

API Documentation

API Documentation

August 3, 2012

Contents

Contents	1
1 Package Products.Zuul.routers	2
1.1 Modules	2
1.2 Variables	2
1.3 Class TreeRouter	2
1.3.1 Methods	3
2 Module Products.Zuul.routers.device	5
2.1 Variables	5
2.2 Class DeviceRouter	5
2.2.1 Methods	6
3 Module Products.Zuul.routers.jobs	36
3.1 Variables	36
3.2 Class JobsRouter	36
3.2.1 Methods	36
4 Module Products.Zuul.routers.messaging	37
4.1 Class MessagingRouter	37
4.1.1 Methods	37
5 Module Products.Zuul.routers.mibs	38
5.1 Variables	38
5.2 Class MibRouter	38
5.2.1 Methods	38
6 Module Products.Zuul.routers.nav	43
6.1 Class DetailNavRouter	43
6.1.1 Methods	43
7 Module Products.Zuul.routers.network	44
7.1 Variables	44
7.2 Class NetworkRouter	44
7.2.1 Methods	44
7.3 Class Network6Router	48
7.3.1 Methods	48

8	Module Products.Zuul.routers.process	49
8.1	Class ProcessRouter	49
8.1.1	Methods	49
9	Module Products.Zuul.routers.report	54
9.1	Variables	54
9.2	Class ReportRouter	54
9.2.1	Methods	54
10	Module Products.Zuul.routers.service	58
10.1	Class ServiceRouter	58
10.1.1	Methods	58
11	Module Products.Zuul.routers.settings	63
11.1	Class SettingsRouter	63
11.1.1	Methods	63
12	Module Products.Zuul.routers.template	64
12.1	Class TemplateRouter	64
12.1.1	Methods	64
13	Module Products.Zuul.routers.triggers	78
13.1	Variables	78
13.2	Class TriggersRouter	78
13.2.1	Methods	78
14	Module Products.Zuul.routers.zenpack	80
14.1	Variables	80
14.2	Class ZenPackRouter	80
14.2.1	Methods	80
15	Module Products.Zuul.routers.zep	82
15.1	Variables	82
15.2	Class EventsRouter	82
15.2.1	Methods	82

1 Package `Products.Zuul.routers`

Zenoss JSON API

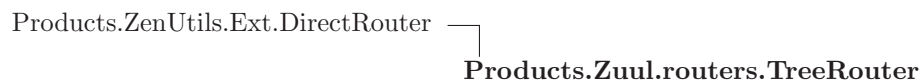
1.1 Modules

- **device**: Operations for Device Organizers and Devices.
(Section 2, p. 5)
- **jobs**: Operations for jobs.
(Section 3, p. 36)
- **messaging**: Operations for Messaging.
(Section 4, p. 37)
- **mibs**: Operations for MIBs.
(Section 5, p. 38)
- **nav**: Operations for Navigation
(Section 6, p. 43)
- **network**: Operations for Networks.
(Section 7, p. 44)
- **process**: Operations for Processes.
(Section 8, p. 49)
- **report**: Operations for Reports.
(Section 9, p. 54)
- **service**: Operations for Services.
(Section 10, p. 58)
- **settings**: Operations for Settings.
(Section 11, p. 63)
- **template**: Operations for Templates.
(Section 12, p. 64)
- **triggers** (Section 13, p. 78)
- **zenpack**: Operations for ZenPacks.
(Section 14, p. 80)
- **zep**: Operations for Events.
(Section 15, p. 82)

1.2 Variables

Name	Description
<code>log</code>	Value: <code>logging.getLogger(__name__)</code>

1.3 Class `TreeRouter`



A common base class for routers that have a hierarchical tree structure.

1.3.1 Methods

addNode (<i>self</i> , <i>type</i> , <i>contextUid</i> , <i>id</i> , <i>description=None</i>)	
Add a node to the existing tree underneath the node specified by the context UID	
Parameters	
type:	Either 'class' or 'organizer' (<i>type=string</i>)
contextUid:	Path to the node that will be the new node's parent (ex. /zport/dmd/Devices) (<i>type=string</i>)
id:	Identifier of the new node, must be unique in the parent context (<i>type=string</i>)
description:	(optional) Describes this new node (default: None) (<i>type=string</i>)
Return Value	
Marshaled form of the created node (<i>type=dictionary</i>)	

deleteNode (<i>self</i> , <i>uid</i>)	
Deletes a node from the tree.	
NOTE: You can not delete a root node of a tree	
Parameters	
uid:	Unique identifier of the node we wish to delete (<i>type=string</i>)
Return Value	
Properties:	
<ul style="list-style-type: none"> • msg: (string) Status message (<i>type=DirectResponse</i>)	

moveOrganizer (<i>self</i> , <i>targetUid</i> , <i>organizerUid</i>)	
Move the organizer uid to be underneath the organizer specified by the targetUid.	
Parameters	
targetUid:	New parent of the organizer (<i>type=string</i>)
organizerUid:	The organizer to move (<i>type=string</i>)
Return Value	
Properties:	
<ul style="list-style-type: none"> • data: (dictionary) Moved organizer (<i>type=DirectResponse</i>)	

asyncGetTree(*self*, *id=None*, *additionalKeys=()*)

Server side method for asynchronous tree calls. Retrieves the immediate children of the node specified by "id"

NOTE: our convention on the UI side is if we are asking for the root node then return the root and its children otherwise just return the children

Parameters

id: The uid of the node we are getting the children for
(*type=string*)

Return Value

Object representing the immediate children
(*type=[dictionary]*)

2 Module Products.Zuul.routers.device

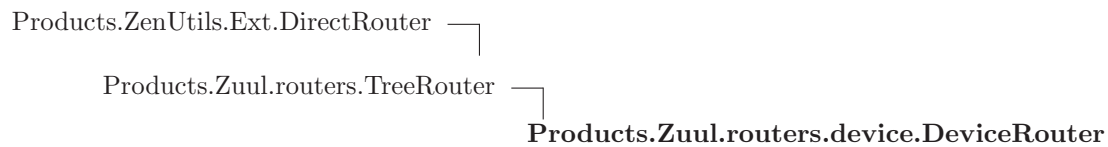
Operations for Device Organizers and Devices.

Available at: /zport/dmd/device_router

2.1 Variables

Name	Description
log	Value: logging.getLogger('zen.Zuul')

2.2 Class DeviceRouter



A JSON/ExtDirect interface to operations on devices

2.2.1 Methods

addLocationNode(*self, type, contextUid, id, description=None, address=None*)

Adds a new location organizer specified by the parameter *id* to the parent organizer specified by *contextUid*.

contextUid must be a path to a Location.

Parameters

- type:** Node type (always 'organizer' in this case)
(*type=string*)
- contextUid:** Path to the location organizer that will be the new node's parent
(ex. /zport/dmd/Devices/Locations)
(*type=string*)
- id:** The identifier of the new node
(*type=string*)
- description:** (optional) Describes the new location
(*type=string*)
- address:** (optional) Physical address of the new location
(*type=string*)

Return Value

Properties:

- **success:** (bool) Success of node creation
- **nodeConfig:** (dictionary) The new location's properties

(*type=dictionary*)

getTree(*self, id*)

Returns the tree structure of an organizer hierarchy where the root node is the organizer identified by the *id* parameter.

Parameters

- id:** Id of the root node of the tree to be returned
(*type=string*)

Return Value

Object representing the tree

(*type=[dictionary]*)

```
getComponents(self, uid=None, meta_type=None, keys=None, start=0, limit=50, page=0, sort='name', dir='ASC', name=None)
```

Retrieves all of the components at a given UID. This method allows for pagination.

Parameters

- uid:** Unique identifier of the device whose components are being retrieved
(*type=string*)
- meta_type:** (optional) The meta type of the components to be retrieved (default: None)
(*type=string*)
- keys:** (optional) List of keys to include in the returned dictionary. If None then all keys will be returned (default: None)
(*type=list*)
- start:** (optional) Offset to return the results from; used in pagination (default: 0)
(*type=integer*)
- limit:** (optional) Number of items to return; used in pagination (default: 50)
(*type=integer*)
- sort:** (optional) Key on which to sort the return results; (default: 'name')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC')
(*type=string*)
- name:** (optional) Used to filter the results (default: None)
(*type=regex*)

Return Value**Properties:**

- **data:** (dictionary) The components returned
- **totalCount:** (integer) Number of items returned
- **hash:** (string) Hashcheck of the current component state (to check whether components have changed since last query)

(*type=DirectResponse*)

getComponentTree(*self*, *uid=None*, *id=None*)

Retrieves all of the components set up to be used in a tree.

Parameters

uid: Unique identifier of the root of the tree to retrieve

(*type=string*)

id: not used

(*type=string*)

Return Value

Component properties in tree form

(*type=[dictionary]*)

findComponentIndex(*self*, *componentUid*, *uid=None*, *meta_type=None*, *sort='name'*, *dir='ASC'*, *name=None*, ***kwargs*)

Given a component uid and the component search criteria, this retrieves the position of the component in the results.

Parameters

componentUid: Unique identifier of the component whose index to return

(*type=string*)

uid: Unique identifier of the device queried for components

(*type=string*)

meta_type: (optional) The meta type of the components to retrieve (default: None)

(*type=string*)

sort: (optional) Key on which to sort the return results (default: 'name')

(*type=string*)

dir: (optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC')

(*type=string*)

name: (optional) Used to filter the results (default: None)

(*type=regex*)

Return Value

Properties:

- **index:** (integer) Index of the component

(*type=DirectResponse*)

getForm (<i>self</i> , <i>uid</i>)
Given an object identifier, this returns all of the editable fields on that object as well as their ExtJs xtype that one would use on a client side form.
Parameters
<i>uid</i> : Unique identifier of an object (<i>type=string</i>)
Return Value
Properties
<ul style="list-style-type: none">form: (dictionary) form fields for the object
(<i>type=DirectResponse</i>)

getInfo (<i>self</i> , <i>uid</i> , <i>keys=None</i>)
Get the properties of a device or device organizer
Parameters
<i>uid</i> : Unique identifier of an object (<i>type=string</i>)
<i>keys</i> : (optional) List of keys to include in the returned dictionary. If None then all keys will be returned (default: None) (<i>type=list</i>)
Return Value
Properties
<ul style="list-style-type: none">data: (dictionary) Object propertiesdisabled: (bool) If current user doesn't have permission to use setInfo
(<i>type=DirectResponse</i>)

setInfo (<i>self</i> , **data)
Set attributes on a device or device organizer. This method accepts any keyword argument for the property that you wish to set. The only required property is "uid".
Parameters
<i>uid</i> : Unique identifier of an object (<i>type=string</i>)
Return Value
DirectResponse

setProductInfo(*self*, *uid*, ****data**)

Sets the ProductInfo on a device. This method has the following valid keyword arguments:

Parameters

uid: Unique identifier of a device
(*type=string*)

hwManufacturer: Hardware manufacturer
(*type=string*)

hwProductName: Hardware product name
(*type=string*)

osManufacturer: Operating system manufacturer
(*type=string*)

osProductName: Operating system product name
(*type=string*)

Return Value

DirectResponse

getDeviceUidsByName(*self*, *query=""*, *start=0*, *limit=25*, *page=1*)

Retrieves a list of device uids. For use in combos.

getDeviceUids(*self*, *uid*)

Return a list of device uids underneath an organizer. This includes all the devices belonging to an child organizers.

Parameters

uid: Unique identifier of the organizer to get devices from
(*type=string*)

Return Value**Properties:**

- devices: (list) device uids

(*type=DirectResponse*)

```
getDevices(self, uid=None, start=0, params=None, limit=50, sort='titleOrId',  
page=None, dir='ASC', keys=None)
```

Retrieves a list of devices. This method supports pagination.

Parameters

- uid:** Unique identifier of the organizer to get devices from
(*type=string*)
- start:** (optional) Offset to return the results from; used in pagination (default: 0)
(*type=integer*)
- params:** (optional) Key-value pair of filters for this search. Can be one of the following: name, ipAddress, deviceClass, or productionState (default: None)
(*type=dictionary*)
- limit:** (optional) Number of items to return; used in pagination (default: 50)
(*type=integer*)
- sort:** (optional) Key on which to sort the return results (default: 'name')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC')
(*type=string*)

Return Value**Properties:**

- **devices:** (list) Dictionaries of device properties
- **totalCount:** (integer) Number of devices returned
- **hash:** (string) Hashcheck of the current device state (to check whether devices have changed since last query)

(*type=DirectResponse*)

```
renameDevice(self, uid, newId)
```

Set the device specified by the uid,"uid" to have the the id "newId" This will raise an exception if it fails.

Parameters

- uid:** The unique id of the device we are renaming
(*type=string*)
- newId:** string of the new id
(*type=string*)

```
moveDevices(self, uids, target, hashcheck=None, ranges=(), uid=None, params=None,
sort='name', dir='ASC', asynchronous=True)
```

Moves the devices specified by uids to the organizer specified by 'target'.

Parameters

- uids:** List of device uids to move
(*type=string*)
- target:** Uid of the organizer to move the devices to
(*type=string*)
- hashcheck:** Hashcheck for the devices (from getDevices())
(*type=string*)
- ranges:** (optional) List of two integers that are the min/max values of a range of uids to include (default: None)
(*type=integer*)
- uid:** (optional) Organizer to use when using ranges to get additional uids (default: None)
(*type=string*)
- params:** (optional) Key-value pair of filters for this search. Can be one of the following: name, ipAddress, deviceClass, or productionState (default: None)
(*type=dictionary*)
- sort:** (optional) Key on which to sort the return result (default: 'name')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC')
(*type=string*)

Return Value**Properties:**

- tree: ([dictionary]) Object representing the new device tree
- exports: (integer) Number of devices moved

(*type=DirectResponse*)

```
pushChanges(self, uids, hashcheck, ranges=(), uid=None, params=None, sort='name', dir='ASC')
```

Push changes on device(s) configuration to collectors.

Parameters

- uids:** List of device uids to push changes
(*type=string*)
- hashcheck:** Hashcheck for the devices (from getDevices())
(*type=string*)
- ranges:** (optional) List of two integers that are the min/max values of a range of uids to include (default: None)
(*type=integer*)
- uid:** (optional) Organizer to use when using ranges to get additional uids (default: None)
(*type=string*)
- params:** (optional) Key-value pair of filters for this search. Can be one of the following: name, ipAddress, deviceClass, or productionState (default: None)
(*type=dictionary*)
- sort:** (optional) Key on which to sort the return result (default: 'name')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC')
(*type=string*)

Return Value

- Success message
(*type=DirectResponse*)

```
lockDevices(self, uids, hashcheck, ranges=(), updates=False, deletion=False,
sendEvent=False, uid=None, params=None, sort='name', dir='ASC')
```

Lock device(s) from changes.

Parameters

- uids:** List of device uids to lock
(*type=string*)
- hashcheck:** Hashcheck for the devices (from getDevices())
(*type=string*)
- ranges:** (optional) List of two integers that are the min/max values of a range of uids to include (default: None)
(*type=integer*)
- updates:** (optional) True to lock device from updates (default: False)
(*type=boolean*)
- deletion:** (optional) True to lock device from deletion (default: False)
(*type=boolean*)
- sendEvent:** (optional) True to send an event when an action is blocked by locking (default: False)
(*type=boolean*)
- uid:** (optional) Organizer to use when using ranges to get additional uids (default: None)
(*type=string*)
- params:** (optional) Key-value pair of filters for this search. Can be one of the following: name, ipAddress, deviceClass, or productionState (default: None)
(*type=dictionary*)
- sort:** (optional) Key on which to sort the return result (default: 'name')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC')
(*type=string*)

Return Value

- Success or failure message
(*type=DirectResponse*)

```
resetIp(self, uids, hashcheck, uid=None, ranges=(), params=None, sort='name',
dir='ASC', ip='')
```

Reset IP address(es) of device(s) to the results of a DNS lookup or a manually set address

Parameters

- uids:** List of device uids with IP's to reset
(*type=string*)
- hashcheck:** Hashcheck for the devices (from getDevices())
(*type=string*)
- uid:** (optional) Organizer to use when using ranges to get additional uids (default: None)
(*type=string*)
- ranges:** (optional) List of two integers that are the min/max values of a range of uids to include (default: None)
(*type=integer*)
- params:** (optional) Key-value pair of filters for this search. Can be one of the following: name, ipAddress, deviceClass, or productionState (default: None)
(*type=dictionary*)
- sort:** (optional) Key on which to sort the return result (default: 'name')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC')
(*type=string*)
- ip:** (optional) IP to set device to. Empty string causes DNS lookup (default: "")
(*type=string*)

Return Value

- Success or failure message
(*type=DirectResponse*)


```
resetCommunity(self, uids, hashcheck, uid=None, ranges=(), params=None,
sort='name', dir='ASC')
```

Reset SNMP community string(s) on device(s)

Parameters

- uids:** List of device uids to reset
(*type=string*)
- hashcheck:** Hashcheck for the devices (from getDevices())
(*type=string*)
- uid:** (optional) Organizer to use when using ranges to get additional uids (default: None)
(*type=string*)
- ranges:** (optional) List of two integers that are the min/max values of a range of uids to include (default: None)
(*type=integer*)
- params:** (optional) Key-value pair of filters for this search. Can be one of the following: name, ipAddress, deviceClass, or productionState (default: None)
(*type=dictionary*)
- sort:** (optional) Key on which to sort the return result (default: 'name')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC')
(*type=string*)

Return Value

- Success or failure message
(*type=DirectResponse*)

```
setProductionState(self, uids, prodState, hashcheck, uid=None, ranges=(), params=None, sort='name', dir='ASC')
```

Set the production state of device(s).

Parameters

- uids:** List of device uids to set
(*type=string*)
- prodState:** Production state to set device(s) to.
(*type=integer*)
- hashcheck:** Hashcheck for the devices (from getDevices())
(*type=string*)
- uid:** (optional) Organizer to use when using ranges to get additional uids (default: None)
(*type=string*)
- ranges:** (optional) List of two integers that are the min/max values of a range of uids to include (default: None)
(*type=integer*)
- params:** (optional) Key-value pair of filters for this search. Can be one of the following: name, ipAddress, deviceClass, or productionState (default: None)
(*type=dictionary*)
- sort:** (optional) Key on which to sort the return result (default: 'name')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC')
(*type=string*)

Return Value

Success or failure message
(*type=DirectResponse*)

```
setPriority(self, uids, priority, hashcheck, uid=None, ranges=(), params=None,
            sort='name', dir='ASC')
```

Set device(s) priority.

Parameters

- uids:** List of device uids to set
(*type=string*)
- priority:** Priority to set device(s) to.
(*type=integer*)
- hashcheck:** Hashcheck for the devices (from getDevices())
(*type=string*)
- uid:** (optional) Organizer to use when using ranges to get additional uids (default: None)
(*type=string*)
- ranges:** (optional) List of two integers that are the min/max values of a range of uids to include (default: None)
(*type=integer*)
- params:** (optional) Key-value pair of filters for this search. Can be one of the following: name, ipAddress, deviceClass, or productionState (default: None)
(*type=dictionary*)
- sort:** (optional) Key on which to sort the return result (default: 'name')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC')
(*type=string*)

Return Value

Success or failure message
(*type=DirectResponse*)

```
setCollector(self, uids, collector, hashcheck, uid=None, ranges=(), params=None,
sort='name', dir='ASC', moveData=False, asynchronous=True)
```

Set device(s) collector.

Parameters

- uids:** List of device uids to set
(*type=string*)
- collector:** Collector to set devices to
(*type=string*)
- hashcheck:** Hashcheck for the devices (from getDevices())
(*type=string*)
- uid:** (optional) Organizer to use when using ranges to get additional uids (default: None)
(*type=string*)
- ranges:** (optional) List of two integers that are the min/max values of a range of uids to include (default: None)
(*type=integer*)
- params:** (optional) Key-value pair of filters for this search. Can be one of the following: name, ipAddress, deviceClass, or productionState (default: None)
(*type=dictionary*)
- sort:** (optional) Key on which to sort the return result (default: 'name')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC')
(*type=string*)

Return Value

Success or failure message
(*type=DirectResponse*)

```
setComponentsMonitored(self, uids, hashcheck, monitor=False, uid=None, ranges=(),
meta_type=None, keys=None, start=0, limit=50, sort='name', dir='ASC', name=None)
```

Set the monitoring flag for component(s)

Parameters

uids: List of component uids to set
(*type=string*)

hashcheck: Hashcheck for the components (from getComponents())
(*type=string*)

monitor: (optional) True to monitor component (default: False)
(*type=boolean*)

uid: (optional) Device to use when using ranges to get additional uids
(default: None)
(*type=string*)

ranges: (optional) List of two integers that are the min/max values of a
range of uids to include (default: None)
(*type=integer*)

meta_type: (optional) The meta type of the components to retrieve (default:
None)
(*type=string*)

keys: not used
(*type=string*)

start: (optional) Offset to return the results from; used in pagination
(default: 0)
(*type=integer*)

limit: (optional) Number of items to return; used in pagination (default:
50)
(*type=integer*)

sort: (optional) Key on which to sort the return result (default: 'name')
(*type=string*)

dir: (optional) Sort order; can be either 'ASC' or 'DESC' (default:
'ASC')
(*type=string*)

name: (optional) Component name to search for when loading ranges
(default: None)
(*type=string*)

Return Value

Success or failure message
(*type=DirectResponse*)

```
lockComponents(self, uids, hashcheck, uid=None, ranges=(), updates=False,
deletion=False, sendEvent=False, meta_type=None, keys=None, start=0, limit=50,
sort='name', dir='ASC', name=None)
```

Lock component(s) from changes.

Parameters

- uids:** List of component uids to lock
(*type=string*)
- hashcheck:** Hashcheck for the components (from getComponents())
(*type=string*)
- uid:** (optional) Device to use when using ranges to get additional uids
(default: None)
(*type=string*)
- ranges:** (optional) List of two integers that are the min/max values of a
range of uids to include (default: None)
(*type=integer*)
- updates:** (optional) True to lock component from updates (default: False)
(*type=boolean*)
- deletion:** (optional) True to lock component from deletion (default: False)
(*type=boolean*)
- sendEvent:** (optional) True to send an event when an action is blocked by
locking (default: False)
(*type=boolean*)
- meta_type:** (optional) The meta type of the components to retrieve (default:
None)
(*type=string*)
- keys:** not used
(*type=string*)
- start:** (optional) Offset to return the results from; used in pagination
(default: 0)
(*type=integer*)
- limit:** (optional) Number of items to return; used in pagination (default:
50)
(*type=integer*)
- sort:** (optional) Key on which to sort the return result (default: 'name')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC' (default:
'ASC')
(*type=string*)
- name:** (optional) Component name to search for when loading ranges
(default: None)
(*type=string*)

Return Value

Success or failure message
(*type=DirectResponse*)

```
deleteComponents(self, uids, hashcheck, uid=None, ranges=(), meta_type=None, keys=None, start=0, limit=50, sort='name', dir='ASC', name=None)
```

Delete device component(s).

Parameters

- uids:** List of component uids to delete
(*type=string*)
- hashcheck:** Hashcheck for the components (from getComponents())
(*type=string*)
- uid:** (optional) Device to use when using ranges to get additional uids (default: None)
(*type=string*)
- ranges:** (optional) List of two integers that are the min/max values of a range of uids to include (default: None)
(*type=integer*)
- meta_type:** (optional) The meta type of the components to retrieve (default: None)
(*type=string*)
- keys:** not used
(*type=string*)
- start:** (optional) Offset to return the results from; used in pagination (default: 0)
(*type=integer*)
- limit:** (optional) Number of items to return; used in pagination (default: 50)
(*type=integer*)
- sort:** (optional) Key on which to sort the return result (default: 'name')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC')
(*type=string*)
- name:** (optional) Component name to search for when loading ranges (default: None)
(*type=string*)

Return Value

- Success or failure message
(*type=DirectResponse*)

```
removeDevices(self, uids, hashcheck, action="remove", uid=None, ranges=(),
params=None, sort='name', dir='ASC', deleteEvents=False, deletePerf=False)
```

Remove/delete device(s).

Parameters

uids:	List of device uids to remove (<i>type=[string]</i>)
hashcheck:	Hashcheck for the devices (from getDevices()) (<i>type=string</i>)
action:	Action to take. 'remove' to remove devices from organizer uid, and 'delete' to delete the device from Zenoss. (<i>type=string</i>)
uid:	(optional) Organizer to use when using ranges to get additional uids and/or to remove device (default: None) (<i>type=string</i>)
ranges:	(optional) List of two integers that are the min/max values of a range of uids to include (default: None) (<i>type=[integer]</i>)
params:	(optional) Key-value pair of filters for this search. Can be one of the following: name, ipAddress, deviceClass, or productionState (default: None) (<i>type=dictionary</i>)
sort:	(optional) Key on which to sort the return result (default: 'name') (<i>type=string</i>)
dir:	(optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC') (<i>type=string</i>)
deleteEvents:	will remove all the events for the devices as well (<i>type=bool</i>)
deletePerf:	will remove all the perf data for the devices (<i>type=bool</i>)

Return Value

Properties:

- devtree: ([dictionary]) Object representing the new device tree
- grpree: ([dictionary]) Object representing the new group tree
- systree: ([dictionary]) Object representing the new system tree
- loctree: ([dictionary]) Object representing the new location tree

(*type=DirectResponse*)

<p>getGraphDefs(<i>self</i>, <i>uid</i>, <i>drange=None</i>)</p> <hr/> <p>Returns the url and title for each graph for the object passed in.</p> <p>Parameters</p> <p>uid: unique identifier of an object (<i>type=string</i>)</p>
<p>setZenProperty(<i>self</i>, <i>uid</i>, <i>zProperty</i>, <i>value</i>)</p> <hr/> <p>Sets the zProperty value</p>
<p>getZenProperties(<i>self</i>, <i>uid</i>, <i>start=0</i>, <i>params="{}</i>", <i>limit=None</i>, <i>sort=None</i>, <i>page=None</i>, <i>dir='ASC'</i>)</p> <hr/> <p>Returns the definition and values of all the zen properties for this context</p> <p>Parameters</p> <p>uid: unique identifier of an object (<i>type=string</i>)</p>
<p>deleteZenProperty(<i>self</i>, <i>uid</i>, <i>zProperty</i>)</p> <hr/> <p>Removes the local instance of the each property in properties. Note that the property will only be deleted if a hasProperty is true</p> <p>Parameters</p> <p>uid: unique identifier of an object (<i>type=String</i>)</p> <p>properties: zenproperty identifier (<i>type=String</i>)</p>
<p>getEvents(<i>self</i>, <i>uid</i>)</p> <hr/> <p>Get events for a device.</p> <p>Parameters</p> <p>uid: Device to get events for (<i>type=[string]</i>)</p> <p>Return Value</p> <p>Properties:</p> <ul style="list-style-type: none"> data: ([dictionary]) List of events for a device (<i>type=DirectResponse</i>)

```
loadRanges(self, ranges, hashcheck, uid=None, params=None, sort='name', dir='ASC')
```

Get a range of device uids.

Parameters

- ranges:** List of two integers that are the min/max values of a range of uids
(*type=[integer]*)
- hashcheck:** Hashcheck for the devices (from getDevices())
(*type=string*)
- uid:** (optional) Organizer to use to get uids (default: None)
(*type=string*)
- params:** (optional) Key-value pair of filters for this search. Can be one of the following: name, ipAddress, deviceClass, or productionState (default: None)
(*type=dictionary*)
- sort:** (optional) Key on which to sort the return result (default: 'name')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC')
(*type=string*)

Return Value

A list of device uids
(*type=[string]*)

```
loadComponentRanges(self, ranges, hashcheck, uid=None, types=(), meta_type=(),  
start=0, limit=None, sort='name', dir='ASC', name=None)
```

Get a range of component uids.

Parameters

- ranges:** List of two integers that are the min/max values of a range of uids
(*type=[integer]*)
- hashcheck:** not used
(*type=string*)
- uid:** (optional) Device to use to get uids (default: None)
(*type=string*)
- types:** (optional) The types of components to retrieve (default: None)
(*type=[string]*)
- meta_type:** (optional) The meta type of the components to retrieve (default: None)
(*type=string*)
- start:** (optional) Offset to return the results from; used in pagination
(default: 0)
(*type=integer*)
- limit:** (optional) Number of items to return; used in pagination (default: None)
(*type=integer*)
- sort:** (optional) Key on which to sort the return result (default: 'name')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC')
(*type=string*)
- name:** (optional) Component name to search for when loading ranges
(default: None)
(*type=string*)

Return Value

A list of component uids
(*type=[string]*)

getUserCommands (<i>self</i> , <i>uid</i>)
Get a list of user commands for a device uid.
Parameters <i>uid</i> : Device to use to get user commands (<i>type=string</i>)
Return Value List of objects representing user commands (<i>type=[dictionary]</i>)

getProductionStates (<i>self</i> , **kwargs)
Get a list of available production states.
Return Value List of name/value pairs of available production states (<i>type=[dictionary]</i>)

getPriorities (<i>self</i> , **kwargs)
Get a list of available device priorities.
Return Value List of name/value pairs of available device priorities (<i>type=[dictionary]</i>)

getCollectors (<i>self</i>)
Get a list of available collectors.
Return Value List of collectors (<i>type=[string]</i>)

getDeviceClasses (<i>self</i> , **data)
Get a list of all device classes.
Return Value Properties: <ul style="list-style-type: none">• <i>deviceClasses</i>: ([dictionary]) List of device classes• <i>totalCount</i>: (integer) Total number of device classes
(<i>type=DirectResponse</i>)

getSystems(*self*, ***data*)

Get a list of all systems.

Return Value**Properties:**

- systems: ([dictionary]) List of systems
- totalCount: (integer) Total number of systems

*(type=DirectResponse)***getGroups**(*self*, ***data*)

Get a list of all groups.

Return Value**Properties:**

- systems: ([dictionary]) List of groups
- totalCount: (integer) Total number of groups

*(type=DirectResponse)***getLocations**(*self*, ***data*)

Get a list of all locations.

Return Value**Properties:**

- systems: ([dictionary]) List of locations
- totalCount: (integer) Total number of locations

*(type=DirectResponse)***getManufacturerNames**(*self*, ***data*)

Get a list of all manufacturer names.

Return Value**Properties:**

- manufacturers: ([dictionary]) List of manufacturer names
- totalCount: (integer) Total number of manufacturer names

(type=DirectResponse)

getHardwareProductNames(*self*, *manufacturer=''*, ***data*)

Get a list of all hardware product names from a manufacturer.

Parameters

manufacturer: Manufacturer name
(*type=string*)

Return Value

Properties:

- **productNames:** ([dictionary]) List of hardware product names
- **totalCount:** (integer) Total number of hardware product names

(*type=DirectResponse*)

getOSProductNames(*self*, *manufacturer=''*, ***data*)

Get a list of all OS product names from a manufacturer.

Parameters

manufacturer: Manufacturer name
(*type=string*)

Return Value

Properties:

- **productNames:** ([dictionary]) List of OS product names
- **totalCount:** (integer) Total number of OS product names

(*type=DirectResponse*)

```

addDevice(self, deviceName, deviceClass, title=None, snmpCommunity="",
snmpPort=161, manageIp="", model=False, collector='localhost', rackSlot=0,
locationPath="", systemPaths=[], groupPaths=[], productionState=1000, comments="",
hwManufacturer="", hwProductName="", osManufacturer="", osProductName="",
priority=3, tag="", serialNumber="", zProperties={})

```

Add a device.

Parameters

deviceName:	Name or IP of the new device (<i>type=string</i>)
deviceClass:	The device class to add new device to (<i>type=string</i>)
title:	(optional) The title of the new device (default: ") (<i>type=string</i>)
snmpCommunity:	(optional) A specific community string to use for this device. (default: ") (<i>type=string</i>)
snmpPort:	(optional) SNMP port on new device (default: 161) (<i>type=integer</i>)
manageIp:	(optional) Management IP address on new device (default: empty/derive from DNS) (<i>type=string</i>)
locationPath:	(optional) Organizer path of the location for this device (<i>type=string</i>)
systemPaths:	(optional) List of organizer paths for the device (<i>type=List (strings)</i>)
groupPaths:	(optional) List of organizer paths for the device (<i>type=List (strings)</i>)
model:	(optional) True to model device at add time (default: False) (<i>type=boolean</i>)
collector:	(optional) Collector to use for new device (default: localhost) (<i>type=string</i>)
rackSlot:	(optional) Rack slot description (default: ") (<i>type=string</i>)
productionState:	(optional) Production state of the new device (default: 1000) (<i>type=integer</i>)
comments:	(optional) Comments on this device (default: ") (<i>type=string</i>)
hwManufacturer:	(optional) Hardware manufacturer name (default: ") (<i>type=string</i>)
hwProductName:	(optional) Hardware product name (default: ") (<i>type=string</i>)
osManufacturer:	(optional) OS manufacturer name (default: ") (<i>type=string</i>)
osProductName:	(optional) OS product name (default: ") (<i>type=string</i>)

remodel(*self*, *deviceUid*)

Submit a job to have a device remodeled.

Parameters

deviceUid: Device uid to have local template
(*type=string*)

Return Value**Properties:**

- **jobId:** (string) ID of the add device job

(*type=DirectResponse*)

addLocalTemplate(*self*, *deviceUid*, *templateId*)

Adds a local template on a device.

Parameters

deviceUid: Device uid to have local template
(*type=string*)

templateId: Name of the new template
(*type=string*)

Return Value

Success message

(*type=DirectResponse*)

removeLocalTemplate(*self*, *deviceUid*, *templateUid*)

Removes a locally defined template on a device.

Parameters

deviceUid: Device uid that has local template
(*type=string*)

templateUid: Name of the template to remove
(*type=string*)

Return Value

Success message

(*type=DirectResponse*)

getLocalTemplates (<i>self</i> , <i>query</i> , <i>uid</i>)
Get a list of locally defined templates on a device.
Parameters
<i>query</i> : not used (<i>type=string</i>)
<i>uid</i> : Device uid to query for templates (<i>type=string</i>)
Return Value
Properties:
• <i>data</i> : ([dictionary]) List of objects representing local templates (<i>type=DirectResponse</i>)

getTemplates (<i>self</i> , <i>id</i>)
Get a list of available templates for a device.
Parameters
<i>id</i> : Device uid to query for templates (<i>type=string</i>)
Return Value
Properties:
• <i>data</i> : ([dictionary]) List of objects representing templates (<i>type=DirectResponse</i>)

getUnboundTemplates (<i>self</i> , <i>uid</i>)
Get a list of unbound templates for a device.
Parameters
<i>uid</i> : Device uid to query for templates (<i>type=string</i>)
Return Value
Properties:
• <i>data</i> : ([dictionary]) List of objects representing templates (<i>type=DirectResponse</i>)

getBoundTemplates(*self, uid*)

Get a list of bound templates for a device.

Parameters

uid: Device uid to query for templates
(*type=string*)

Return Value**Properties:**

- **data:** ([dictionary]) List of objects representing templates
(*type=DirectResponse*)

setBoundTemplates(*self, uid, templateIds*)

Set a list of templates as bound to a device.

Parameters

uid: Device uid to bind templates to
(*type=string*)

templateIds: List of template uids to bind to device
(*type=[string]*)

Return Value

Success message
(*type=DirectResponse*)

resetBoundTemplates(*self, uid*)

Remove all bound templates from a device.

Parameters

uid: Device uid to remove bound templates from
(*type=string*)

Return Value

Success message
(*type=DirectResponse*)

bindOrUnbindTemplate(*self, uid, templateUid*)

Bind an unbound template or unbind a bound template from a device.

Parameters

uid: Device uid to bind/unbind template
(*type=string*)

templateUid: Template uid to bind/unbind
(*type=string*)

Return Value

Success message
(*type=DirectResponse*)

getOverridableTemplates (<i>self, query, uid</i>)
Get a list of available templates on a device that can be overridden.
Parameters
query: not used (<i>type=string</i>)
uid: Device to query for overridable templates (<i>type=string</i>)
Return Value
Properties:
• data: ([dictionary]) List of objects representing templates (<i>type=DirectResponse</i>)

clearGeocodeCache (<i>self</i>)
Clear the Google Maps geocode cache.
Return Value
Success message (<i>type=DirectResponse</i>)

getZenProperty (<i>self, uid, zProperty</i>)
Returns information about a zproperty for a given context, including its value
Return Value
Properties:
• path: (string) where the property is defined
• type: (string) type of zproperty it is
• options: (Array) available options for the zproperty
• value (Array) value of the zproperty
• valueAsString (string)
(<i>type=Dictionary</i>)

getModelerPluginDocStrings (<i>self, uid</i>)
Given a uid returns the documentation for all the modeler plugins.

addIpRouteEntry (<i>self, uid, dest='', routemask='', nexthopid='', interface='', routeproto='', routetype='', userCreated=True</i>)
Adds an Ip Route Entry to this device

addIpInterface (<i>self, uid, newId, userCreated=True</i>)
Adds an Ip Interface

addOSProcess (<i>self</i> , <i>uid</i> , <i>newClassName</i> , <i>userCreated=True</i>)
--

Adds an os processes

addFileSystem (<i>self</i> , <i>uid</i> , <i>newId</i> , <i>userCreated=True</i>)
--

Adds an Ip Interface

addIpService (<i>self</i> , <i>uid</i> , <i>newClassName</i> , <i>protocol</i> , <i>userCreated=True</i>)
--

Adds an Ip Service

addWinService (<i>self</i> , <i>uid</i> , <i>newClassName</i> , <i>userCreated=True</i>)

Adds an Ip Service

Inherited from Products.Zuul.routers.TreeRouter(Section 1.3)

addNode(), asyncGetTree(), deleteNode(), moveOrganizer()

3 Module `Products.Zuul.routers.jobs`

Operations for jobs.

Available at: `/zport/dmd/jobs_router`

3.1 Variables

Name	Description
<code>log</code>	Value: <code>logging.getLogger('zen.JobsRouter')</code>
<code>JOBKEYS</code>	Value: <code>['uuid', 'type', 'description', 'scheduled', 'started', '...']</code>

3.2 Class `JobsRouter`

`Products.ZenUtils.Ext.DirectRouter`  `Products.Zuul.routers.jobs.JobsRouter`

A JSON/Ext.Direct interface to operations on jobs

3.2.1 Methods

```
__init__(self, context, request)
```

```
getJobs(self, start, limit, page, sort, dir, uid=None)
```

```
abort(self, jobids)
```

```
deleteJobs(self, jobids)
```

```
detail(self, jobid)
```

```
userjobs(self)
```

4 Module `Products.Zuul.routers.messaging`

Operations for Messaging.

Available at: `/zport/dmd/messaging_router`

4.1 Class `MessagingRouter`

`Products.ZenUtils.Ext.DirectRouter`  `Products.Zuul.routers.messaging.MessagingRouter`

A JSON/ExtDirect interface to operations on messages

4.1.1 Methods

setBrowserState(*self*, *state*)

Save the browser state for the current user.

Parameters

state: The browser state as a JSON-encoded string

(*type=string*)

clearBrowserState(*self*, *user=None*)

Removes all the stored state associated with the current user

getUserMessages(*self*)

Get the queued messages for the logged in user.

Return Value

Properties:

- **messages**: ([string]) A list of queued messages.

(*type=dictionary*)

5 Module Products.Zuul.routers.mibs

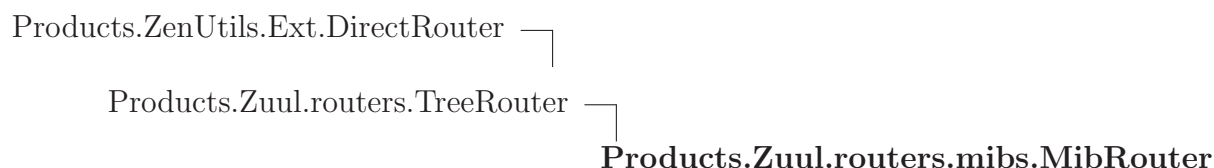
Operations for MIBs.

Available at: /zport/dmd/mib_router

5.1 Variables

Name	Description
log	Value: logging.getLogger('zen.MibRouter')

5.2 Class MibRouter



A JSON/ExtDirect interface to operations on MIBs

5.2.1 Methods

<code>__init__(self, context, request)</code>
<p>getTree(<i>self</i>, <i>id</i>='/zport/dmd/Mibs')</p> <p>Returns the tree structure of an organizer hierarchy. Default tree root is MIBs.</p> <p>Parameters</p> <p>id: (optional) Id of the root node of the tree to be returned (default: '/zport/dmd/Mibs') (<i>type=string</i>)</p> <p>Return Value</p> <p>Object representing the tree (<i>type=[dictionary]</i>)</p>

getOrganizerTree (<i>self</i> , <i>id</i>)
Returns the tree structure of an organizer hierarchy, only including organizers.
Parameters
<i>id</i> : Id of the root node of the tree to be returned (<i>type=string</i>)
Return Value
Object representing the organizer tree (<i>type=[dictionary]</i>)

addNode (<i>self</i> , <i>contextUid</i> ='', <i>id</i> ='', <i>type</i> ='')
Add an organizer or new blank MIB.
Parameters
<i>contextUid</i> : Context to attach new node (<i>type=string</i>)
<i>id</i> : Id of the new organizer or blank MIB (<i>type=string</i>)
<i>type</i> : Type of new node. Can be 'organizer' or 'MIB' (<i>type=string</i>)
Return Value
Properties:
• <i>tree</i> : ([dictionary]) Object representing the new tree (<i>type=DirectResponse</i>)
Overrides: Products.Zuul.routers.TreeRouter.addNode

addMIB (<i>self</i> , <i>package</i> , <i>organizer</i> =' / ')
Add a new MIB by URL or local file.
Parameters
<i>package</i> : URL or local file path to MIB file (<i>type=string</i>)
<i>organizer</i> : ID of the organizer to add MIB to (<i>type=string</i>)
Return Value
Properties:
• <i>jobId</i> : (string) ID of the add MIB job (<i>type=DirectResponse</i>)

deleteNode (<i>self</i> , <i>uid</i>)
Remove an organizer or MIB.
Parameters
<i>uid</i> : UID of organizer or MIB to remove (<i>type=string</i>)
Return Value
Properties:
• <i>tree</i> : ([dictionary]) Object representing the new tree (<i>type=DirectResponse</i>)
Overrides: Products.Zuul.routers.TreeRouter.deleteNode

moveNode (<i>self</i> , <i>uids</i> , <i>target</i>)
Move an organizer or MIB from one organizer to another.
Parameters
<i>uids</i> : UIDs of organizers and MIBs to move (<i>type=[string]</i>)
<i>target</i> : UID of the organizer to move to (<i>type=string</i>)
Return Value
Properties:
• <i>data</i> : (dictionary) Object representing the new parent organizer (<i>type=DirectResponse</i>)

getInfo(*self*, *uid*, *useFieldSets=True*)

Get the properties of a MIB

Parameters

uid: Unique identifier of a MIB
(*type=string*)

useFieldSets: True to return a fieldset version of the info form
(default: True)
(*type=boolean*)

Return Value

Properties

- **data:** (dictionary) Object representing a MIB's properties
- **form:** (dictionary) Object representing an edit form for a MIB's properties

(*type=DirectResponse*)

setInfo(*self*, ****data**)

Set attributes on a MIB. This method accepts any keyword argument for the property that you wish to set. The only required property is "uid".

Parameters

uid: Unique identifier of a MIB
(*type=string*)

Return Value

Properties

- **data:** (dictionary) Object representing a MIB's new properties

(*type=DirectResponse*)

addOidMapping(*self*, *uid*, *id*, *oid*, *nodetype='node'*)

addTrap(*self*, *uid*, *id*, *oid*, *nodetype='notification'*)

deleteOidMapping(*self*, *uid*)

deleteTrap(*self*, *uid*)

getOidMappings(*self*, *uid*, *dir='ASC'*, *sort='name'*, *start=0*, *page=None*, *limit=256*)

```
getTraps(self, uid, dir='ASC', sort='name', start=0, page=None,  
limit=256)
```

```
getMibNodeTree(self, id=None)
```

A MIB node is a regular OID (ie you can hit it with snmpwalk)

```
getMibTrapTree(self, id=None)
```

A MIB trap node is an OID received from a trap

Inherited from Products.Zuul.routers.TreeRouter(Section 1.3)

asyncGetTree(), moveOrganizer()

6 Module Products.Zuul.routers.nav

Operations for Navigation

Available at: /zport/dmd/detailnav_router

6.1 Class DetailNavRouter

Products.ZenUtils.Ext.DirectRouter —
Products.Zuul.routers.nav.DetailNavRouter

Router to Details navigation for given uid

6.1.1 Methods

getDetailNavConfigs(*self*, *uid*=None, *menuIds*=None)

return a list of Detail navigation configurations. Can be used to create navigation links. Format is: { id: <id of the configuration>, 'viewName': <view to display>, 'xtype': <Ext type for the panel>, 'text': <display name of the config info> }

getContextMenus(*self*, *uid*=None, *menuIds*=None)

getSecurityPermissions(*self*, *uid*)

returns a dictionary of all the permissions a user has on the context

7 Module Products.Zuul.routers.network

Operations for Networks.

Available at: /zport/dmd/network_router

7.1 Variables

Name	Description
log	Value: logging.getLogger('zen.NetworkRouter')

7.2 Class NetworkRouter



A JSON/ExtDirect interface to operations on networks

7.2.1 Methods

<code>__init__(self, context, request)</code>
discoverDevices (<i>self</i> , <i>uid</i>) Discover devices on a network. Parameters uid: Unique identifier of the network to discover (<i>type=string</i>) Return Value Properties: <ul style="list-style-type: none"> jobId: (integer) The id of the discovery job (<i>type=DirectResponse</i>)

addNode(*self*, *newSubnet*, *contextUid*)

Add a new subnet.

Parameters

newSubnet: New subnet to add
(*type=string*)

contextUid: Unique identifier of the network parent of the new subnet
(*type=string*)

Return Value**Properties:**

- **newNode:** (dictionary) An object representing the new subnet node
(*type=DirectResponse*)

Overrides: Products.Zuul.routers.TreeRouter.addNode

deleteNode(*self*, *uid*)

Delete a subnet.

Parameters

uid: Unique identifier of the subnet to delete
(*type=string*)

Return Value**Properties:**

- **tree:** (dictionary) An object representing the new network tree
(*type=DirectResponse*)

Overrides: Products.Zuul.routers.TreeRouter.deleteNode

getTree(*self*, *id*='/zport/dmd/Networks')

Returns the tree structure of an organizer hierarchy where the root node is the organizer identified by the id parameter.

Parameters

id: Id of the root node of the tree to be returned. Defaults to the Networks tree root.
(*type=string*)

Return Value

Object representing the tree
(*type=[dictionary]*)

getInfo(*self*, *uid*, *keys=None*)

Returns a dictionary of the properties of an object

Parameters

uid: Unique identifier of an object
(*type=string*)

keys: (optional) List of keys to include in the returned dictionary. If None then all keys will be returned
(*type=list*)

Return Value**Properties**

- **data**: (dictionary) Object properties

(*type=DirectResponse*)

setInfo(*self*, ****data**)

Main method for setting attributes on a network or network organizer. This method accepts any keyword argument for the property that you wish to set. The only required property is "uid".

Parameters

uid: Unique identifier of an object
(*type=string*)

Return Value

DirectResponse

```
getIpAddresses(self, uid, start=0, params=None, limit=50,
sort='ipAddressAsInt', page=None, dir='ASC')
```

Given a subnet, get a list of IP addresses and their relations.

Parameters

- uid:** Unique identifier of a subnet
(*type=string*)
- start:** Offset to return the results from; used in pagination
(*type=integer*)
- params:** Not used
(*type=string*)
- limit:** Number of items to return; used in pagination
(*type=integer*)
- sort:** (optional) Key on which to sort the return results; defaults to 'name'
(*type=string*)
- order:** Sort order; can be either 'ASC' or 'DESC'
(*type=string*)

Return Value

DirectResponse

```
removeIpAddresses(self, uids=None)
```

Removes every ip address specified by uids that are not attached to any device

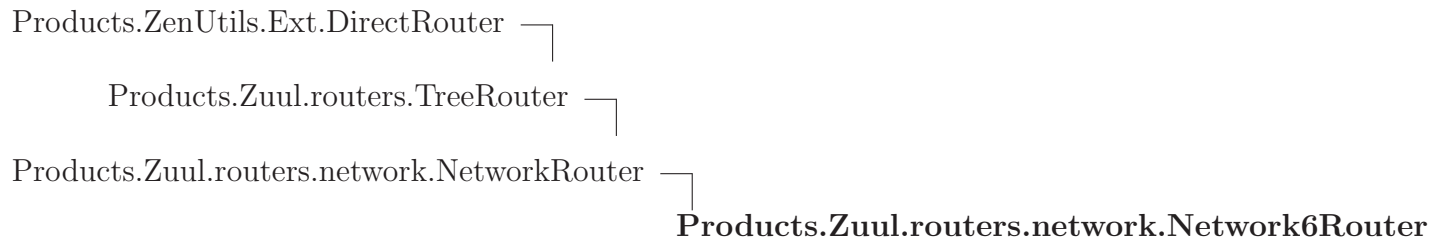
Parameters

- uids:** unique identifiers of the ip addresses to delete
(*type=Array of Strings*)

Inherited from Products.Zuul.routers.TreeRouter(Section 1.3)

asyncGetTree(), moveOrganizer()

7.3 Class Network6Router



A JSON/ExtDirect interface to operations on IPv6 networks

7.3.1 Methods

__init__ (<i>self</i> , <i>context</i> , <i>request</i>)

Overrides: Products.Zuul.routers.network.NetworkRouter.__init__

Inherited from Products.Zuul.routers.network.NetworkRouter(Section 7.2)

addNode(), deleteNode(), discoverDevices(), getInfo(), getIpAddresses(), getTree(),
removeIpAddresses(), setInfo()

Inherited from Products.Zuul.routers.TreeRouter(Section 1.3)

asyncGetTree(), moveOrganizer()

8 Module `Products.Zuul.routers.process`

Operations for Processes.

Available at: `/zport/dmd/process_router`

8.1 Class `ProcessRouter`



A JSON/ExtDirect interface to operations on processes

8.1.1 Methods

getTree(*self*, *id*)

Returns the tree structure of an organizer hierarchy where the root node is the organizer identified by the *id* parameter.

Parameters

id: Id of the root node of the tree to be returned

(*type=string*)

Return Value

Object representing the tree

(*type=[dictionary]*)

moveProcess (<i>self</i> , <i>uid</i> , <i>targetUid</i>)
Move a process or organizer from one organizer to another.
Parameters
uid : UID of the process or organizer to move (<i>type=string</i>)
targetUid : UID of the organizer to move to (<i>type=string</i>)
Return Value
Properties:
• uid : (dictionary) The new uid for moved process or organizer (<i>type=DirectResponse</i>)

getInfo (<i>self</i> , <i>uid</i> , <i>keys=None</i>)
Get the properties of a process.
Parameters
uid : Unique identifier of a process (<i>type=string</i>)
keys : (optional) List of keys to include in the returned dictionary. If None then all keys will be returned (default: None) (<i>type=list</i>)
Return Value
Properties
• data : (dictionary) Object representing a process's properties (<i>type=DirectResponse</i>)

setInfo (<i>self</i> , **data)
Set attributes on a process. This method accepts any keyword argument for the property that you wish to set. The only required property is "uid".
Parameters
uid : Unique identifier of a process (<i>type=string</i>)
Return Value
Properties
• data : (dictionary) Object representing a process's new properties (<i>type=DirectResponse</i>)

getInstances(*self*, *uid*, *start*=0, *params*=None, *limit*=50, *sort*='name', *page*=None, *dir*='ASC')

Get a list of instances for a process UID.

Parameters

- uid:** Process UID to get instances of
(*type=string*)
- start:** (optional) Offset to return the results from; used in pagination (default: 0)
(*type=integer*)
- params:** (optional) Key-value pair of filters for this search.
(*type=dictionary*)
- limit:** (optional) Number of items to return; used in pagination (default: 50)
(*type=integer*)
- sort:** (optional) Key on which to sort the return results (default: 'name')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC')
(*type=string*)

Return Value

Properties:

- **data:** ([dictionary]) List of objects representing process instances
- **total:** (integer) Total number of instances

(*type=DirectResponse*)

getSequence(*self*, **args*, ***kwargs*)

Get the current processes sequence.

Return Value

Properties:

- **data:** ([dictionary]) List of objects representing processes in sequence order

(*type=DirectResponse*)

setSequence(*self*, *uids*)

Set the current processes sequence.

Parameters

uids: The set of process uid's in the desired sequence
(*type*=*[string]*)

Return Value

Success message
(*type*=*DirectResponse*)

```
query(self, limit=None, start=None, sort=None, dir=None, params=None,
page=None, history=False, uid=None, criteria=())
```

Retrieve a list of processes based on a set of parameters.

Parameters

- limit:** (optional) Number of items to return; used in pagination
(default: None)
(*type=integer*)
- start:** (optional) Offset to return the results from; used in
pagination (default: None)
(*type=integer*)
- sort:** (optional) Key on which to sort the return results
(default: None)
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC'
(default: None)
(*type=string*)
- params:** (optional) Key-value pair of filters for this search.
(*type=dictionary*)
- history:** not used
(*type=boolean*)
- uid:** Service class UID to query
(*type=string*)
- criteria:** not used
(*type=list*)

Return Value

Properties:

- **processes:** ([dictionary]) List of objects representing processes
- **totalCount:** (integer) Total number of processes
- **hash:** (string) Hashcheck of the current processes state
- **disabled:** (boolean) True if current user cannot manage processes

(*type=DirectResponse*)

Inherited from Products.Zuul.routers.TreeRouter(Section 1.3)

addNode(), asyncGetTree(), deleteNode(), moveOrganizer()

9 Module Products.Zuul.routers.report

Operations for Reports.

Available at: /zport/dmd/report_router

9.1 Variables

Name	Description
log	Value: logging.getLogger('zen.ReportRouter')
reportTypes	Value: ['customDeviceReport', 'graphReport', 'multiGraphReport',]
menuText	Value: [_t('Custom Device Report'), _t('Graph Report'), _t('Mult...)]
essentialReportOrganizers	Value: ['/zport/dmd/Reports/Custom Device Reports', '/zport/dmd/...']

9.2 Class ReportRouter



A JSON/ExtDirect interface to operations on reports

9.2.1 Methods

<code>__init__(self, context, request)</code>

getTree(*self*, *id*)

Returns the tree structure of an organizer hierarchy where the root node is the organizer identified by the id parameter.

Parameters

id: Id of the root node of the tree to be returned
(*type=string*)

Return Value

Object representing the tree
(*type=[dictionary]*)

getReportTypes(*self*)

Get the available report types.

Return Value**Properties:**

- **menuText**: ([string]) Human readable list of report types
- **reportTypes**: ([string]) A list of the available report types

(*type=DirectResponse*)

asyncGetTree(*self*, *id=None*)

Server side method for asynchronous tree calls. Retrieves the immediate children of the node specified by "id"

NOTE: our convention on the UI side is if we are asking for the root node then return the root and its children otherwise just return the children

Parameters

id: The uid of the node we are getting the children for

Return Value

Object representing the immediate children
(*type=[dictionary]*)

Overrides: Products.Zuul.routers.TreeRouter.asyncGetTree extit(inherited documentation)

addNode(*self*, *nodeType*, *contextUid*, *id*)

Add a new report or report organizer.

Parameters

- nodeType:** Type of new node. Can either be 'organizer' or one of the report types returned from getReportTypes()
(*type=string*)
- contextUid:** The organizer where the new node should be added
(*type=string*)
- id:** The new node's ID
(*type=string*)

Return Value**Properties:**

- tree: (dictionary) Object representing the new Reports tree
- newNode: (dictionary) Object representing the added node

(*type=DirectResponse*)

Overrides: Products.Zuul.routers.TreeRouter.addNode

deleteNode(*self*, *uid*)

Remove a report or report organizer.

Parameters

- uid:** The UID of the node to delete
(*type=string*)

Return Value**Properties:**

- tree: (dictionary) Object representing the new Reports tree

(*type=[dictionary]*)

Overrides: Products.Zuul.routers.TreeRouter.deleteNode

moveNode(*self*, *uid*, *target*)

Move a report or report organizer from one organizer to another.

Parameters

uid: The UID of node to move
(*type=string*)

target: The UID of the target Report organizer
(*type=string*)

Return Value

Properties:

- **tree:** (dictionary) Object representing the new Reports tree
- **newNode:** (dictionary) Object representing the moved node

(*type=[dictionary]*)

Inherited from Products.Zuul.routers.TreeRouter(Section 1.3)

moveOrganizer()

10 Module Products.Zuul.routers.service

Operations for Services.

Available at: /zport/dmd/service_router

10.1 Class ServiceRouter



A JSON/ExtDirect interface to operations on services

10.1.1 Methods

```
__init__(self, context, request)
```

```
getClassNames(self, uid=None, query=None)
```

```
addClass(self, contextUid, id, posQuery=None)
```

Add a new service class.

Parameters

- contextUid:** Unique ID of the service organizer to add new class to
(*type=string*)
- id:** ID of the new service
(*type=string*)
- posQuery:** Object defining a query where the returned position will lie
(*type=dictionary*)

Return Value

Properties:

- **newIndex:** (integer) Index of the newly added class in the query defined by posQuery

(*type=DirectResponse*)

```
query(self, limit=None, start=None, sort=None, dir=None, params=None,
page=None, history=False, uid=None, criteria=())
```

Retrieve a list of services based on a set of parameters.

Parameters

- limit:** (optional) Number of items to return; used in pagination
(default: None)
(*type=integer*)
- start:** (optional) Offset to return the results from; used in
pagination (default: None)
(*type=integer*)
- sort:** (optional) Key on which to sort the return results
(default: None)
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC'
(default: None)
(*type=string*)
- params:** (optional) Key-value pair of filters for this search.
(*type=dictionary*)
- history:** not used
(*type=boolean*)
- uid:** Service class UID to query
(*type=string*)
- criteria:** not used
(*type=list*)

Return Value

Properties:

- **services:** ([dictionary]) List of objects representing services
- **totalCount:** (integer) Total number of services
- **hash:** (string) Hashcheck of the current services state
- **disabled:** (boolean) True if current user cannot manage services

(*type=DirectResponse*)

getTree (<i>self</i> , <i>id</i>)
Returns the tree structure of an organizer hierarchy.
Parameters
<i>id</i> : Id of the root node of the tree to be returned (<i>type=string</i>)
Return Value
Object representing the tree (<i>type=[dictionary]</i>)

getOrganizerTree (<i>self</i> , <i>id</i>)
Returns the tree structure of an organizer hierarchy, only including organizers.
Parameters
<i>id</i> : Id of the root node of the tree to be returned (<i>type=string</i>)
Return Value
Object representing the organizer tree (<i>type=[dictionary]</i>)

getInfo (<i>self</i> , <i>uid</i> , <i>keys=None</i>)
Get the properties of a service.
Parameters
<i>uid</i> : Unique identifier of a service (<i>type=string</i>)
<i>keys</i> : (optional) List of keys to include in the returned dictionary. If None then all keys will be returned (default: None) (<i>type=list</i>)
Return Value
Properties
<ul style="list-style-type: none">• <i>data</i>: (dictionary) Object representing a service's properties• <i>disabled</i>: (boolean) True if current user cannot manage services
(<i>type=DirectResponse</i>)

setInfo(*self*, ****data**)

Set attributes on a service. This method accepts any keyword argument for the property that you wish to set. The only required property is "uid".

Parameters

uid: Unique identifier of a service
(*type=string*)

Return Value

Success message
(*type=DirectResponse*)

getInstances(*self*, *uid*, *start=0*, *params=None*, *limit=50*, *sort='name'*, *page=None*, *dir='ASC'*)

Get a list of instances for a service UID.

Parameters

uid: Service UID to get instances of
(*type=string*)

start: (optional) Offset to return the results from; used in pagination (default: 0)
(*type=integer*)

params: (optional) Key-value pair of filters for this search.
(*type=dictionary*)

limit: (optional) Number of items to return; used in pagination (default: 50)
(*type=integer*)

sort: (optional) Key on which to sort the return results (default: 'name')
(*type=string*)

dir: (optional) Sort order; can be either 'ASC' or 'DESC' (default: 'ASC')
(*type=string*)

Return Value**Properties:**

- **data:** ([dictionary]) List of objects representing service instances
- **totalCount:** (integer) Total number of instances

(*type=DirectResponse*)

moveServices (<i>self</i> , <i>sourceUids</i> , <i>targetUid</i>)
Move service(s) from one organizer to another.
Parameters
<i>sourceUids</i> : UID(s) of the service(s) to move (<i>type</i> =[string])
<i>targetUid</i> : UID of the organizer to move to (<i>type</i> =string)
Return Value
Success message (<i>type</i> =DirectResponse)

getUnmonitoredStartModes (<i>self</i> , <i>uid</i>)
Get a list of unmonitored start modes for a Windows service.
Parameters
<i>uid</i> : Unique ID of a Windows service. (<i>type</i> =string)
Return Value
Properties:
• <i>data</i> : ([string]) List of unmonitored start modes for a Windows service
(<i>type</i> =DirectResponse)

getMonitoredStartModes (<i>self</i> , <i>uid</i> , <i>query</i> ='')
Get a list of monitored start modes for a Windows service.
Parameters
<i>uid</i> : Unique ID of a Windows service. (<i>type</i> =string)
Return Value
Properties:
• <i>data</i> : ([string]) List of monitored start modes for a Windows service
(<i>type</i> =DirectResponse)

Inherited from Products.Zuul.routers.TreeRouter(Section 1.3)

addNode(), asyncGetTree(), deleteNode(), moveOrganizer()

11 Module `Products.Zuul.routers.settings`

Operations for Settings.

Available at: `/zport/dmd/settings_router`

11.1 Class `SettingsRouter`

`Products.ZenUtils.Ext.DirectRouter`  `Products.Zuul.routers.settings.SettingsRouter`

A JSON/ExtDirect interface to operations on settings

11.1.1 Methods

<code>getUserInterfaceSettings(self)</code>
--

Retrieves the collection of User interface settings

<code>setUserInterfaceSettings(self, **kwargs)</code>
--

Accepts key value pair of user interface settings.
--

12 Module `Products.Zuul.routers.template`

Operations for Templates.

Available at: `/zport/dmd/template_router`

12.1 Class `TemplateRouter`



A JSON/ExtDirect interface to operations on templates

12.1.1 Methods

`getTemplates(self, id)`

Get all templates.

Parameters

`id`: not used

(type=string)

Return Value

List of objects representing the templates in tree hierarchy

(type=[dictionary])

`getDeviceClassTemplates(self, id)`

Get all templates by device class. This will return a tree where device classes are nodes, and templates are leaves.

Parameters

`id`: not used

(type=string)

Return Value

List of objects representing the templates in tree hierarchy

(type=[dictionary])

getAddTemplateTargets(*self*, *query=None*)

Get a list of available device classes where new templates can be added.

Parameters

query: not used
(*type=string*)

Return Value**Properties:**

- **data:** ([dictionary]) List of objects containing an available device class UID and a human-readable label for that class

(*type=DirectResponse*)

addTemplate(*self*, *id*, *targetUid*)

Add a template to a device class.

Parameters

id: Unique ID of the template to add
(*type=string*)

targetUid: Unique ID of the device class to add template to
(*type=string*)

Return Value**Properties:**

- **nodeConfig:** (dictionary) Object representing the added template

(*type=DirectResponse*)

deleteTemplate(*self*, *uid*)

Delete a template.

Parameters

uid: Unique ID of the template to delete
(*type=string*)

Return Value

Success message

(*type=DirectResponse*)

getObjTemplates(*self*, *uid*)**Parameters**

uid: Identifier for the object we want templates on, must descend from RRDView

(*type=string*)

Return Value

List of templates

(*type=DirectResponse*)

makeLocalRRDTemplate(*self*, *uid*, *templateName*)**Parameters**

uid: Identifier of the obj we wish to make the template local for

(*type=string*)

templateName: identifier of the template

(*type=string*)

removeLocalRRDTemplate(*self*, *uid*, *templateName*)**Parameters**

uid: Identifier of the obj we wish to remove the local template

(*type=string*)

templateName: identifier of the local template

(*type=string*)

getThresholds(*self*, *uid*, *query*='')

Get the thresholds for a template.

Parameters

uid: Unique ID of a template

(*type=string*)

query: not used

(*type=string*)

Return Value

List of objects representing representing thresholds

(*type=[dictionary]*)

getThresholdDetails(*self*, *uid*)

Get a threshold's details.

Parameters

uid: Unique ID of a threshold
(*type=string*)

Return Value**Properties:**

- **record:** (dictionary) Object representing the threshold
- **form:** (dictionary) Object representing an ExtJS form for the threshold

(*type=dictionary*)

getDataPoints(*self*, *uid*, *query=''*)

Get a list of available data points for a template.

Parameters

query: not used
(*type=string*)

uid: Unique ID of a template
(*type=string*)

Return Value**Properties:**

- **data:** ([dictionary]) List of objects representing data points

(*type=DirectResponse*)

addDataPoint(*self*, *dataSourceUid*, *name*)

Add a new data point to a data source.

Parameters

dataSourceUid: Unique ID of the data source to add data point to
(*type=string*)

name: ID of the new data point
(*type=string*)

Return Value

Success message

(*type=DirectResponse*)

addDataSource(*self, templateUid, name, type*)

Add a new data source to a template.

Parameters

- templateUid:** Unique ID of the template to add data source to
(*type=string*)
- name:** ID of the new data source
(*type=string*)
- type:** Type of the new data source. From `getDataSourceTypes()`
(*type=string*)

Return Value

Success message
(*type=DirectResponse*)

getDataSources(*self, uid*)

Get the data sources for a template.

Parameters

- id:** Unique ID of a template
(*type=string*)

Return Value

List of objects representing representing data sources
(*type=[dictionary]*)

getDataSourceDetails(*self, uid*)

Get a data source's details.

Parameters

- uid:** Unique ID of a data source
(*type=string*)

Return Value**Properties:**

- **record:** (dictionary) Object representing the data source
- **form:** (dictionary) Object representing an ExtJS form for the data source

(*type=dictionary*)

getDataPointDetails(*self*, *uid*)

Get a data point's details.

Parameters

uid: Unique ID of a data point
(*type=string*)

Return Value**Properties:**

- **record**: (dictionary) Object representing the data point
- **form**: (dictionary) Object representing an ExtJS form for the data point

(*type=dictionary*)

setInfo(*self*, ****data**)

Set attributes on an object. This method accepts any keyword argument for the property that you wish to set. The only required property is "uid".

Parameters

uid: Unique identifier of an object
(*type=string*)

Return Value**Properties:**

- **data**: (dictionary) The modified object

(*type=DirectResponse*)

addThreshold(*self*, ***data*)

Add a threshold.

Parameters

- uid:** Unique identifier of template to add threshold to
(*type=string*)
- thresholdType:** Type of the new threshold. From
getThresholdTypes()
(*type=string*)
- thresholdId:** ID of the new threshold
(*type=string*)
- dataPoints:** List of data points to select for this threshold
(*type=[string]*)

Return Value

Success message
(*type=DirectResponse*)

removeThreshold(*self*, *uid*)

Remove a threshold.

Parameters

- uid:** Unique identifier of threshold to remove
(*type=string*)

Return Value

Success message
(*type=DirectResponse*)

getThresholdTypes(*self*, *query=None*)

Get a list of available threshold types.

Parameters

- query:** not used
(*type=string*)

Return Value

List of objects representing threshold types
(*type=[dictionary]*)

getDataSourceTypes(*self*, *query*)

Get a list of available data source types.

Parameters

query: not used
(*type=string*)

Return Value

List of objects representing data source types
(*type=[dictionary]*)

getGraphs(*self*, *uid*, *query=None*)

Get the graph definitions for a template.

Parameters

uid: Unique ID of a template
(*type=string*)
query: not used
(*type=string*)

Return Value

List of objects representing representing graphs
(*type=[dictionary]*)

addDataPointToGraph(*self*, *dataPointUid*, *graphUid*,
includeThresholds=False)

Add a data point to a graph.

Parameters

dataPointUid: Unique ID of the data point to add to graph
(*type=string*)
graphUid: Unique ID of the graph to add data point to
(*type=string*)
includeThresholds: (optional) True to include related thresholds
(default: False)
(*type=boolean*)

Return Value

Success message
(*type=DirectResponse*)

getCopyTargets(*self*, *uid*, *query*='')

Get a list of available device classes to copy a template to.

Parameters

uid: Unique ID of the template to copy
(*type=string*)

query: (optional) Filter the returned targets' names based on this parameter (default: '')
(*type=string*)

Return Value**Properties:**

- **data:** ([dictionary]) List of objects containing an available device class UID and a human-readable label for that class

(*type=DirectResponse*)

copyTemplate(*self*, *uid*, *targetUid*)

Copy a template to a device or device class.

Parameters

uid: Unique ID of the template to copy
(*type=string*)

targetUid: Unique ID of the device or device class to bind to template
(*type=string*)

Return Value

Success message

(*type=DirectResponse*)

addGraphDefinition(*self, templateUid, graphDefinitionId*)

Add a new graph definition to a template.

Parameters

templateUid: Unique ID of the template to add graph definition to
(*type=string*)

graphDefinitionId: ID of the new graph definition
(*type=string*)

Return Value

Success message
(*type=DirectResponse*)

deleteDataSource(*self, uid*)

Delete a data source.

Parameters

uid: Unique ID of the data source to delete
(*type=string*)

Return Value

Success message
(*type=DirectResponse*)

deleteDataPoint(*self, uid*)

Delete a data point.

Parameters

uid: Unique ID of the data point to delete
(*type=string*)

Return Value

Success message
(*type=DirectResponse*)

deleteGraphDefinition(*self*, *uid*)

Delete a graph definition.

Parameters

uid: Unique ID of the graph definition to delete
(*type=string*)

Return Value

Success message
(*type=DirectResponse*)

deleteGraphPoint(*self*, *uid*)

Delete a graph point.

Parameters

uid: Unique ID of the graph point to delete
(*type=string*)

Return Value

Success message
(*type=DirectResponse*)

getGraphPoints(*self*, *uid*)

Get a list of graph points for a graph definition.

Parameters

uid: Unique ID of a graph definition
(*type=string*)

Return Value**Properties:**

- **data**: ([dictionary]) List of objects representing graph points

(*type=DirectResponse*)

getInfo(*self*, *uid*)

Get the properties of an object.

Parameters

uid: Unique identifier of an object
(*type=string*)

Return Value**Properties**

- **data:** (dictionary) Object properties
- **form:** (dictionary) Object representing an ExtJS form for the object

(*type=DirectResponse*)

addThresholdToGraph(*self*, *graphUid*, *thresholdUid*)

Add a threshold to a graph definition.

Parameters

graphUid: Unique ID of the graph definition to add threshold to
(*type=string*)

thresholdUid: Unique ID of the threshold to add
(*type=string*)

Return Value

Success message

(*type=DirectResponse*)

addCustomToGraph (<i>self, graphUid, customId, customType</i>)
Add a custom graph point to a graph definition.
Parameters
graphUid : Unique ID of the graph definition to add graph point to (<i>type=string</i>)
customId : ID of the new custom graph point (<i>type=string</i>)
customType : Type of the new graph point. From <code>getGraphInstructionTypes()</code> (<i>type=string</i>)
Return Value
Success message (<i>type=DirectResponse</i>)

getGraphInstructionTypes (<i>self, query=''</i>)
Get a list of available instruction types for graph points.
Parameters
query : not used (<i>type=string</i>)
Return Value
Properties:
• data : ([dictionary]) List of objects representing instruction types (<i>type=DirectResponse</i>)

setGraphPointSequence (<i>self, uids</i>)
Sets the sequence of graph points in a graph definition.
Parameters
uids : List of graph point UID's in desired order (<i>type=[string]</i>)
Return Value
Success message (<i>type=DirectResponse</i>)

<p>getGraphDefinition(<i>self</i>, <i>uid</i>)</p> <hr/> <p>Get a graph definition.</p> <p>Parameters</p> <p> <i>uid</i>: Unique ID of the graph definition to retrieve (<i>type=string</i>)</p> <p>Return Value</p> <p> Properties:</p> <ul style="list-style-type: none"> • <i>data</i>: (dictionary) Object representing a graph definition (<i>type=DirectResponse</i>)
--

<p>setGraphDefinition(<i>self</i>, **data)</p> <hr/> <p>Set attributes on an graph definition. This method accepts any keyword argument for the property that you wish to set. Properties are enumerated via <code>getGraphDefinition()</code>. The only required property is "uid".</p> <p>Parameters</p> <p> <i>uid</i>: Unique identifier of an object (<i>type=string</i>)</p> <p>Return Value</p> <p> Properties:</p> <ul style="list-style-type: none"> • <i>data</i>: (dictionary) The modified object (<i>type=DirectResponse</i>)
--

<p>setGraphDefinitionSequence(<i>self</i>, <i>uids</i>)</p> <hr/> <p>Sets the sequence of graph definitions.</p> <p>Parameters</p> <p> <i>uids</i>: List of graph definition UID's in desired order (<i>type=[string]</i>)</p> <p>Return Value</p> <p> Success message (<i>type=DirectResponse</i>)</p>
--

Inherited from Products.Zuul.routers.TreeRouter(Section 1.3)

`addNode()`, `asyncGetTree()`, `deleteNode()`, `moveOrganizer()`

13 Module `Products.Zuul.routers.triggers`

13.1 Variables

Name	Description
<code>log</code>	Value: <code>logging.getLogger('zen.triggers')</code>

13.2 Class `TriggersRouter`



Router for Triggers UI section under Events.

13.2.1 Methods

```
getTriggers(self, **kwargs)
```

```
getTriggerList(self, **unused)
```

```
addTrigger(self, newId)
```

```
removeTrigger(self, uuid)
```

```
getTrigger(self, uuid)
```

```
updateTrigger(self, **data)
```

```
parseFilter(self, source)
```

```
getNotifications(self)
```

```
addNotification(self, newId, action)
```

```
removeNotification(self, uid)
```

```
getNotificationTypes(self, query='')
```

```
getNotification(self, uid)
```

```
updateNotification(self, **data)
```

```
getRecipientOptions(self, **kwargs)
```

```
getWindows(self, uid, **kwargs)
```

```
addWindow(self, contextUid, newId)
```

```
removeWindow(self, uid)
```

```
getWindow(self, uid)
```

```
updateWindow(self, **data)
```


14 Module Products.Zuul.routers.zenpack

Operations for ZenPacks.

Available at: /zport/dmd/zenpack_router

14.1 Variables

Name	Description
log	Value: logging.getLogger('zen.ZenPackRouter')

14.2 Class ZenPackRouter

Products.ZenUtils.Ext.DirectRouter —
Products.Zuul.routers.zenpack.ZenPackRouter

A JSON/ExtDirect interface to operations on ZenPacks

14.2.1 Methods

getEligiblePacks (<i>self</i> , <i>**data</i>)
Get a list of eligible ZenPacks to add to.
Return Value
Properties:
<ul style="list-style-type: none"> • packs: ([dictionary]) List of objects representing ZenPacks • totalCount: (integer) Total number of eligible ZenPacks
(<i>type=DirectResponse</i>)

addToZenPack(*self*, *topack*, *zenpack*)

Add an object to a ZenPack.

Parameters

topack: Unique ID of the object to add to ZenPack
(*type=string*)

zenpack: Unique ID of the ZenPack to add object to
(*type=string*)

Return Value

Success message

(*type=DirectResponse*)

15 Module `Products.Zuul.routers.zep`

Operations for Events.

Available at: `/zport/dmd/evconsole_router`

15.1 Variables

Name	Description
<code>log</code>	Value: <code>logging.getLogger('zen.%s' % _name_)</code>

15.2 Class `EventsRouter`

`Products.ZenUtils.Ext.DirectRouter`  `Products.Zuul.routers.zep.EventsRouter`

A JSON/ExtDirect interface to operations on events in ZEP

15.2.1 Methods

```
__init__(self, context, request)
```

```
queryArchive(self, page=None, limit=0, start=0, sort='lastTime',
dir='desc', params=None, exclusion_filter=None, keys=None, uid=None,
detailFormat=False)
```

```
query(self, limit=0, start=0, sort='lastTime', dir='desc', params=None,
exclusion_filter=None, keys=None, page=None, archive=False, uid=None,
detailFormat=False)
```

Query for events.

Parameters

- limit:** (optional) Max index of events to retrieve (default: 0)
(*type=integer*)
- start:** (optional) Min index of events to retrieve (default: 0)
(*type=integer*)
- sort:** (optional) Key on which to sort the return results
(default: 'lastTime')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC'
(default: 'DESC')
(*type=string*)
- params:** (optional) Key-value pair of filters for this search.
(default: None)
(*type=dictionary*)
- history:** (optional) True to search the event history table instead of
active events (default: False)
(*type=boolean*)
- uid:** (optional) Context for the query (default: None)
(*type=string*)

Return Value

Properties:

- **events:** ([dictionary]) List of objects representing events
- **totalCount:** (integer) Total count of events returned
- **asof:** (float) Current time

(*type=dictionary*)

```
queryGenerator(self, sort='lastTime', dir='desc', evids=None,  
excludeIds=None, params=None, archive=False, uid=None,  
detailFormat=False)
```

Query for events.

Parameters

- sort:** (optional) Key on which to sort the return results
(default: 'lastTime')
(*type=string*)
- dir:** (optional) Sort order; can be either 'ASC' or 'DESC'
(default: 'DESC')
(*type=string*)
- params:** (optional) Key-value pair of filters for this search.
(default: None)
(*type=dictionary*)
- archive:** (optional) True to search the event archive instead of
active events (default: False)
(*type=boolean*)
- uid:** (optional) Context for the query (default: None)
(*type=string*)

Return Value

Generator returning events.
(*type=generator*)

```
detail(self, evid)
```

Get event details.

Parameters

- evid:** Event ID to get details
(*type=string*)
- history:** Deprecated
(*type=boolean*)

Return Value

Properties:

- **event:** ([dictionary]) List containing a dictionary representing event details

(*type=DirectResponse*)

write_log(*self*, *evid=None*, *message=None*)

Write a message to an event's log.

Parameters

evid: Event ID to log to
(*type=string*)

message: Message to log
(*type=string*)

Return Value

Success message
(*type=DirectResponse*)

postNote(*self*, *uuid*, *note*)

nextEventSummaryUpdate(*self*, *next_request*)

When performing updates from the event console, updates are performed in batches to allow the user to see the progress of event changes and cancel out of updates while they are in progress. This works by specifying a limit to one of the `close`, `acknowledge`, or `reopen` calls in this router. The response will contain an `EventSummaryUpdateResponse`, and if there are additional updates to be performed, it will contain a `next_request` field with all of the parameters used to update the next range of events.

Parameters

next_request: The `next_request` field from the previous updates.
(*type=dictionary*)

clear_device_heartbeats(*self*, *params*, *limit=None*)**Parameters**

params: Key-value pair of filters for this search.
(*type=dictionary*)

```
close(self, euids=None, excludeIds=None, params=None, uid=None,  
asof=None, limit=None)
```

Close event(s).

Parameters

- euids:** (optional) List of event IDs to close (default: None)
(*type=[string]*)
- excludeIds:** (optional) List of event IDs to exclude from close
(default: None)
(*type=[string]*)
- params:** (optional) Key-value pair of filters for this search.
(default: None)
(*type=dictionary*)
- uid:** (optional) Context for the query (default: None)
(*type=string*)
- asof:** (optional) Only close if there has been no state change
since this time (default: None)
(*type=float*)
- limit:** (optional) Maximum number of events to update
(default: None).
(*type=The maximum number of events to update in
this batch.*)

Return Value

- Success message
(*type=DirectResponse*)

acknowledge(*self*, *evids*=None, *excludeIds*=None, *params*=None, *uid*=None, *asof*=None, *limit*=None)

Acknowledge event(s).

Parameters

- evids:** (optional) List of event IDs to acknowledge (default: None)
(*type*=[string])
- excludeIds:** (optional) List of event IDs to exclude from acknowledgment (default: None)
(*type*=[string])
- params:** (optional) Key-value pair of filters for this search. (default: None)
(*type*=dictionary)
- uid:** (optional) Context for the query (default: None)
(*type*=string)
- asof:** (optional) Only acknowledge if there has been no state change since this time (default: None)
(*type*=float)
- limit:** (optional) Maximum number of events to update (default: None).
(*type*=The maximum number of events to update in this batch.)

Return Value

Success message
(*type*=DirectResponse)

unacknowledge(*self*, **args*, ***kwargs*)

Deprecated, Use reopen


```
reopen(self, evids=None, excludeIds=None, params=None, uid=None, asof=None, limit=None)
```

Reopen event(s).

Parameters

- evids:** (optional) List of event IDs to reopen (default: None)
(*type=[string]*)
- excludeIds:** (optional) List of event IDs to exclude from reopen
(default: None)
(*type=[string]*)
- params:** (optional) Key-value pair of filters for this search.
(default: None)
(*type=dictionary*)
- uid:** (optional) Context for the query (default: None)
(*type=string*)
- asof:** (optional) Only reopen if there has been no state change since this time (default: None)
(*type=float*)
- limit:** (optional) Maximum number of events to update
(Default: None).
(*type=The maximum number of events to update in this batch.*)

Return Value

Success message
(*type=DirectResponse*)

```
updateEventSummaries(self, update, event_filter=None, exclusion_filter=None, limit=None)
```

add_event(*self*, *summary*, *device*, *component*, *severity*, *evclasskey*,
evclass=None)

Create a new event.

Parameters

- summary:** New event's summary
(*type=string*)
- device:** Device uid to use for new event
(*type=string*)
- component:** Component uid to use for new event
(*type=string*)
- severity:** Severity of new event. Can be one of the following:
Critical, Error, Warning, Info, Debug, or Clear
(*type=string*)
- evclasskey:** The Event Class Key to assign to this event
(*type=string*)
- evclass:** Event class for the new event
(*type=string*)

Return Value

DirectResponse

configSchema(*self*)

getConfig(*self*)

setConfigValues(*self*, *values*)

Parameters

- values:** Key Value pairs of config values
(*type=Dictionary*)

column_config(*self*, *uid*=None, *archive*=False)

Get the current event console field column configuration.

Parameters

- uid:** (optional) UID context to use (default: None)
(*type*=string)
- archive:** (optional) True to use the event archive instead of active events (default: False)
(*type*=boolean)

Return Value

A list of objects representing field columns
(*type*=[dictionary])

classify(*self*, *evrows*, *evclass*)

Associate event(s) with an event class.

Parameters

- evrows:** List of event rows to classify
(*type*=[dictionary])
- evclass:** Event class to associate events to
(*type*=string)

Return Value**Properties:**

- **msg:** (string) Success/failure message
- **success:** (boolean) True if class update successful

(*type*=`DirectResponse`)

clear_heartbeats(*self*)

Clear all heartbeat events

Return Value**Properties:**

- **success:** (boolean) True if heartbeats deleted successfully

(*type*=`DirectResponse`)

Index

Products (*package*)

 Products.Zuul (*package*)

 Products.Zuul.routers (*package*), 2–4