test.urls Module

The ironic_ui.test.settings Module

The ironic_ui.test Module

The ironic_ui.test.tests Module

The ironic_ui.test.tests.test_registration Module

class ironic_ui.test.tests.test_registration.**RegistrationTests**(*methodName='runTest'*)

Bases: openstack_dashboard.test.helpers.TestCase

test_registered()

The ironic_ui.test.integration.test_basic Module

class ironic_ui.test.integration.test_basic.**TestIronicDashboardInstalled**(**args*, ***kwargs*)

Bases: openstack_dashboard.test.integration_tests.helpers.AdminTestCase

test_ironic_bare_metal_provisioning_page_opened()

The ironic_ui.test.integration Module

The ironic_ui.test.integration.pages Module

The ironic_ui.test.integration.pages.admin Module

The ironic_ui.test.integration.pages.admin.system.ironicbaremetalprovisioningpage Module

class ironic_ui.test.integration.pages.admin.system.ironicbaremetalprovisioningpage.**Ironicba**

Bases: openstack_dashboard.test.integration_tests.pages.basepage.BaseNavigationPage

The ironic_ui.test.integration.pages.admin.system Module

INDEX

L