API Documentation

API Documentation

September 22, 2010

Contents

1 Package apidoc.Zuul.routers

Zenoss JSON API

1.1 Modules

- **device**: Operations for Device Organizers and Devices.
  (Section 2, p. 4)
- **events**: Operations for Events.
  (Section 3, p. 31)
- **messaging**: Operations for Messaging.
  (Section 4, p. 41)
- **mibs**: Operations for MIBs.
  (Section 5, p. 42)
- **nav**: Operations for Navigation
  (Section 6, p. 45)
- **network**: Operations for Networks.
  (Section 7, p. 46)
- **process**: Operations for Processes.
  (Section 8, p. 49)
- **report**: Operations for Reports.
  (Section 9, p. 52)
- **service**: Operations for Services.
  (Section 10, p. 54)
- **template**: Operations for Templates.
  (Section 11, p. 59)
- **zenpack**: Operations for ZenPacks.
  (Section 12, p. 71)

1.2 Variables

Name	Description
log	**Value:** `logging.getLogger(__name__)`

1.3 Class TreeRouter

```
Products.ZenUtils.Ext.DirectRouter ─┐
                                    apidoc.Zuul.routers.TreeRouter
```

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

1.3.1 Methods

addNode(*self*, *type*, *contextUid*, *id*, *description*=`None`)

Add a node to the existing tree underneath the node specified by the context UID

Parameters

`type:`	Either 'class' or 'organizer' *(type=string)*
`contextUid:`	Path to the node that will be the new node's parent (ex. /zport/dmd/Devices) *(type=string)*
`id:`	Identifier of the new node, must be unique in the parent context *(type=string)*
`description:`	(optional) Describes this new node (default: None) *(type=string)*

Return Value

Marshaled form of the created node

(type=dictionary)

deleteNode(*self*, *uid*)

Deletesa 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 *(type=string)*

Return Value

Properties:

- msg: (string) Status message

(type=DirectResponse)

moveOrganizer(*self*, *targetUid*, *organizerUid*)

Move the organizer uid to be underneath the organizer specified by the targetUid.

Parameters

`targetUid:`	New parent of the organizer *(type=string)*
`organizerUid:`	The organizer to move *(type=string)*

Return Value

Properties:

- data: (dictionary) Moved organizer

(type=DirectResponse)

2 Module apidoc.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

Products.Zuul.routers.TreeRouter

apidoc.Zuul.routers.device.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`, *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(*self*, *uid*)

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

`uid`: Unique identifier of an object

(type=string)

Return Value

Properties

- form: (dictionary) form fields for the object

(type=DirectResponse)

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

Get the properties of a device or device organizer

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 (default: None)

(type=list)

Return Value

Properties

- data: (dictionary) Object properties
- disabled: (bool) If current user doesn't have permission to use setInfo

(type=DirectResponse)

setInfo(*self*, ***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

`uid`: Unique identifier of an object

(type=string)

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

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

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)

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

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'`)

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'`)

Remove/delete device(s).

Parameters

`uids:` List of device uids to remove

(type=[string])

`hashcheck:` Hashcheck for the devices (from getDevices())

(type=string)

`action:` Action to take. 'remove' to remove devices from organizer uid, and 'delete' to delete the device from Zenoss.

(type=string)

`uid:` (optional) Organizer to use when using ranges to get additional uids and/or to remove device (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

Properties:

- devtree: ([dictionary]) Object representing the new device tree
- grptree: ([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)

getEvents(*self*, *uid*)

Get events for a device.

Parameters

`uid`: Device to get events for

(type=[string])

Return Value

Properties:

- data: ([dictionary]) List of events for a device

(type=DirectResponse)

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(*self*, *uid*)

Get a list of user commands for a device uid.

Parameters

`uid`: Device to use to get user commands

(type=string)

Return Value

List of objects representing user commands

(type=[dictionary])

getProductionStates(*self*, ***kwargs*)

Get a list of available production states.

Return Value

List of name/value pairs of available production states

(type=[dictionary])

getPriorities(*self*, ***kwargs*)

Get a list of available device priorities.

Return Value

List of name/value pairs of available device priorities

(type=[dictionary])

getCollectors(*self*)

Get a list of available collectors.

Return Value

List of collectors

(type=[string])

getDeviceClasses(*self*, ***data*)

Get a list of all device classes.

Return Value

Properties:

- deviceClasses: ([dictionary]) List of device classes
- totalCount: (integer) Total number of device classes

(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`, *model*=`False`, *collector*=`'localhost'`, *rackSlot*=`0`, *productionState*=`1000`, *comments*=`""`, *hwManufacturer*=`""`, *hwProductName*=`""`, *osManufacturer*=`""`, *osProductName*=`""`, *priority*=`3`, *tag*=`""`, *serialNumber*=`""`)

Add a device.

Parameters

`deviceName:`	Name or IP of the new device *(type=string)*
`deviceClass:`	The device class to add new device to *(type=string)*
`title:`	(optional) The title of the new device (default: ") *(type=string)*
`snmpCommunity:`	(optional) A specific community string to use for this device. (default: ") *(type=string)*
`snmpPort:`	(optional) SNMP port on new device (default: 161) *(type=integer)*
`model:`	(optional) True to model device at add time (default: False) *(type=boolean)*
`collector:`	(optional) Collector to use for new device (default: localhost) *(type=string)*
`rackSlot:`	(optional) Rack slot description (default: ") *(type=string)*
`productionState:`	(optional) Production state of the new device (default: 1000) *(type=integer)*
`comments:`	(optional) Comments on this device (default: ") *(type=string)*
`hwManufacturer:`	(optional) Hardware manufacturer name (default: ") *(type=string)*
`hwProductName:`	(optional) Hardware product name (default: ") *(type=string)*
`osManufacturer:`	(optional) OS manufacturer name (default: ") *(type=string)*
`osProductName:`	(optional) OS product name (default: ") *(type=string)*
`priority:`	(optional) Priority of this device (default: 3) *(type=integer)*
`tag:`	(optional) Tag number of this device (default: ") *(type=string)*
`serialNumber:`	(optional) Serial number of this device (default: ") *(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(*self*, *query*, *uid*)

Get a list of locally defined templates on a device.

Parameters

`query`: not used

(type=string)

`uid`: Device uid to query for templates

(type=string)

Return Value

Properties:

- data: ([dictionary]) List of objects representing local templates

(type=DirectResponse)

getTemplates(*self*, *id*)

Get a list of available templates for a device.

Parameters

`id`: Device uid to query for templates

(type=string)

Return Value

Properties:

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

(type=DirectResponse)

getUnboundTemplates(*self*, *uid*)

Get a list of unbound 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)

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(*self*, *query*, *uid*)

Get a list of available templates on a device that can be overridden.

Parameters

`query`: not used

(type=string)

`uid`: Device to query for overridable templates

(type=string)

Return Value

Properties:

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

(type=DirectResponse)

clearGeocodeCache(*self*)

Clear the Google Maps geocode cache.

Return Value

Success message

(type=DirectResponse)

3 Module apidoc.Zuul.routers.events

Operations for Events.

Available at: /zport/dmd/evconsole_router

3.1 Variables

Name	Description
log	**Value:** `logging.getLogger('zen.event_router')`

3.2 Class EventsRouter

Products.ZenUtils.Ext.DirectRouter

apidoc.Zuul.routers.events.EventsRouter

A JSON/ExtDirect interface to operations on events

3.2.1 Methods

__init__(*self*, *context*, *request*)

query(*self*, *limit*=`0`, *start*=`0`, *sort*=`'lastTime'`, *dir*=`'DESC'`, *params*=`None`, *history*=`False`, *uid*=`None`, *criteria*=`()`)

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)*
`criteria:`	(optional) A list of key-value pairs to to build query's where clause (default: None) *(type=[dictionary])*

Return Value

Properties:

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

(type=dictionary)

queryHistory(*self*, *limit*, *start*, *sort*, *dir*, *params*)

Query history table 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)*

Return Value

Properties:

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

(type=dictionary)

acknowledge(*self*, *evids*=`None`, *excludeIds*=`None`, *selectState*=`None`, *field*=`None`, *direction*=`None`, *params*=`None`, *history*=`False`, *uid*=`None`, *asof*=`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])*
`selectState:`	(optional) Select event ids based on select state. Available values are: All, New, Acknowledged, and Suppressed (default: None) *(type=string)*
`field:`	(optional) Field key to filter gathered events (default: None) *(type=string)*
`direction:`	(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 use the event history table instead of active events (default: False) *(type=boolean)*
`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)*

Return Value

Success message

(type=DirectResponse)

unacknowledge(*self*, *evids*=`None`, *excludeIds*=`None`, *selectState*=`None`, *field*=`None`, *direction*=`None`, *params*=`None`, *history*=`False`, *uid*=`None`, *asof*=`None`)

Unacknowledge event(s).

Parameters

`evids:`	(optional) List of event IDs to unacknowledge (default: None) *(type=[string])*
`excludeIds:`	(optional) List of event IDs to exclude from unacknowledgment (default: None) *(type=[string])*
`selectState:`	(optional) Select event ids based on select state. Available values are: All, New, Acknowledged, and Suppressed (default: None) *(type=string)*
`field:`	(optional) Field key to filter gathered events (default: None) *(type=string)*
`direction:`	(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 use the event history table instead of active events (default: False) *(type=boolean)*
`uid:`	(optional) Context for the query (default: None) *(type=string)*
`asof:`	(optional) Only unacknowledge if there has been no state change since this time (default: None) *(type=float)*

Return Value

Success message

(type=DirectResponse)

reopen(*self*, *evids*=`None`, *excludeIds*=`None`, *selectState*=`None`, *field*=`None`, *direction*=`None`, *params*=`None`, *history*=`False`, *uid*=`None`, *asof*=`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])*
`selectState:`	(optional) Select event ids based on select state. Available values are: All, New, Acknowledged, and Suppressed (default: None) *(type=string)*
`field:`	(optional) Field key to filter gathered events (default: None) *(type=string)*
`direction:`	(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 use the event history table instead of active events (default: False) *(type=boolean)*
`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)*

Return Value

Success message

(type=DirectResponse)

close(*self*, *evids*=`None`, *excludeIds*=`None`, *selectState*=`None`, *field*=`None`, *direction*=`None`, *params*=`None`, *history*=`False`, *uid*=`None`, *asof*=`None`)

Close event(s).

Parameters

`evids:`	(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])*
`selectState:`	(optional) Select event ids based on select state. Available values are: All, New, Acknowledged, and Suppressed (default: None) *(type=string)*
`field:`	(optional) Field key to filter gathered events (default: None) *(type=string)*
`direction:`	(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 use the event history table instead of active events (default: False) *(type=boolean)*
`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)*

Return Value

Success message

(type=DirectResponse)

detail(*self*, *evid*, *history*=`False`)

Get event details.

Parameters

`evid:` Event ID to get details

(type=string)

`history:` (optional) True to search the event history table instead of active events (default: False)

(type=boolean)

Return Value

Properties:

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

(type=DirectResponse)

write_log(*self*, *evid*=`None`, *message*=`None`, *history*=`False`)

Write a message to an event's log.

Parameters

`evid:` Event ID to log to

(type=string)

`message:` Message to log

(type=string)

`history:` (optional) True to use the event history table instead of active events (default: False)

(type=boolean)

Return Value

Success message

(type=DirectResponse)

classify(*self*, *evids*, *evclass*, *history*=`False`)

Associate event(s) with an event class.

Parameters

`evids`: List of event ID's to classify

(type=[string])

`evclass`: Event class to associate events to

(type=string)

`history`: (optional) True to use the event history table instead of active events (default: False)

(type=boolean)

Return Value

Properties:

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

(type=DirectResponse)

add_event(*self*, *summary*, *device*, *component*, *severity*, *evclasskey*, *evclass*)

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

Properties:

- evid: (string) The id of the created event

(type=DirectResponse)

column_config(*self*, *uid*=`None`, *history*=`False`)

Get the current event console field column configuration.

Parameters

`uid:` (optional) UID context to use (default: None)

(type=string)

`history:` (optional) True to use the event history table instead of active events (default: False)

(type=boolean)

Return Value

A list of objects representing field columns

(type=[dictionary])

4 Module apidoc.Zuul.routers.messaging

Operations for Messaging.

Available at: /zport/dmd/messaging_router

4.1 Class MessagingRouter

Products.ZenUtils.Ext.DirectRouter ─┐
apidoc.Zuul.routers.messaging.MessagingRouter

A JSON/ExtDirect interface to operations on messages

4.1.1 Methods

__init__(*self*, *context*, *request*)

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 apidoc.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

Products.Zuul.routers.TreeRouter ─┐
apidoc.Zuul.routers.mibs.MibRouter

A JSON/ExtDirect interface to operations on MIBs

5.2.1 Methods

__init__(*self*, *context*, *request*)

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

Returns the tree structure of an organizer hierarchy. Default tree root is MIBs.

Parameters

`id`: (optional) Id of the root node of the tree to be returned (default: '/zport/dmd/Mibs')

(type=string)

Return Value

Object representing the tree

(type=[dictionary])

getOrganizerTree(*self*, *id*)

Returns the tree structure of an organizer hierarchy, only including organizers.

Parameters

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

(type=string)

Return Value

Object representing the organizer tree

(type=[dictionary])

addNode(*self*, *contextUid*=`''`, *id*=`''`, *type*=`''`)

Add an organizer or new blank MIB.

Parameters

`contextUid`: Context to attach new node

(type=string)

`id`: Id of the new orgainzer or blank MIB

(type=string)

`type`: Type of new node. Can be 'organizer' or 'MIB'

(type=string)

Return Value

Properties:

- tree: ([dictionary]) Object representing the new tree

(type=DirectResponse)

addMIB(*self*, *package*, *organizer*=`'/'`)

Add a new MIB by URL or local file.

Parameters

`package`: URL or local file path to MIB file

(type=string)

`organizer`: ID of the organizer to add MIB to

(type=string)

Return Value

Properties:

- jobId: (string) ID of the add MIB job

(type=DirectResponse)

deleteNode(*self*, *uid*)

Remove an organizer or MIB.

Parameters

`uid`: UID of organizer or MIB to remove

(type=string)

Return Value

Properties:

- tree: ([dictionary]) Object representing the new tree

(type=DirectResponse)

moveNode(*self*, *uids*, *target*)

Move an organizer or MIB from one organizer to another.

Parameters

`uids`: UIDs of organizers and MIBs to move

(type=[string])

`target`: UID of the organizer to move to

(type=string)

Return Value

Properties:

- data: (dictionary) Object representing the new parent organizer

(type=DirectResponse)

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)

6 Module apidoc.Zuul.routers.nav

Operations for Navigation

Available at: /zport/dmd/detailnav_router

6.1 Class DetailNavRouter

Products.ZenUtils.Ext.DirectRouter ─┐

apidoc.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 apidoc.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

Products.ZenUtils.Ext.DirectRouter

apidoc.Zuul.routers.network.NetworkRouter

A JSON/ExtDirect interface to operations on networks

7.2.1 Methods

__init__(*self*, *context*, *request*)

discoverDevices(*self*, *uid*)

Discover devices on a network.

Parameters

`uid`: Unique identifier of the network to discover

(type=string)

Return Value

Properties:

- jobId: (integer) The id of the discovery job

(type=DirectResponse)

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)

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)

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 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

`uid`: Unique identifier of an object

(type=string)

Return Value

DirectResponse

getIpAddresses(*self*, *uid*, *start*=`0`, *params*=`None`, *limit*=`50`, *sort*=`'name'`, *order*=`'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

8 Module apidoc.Zuul.routers.process

Operations for Processes.

Available at: /zport/dmd/process_router

8.1 Class ProcessRouter

Products.Zuul.routers.TreeRouter ─┐
apidoc.Zuul.routers.process.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(*self*, *uid*, *targetUid*)

Move a process or organizer from one organizer to another.

Parameters

`uid`: UID of the process or organizer to move

(type=string)

`targetUid`: UID of the organizer to move to

(type=string)

Return Value

Properties:

- uid: (dictionary) The new uid for moved process or organizer

(type=DirectResponse)

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

Get the properties of a process.

Parameters

`uid`: Unique identifier of a process

(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)

Return Value

Properties

- data: (dictionary) Object representing a process's properties

(type=DirectResponse)

setInfo(*self*, ***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

(type=string)

Return Value

Properties

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

(type=DirectResponse)

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

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)

9 Module apidoc.Zuul.routers.report

Operations for Reports.

Available at: /zport/dmd/report_router

9.1 Variables

Name	Description
log	**Value:** `logging.getLogger('zen.ReportRouter')`

9.2 Class ReportRouter

```
Products.ZenUtils.Ext.DirectRouter ─┐
                                    │
                 apidoc.Zuul.routers.report.ReportRouter
```

A JSON/ExtDirect interface to operations on reports

9.2.1 Methods

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)

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

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

Parameters

`id`: (optional) Id of the root node of the tree to be returned (default: Reports)

(type=string)

Return Value

Object representing the tree

(type=[dictionary])

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)

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])

moveNode(*self*, *uids*, *target*)

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

Parameters

`uids`: The UID's of nodes 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])

10 Module apidoc.Zuul.routers.service

Operations for Services.

Available at: /zport/dmd/service_router

10.1 Class ServiceRouter

Products.Zuul.routers.TreeRouter
└─ **apidoc.Zuul.routers.service.ServiceRouter**

A JSON/ExtDirect interface to operations on services

10.1.1 Methods

__init__(*self*, *context*, *request*)

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

Add a new service class.

Parameters

`contextUid:`	Unique ID of the service ogranizer 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`, *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(*self*, *id*)

Returns the tree structure of an organizer hierarchy.

Parameters

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

(type=string)

Return Value

Object representing the tree

(type=[dictionary])

getOrganizerTree(*self*, *id*)

Returns the tree structure of an organizer hierarchy, only including organizers.

Parameters

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

(type=string)

Return Value

Object representing the organizer tree

(type=[dictionary])

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

Get the properties of a service.

Parameters

`uid`: Unique identifier of a service

(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)

Return Value

Properties

- data: (dictionary) Object representing a service's properties
- disabled: (boolean) True if current user cannot manage services

(type=DirectResponse)

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'`, *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(*self*, *sourceUids*, *targetUid*)

Move service(s) from one organizer to another.

Parameters

`sourceUids:` UID(s) of the service(s) to move

(type=[string])

`targetUid:` UID of the organizer to move to

(type=string)

Return Value

Success messsage

(type=DirectResponse)

getUnmonitoredStartModes(*self*, *uid*)

Get a list of unmonitored start modes for a Windows service.

Parameters

`uid`: Unique ID of a Windows service.

(type=string)

Return Value

Properties:

- data: ([string]) List of unmonitored start modes for a Windows service

(type=DirectResponse)

getMonitoredStartModes(*self*, *uid*)

Get a list of monitored start modes for a Windows service.

Parameters

`uid`: Unique ID of a Windows service.

(type=string)

Return Value

Properties:

- data: ([string]) List of monitored start modes for a Windows service

(type=DirectResponse)

11 Module apidoc.Zuul.routers.template

Operations for Templates.

Available at: /zport/dmd/template_router

11.1 Class TemplateRouter

Products.ZenUtils.Ext.DirectRouter ─┐
apidoc.Zuul.routers.template.TemplateRouter

A JSON/ExtDirect interface to operations on templates

11.1.1 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*)

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)

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*, *query*, *uid*)

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*, *id*)

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*)

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(*self*, *graphUid*, *customId*, *customType*)

Add a custom graph point to a graph definition.

Parameters

`graphUid`: Unique ID of the graph definition to add graph point to
(type=string)

`customId`: ID of the new custom graph point
(type=string)

`customType`: Type of the new graph point. From getGraphInstructionTypes()
(type=string)

Return Value

Success message
(type=DirectResponse)

getGraphInstructionTypes(*self*, *query*=`''`)

Get a list of available instruction types for graph points.

Parameters

`query`: not used
(type=string)

Return Value

Properties:

- data: ([dictionary]) List of objects representing instruction types

(type=DirectResponse)

setGraphPointSequence(*self*, *uids*)

Sets the sequence of graph points in a graph definition.

Parameters

`uids`: List of graph point UID's in desired order
(type=[string])

Return Value

Success message
(type=DirectResponse)

getGraphDefinition(*self*, *uid*)

Get a graph definition.

Parameters

`uid`: Unique ID of the graph definition to retrieve

(type=string)

Return Value

Properties:

- data: (dictionary) Object representing a graph definition

(type=DirectResponse)

setGraphDefinition(*self*, ***data*)

Set attributes on an graph definition. This method accepts any keyword argument for the property that you wish to set. Properties are enumerated via getGraphDefinition(). 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)

setGraphDefinitionSequence(*self*, *uids*)

Sets the sequence of graph definitions.

Parameters

`uids`: List of graph definition UID’s in desired order

(type=[string])

Return Value

Success message

(type=DirectResponse)

12 Module apidoc.Zuul.routers.zenpack

Operations for ZenPacks.

Available at: /zport/dmd/zenpack_router

12.1 Variables

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

12.2 Class ZenPackRouter

Products.ZenUtils.Ext.DirectRouter

apidoc.Zuul.routers.zenpack.ZenPackRouter

A JSON/ExtDirect interface to operations on ZenPacks

12.2.1 Methods

getEligiblePacks(*self*, ***data*)

Get a list of eligible ZenPacks to add to.

Return Value

Properties:

- packs: ([dictionary]) List of objects representing ZenPacks
- totalCount: (integer) Total number of eligible ZenPacks

(type=DirectResponse)

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)

Index