

TuxCare Portal APIs

API URL

Root URL for all REST services is: <https://portal.tuxcare.com/api>

API overview

Authentication token is produced by combining: `user_login|timestamp|sha1(secret_key + timestamp)`.

```
customer_login|1401189361|97e17a41ea904e0ca2bf03c7c36dee2492ba89cd
```

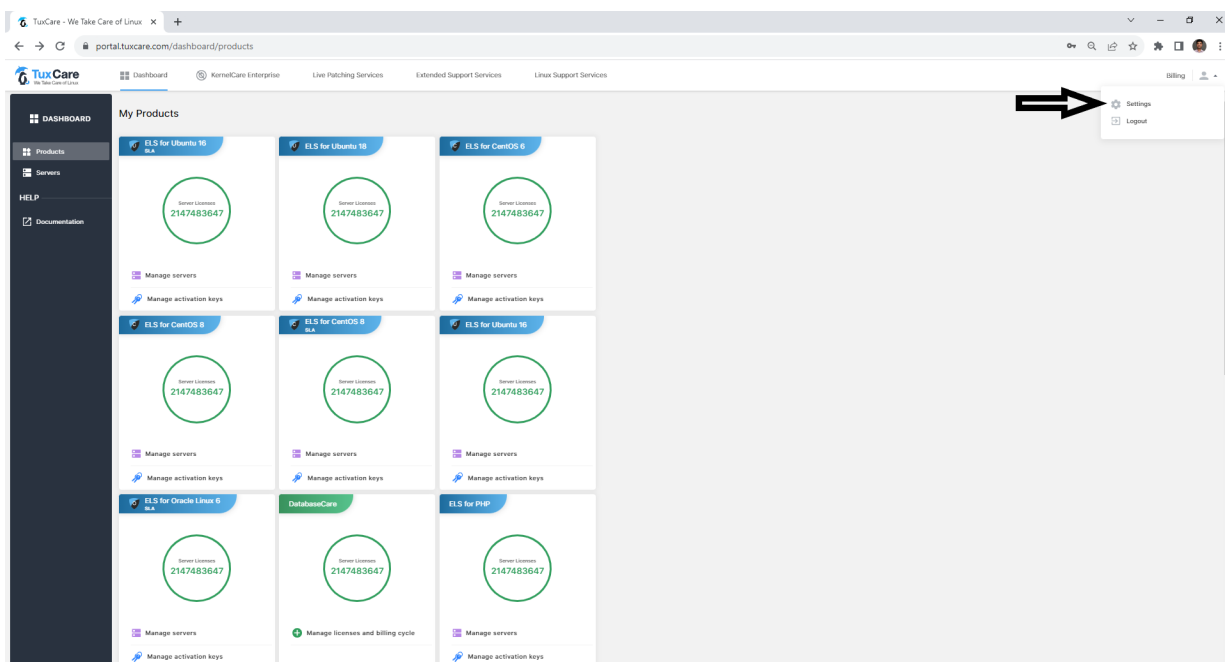
The timestamp is a number of seconds since January 1, 1970 (UTC), or POSIX time.
<https://www.epochconverter.com/> - to find the current POSIX time

Date/time formatting

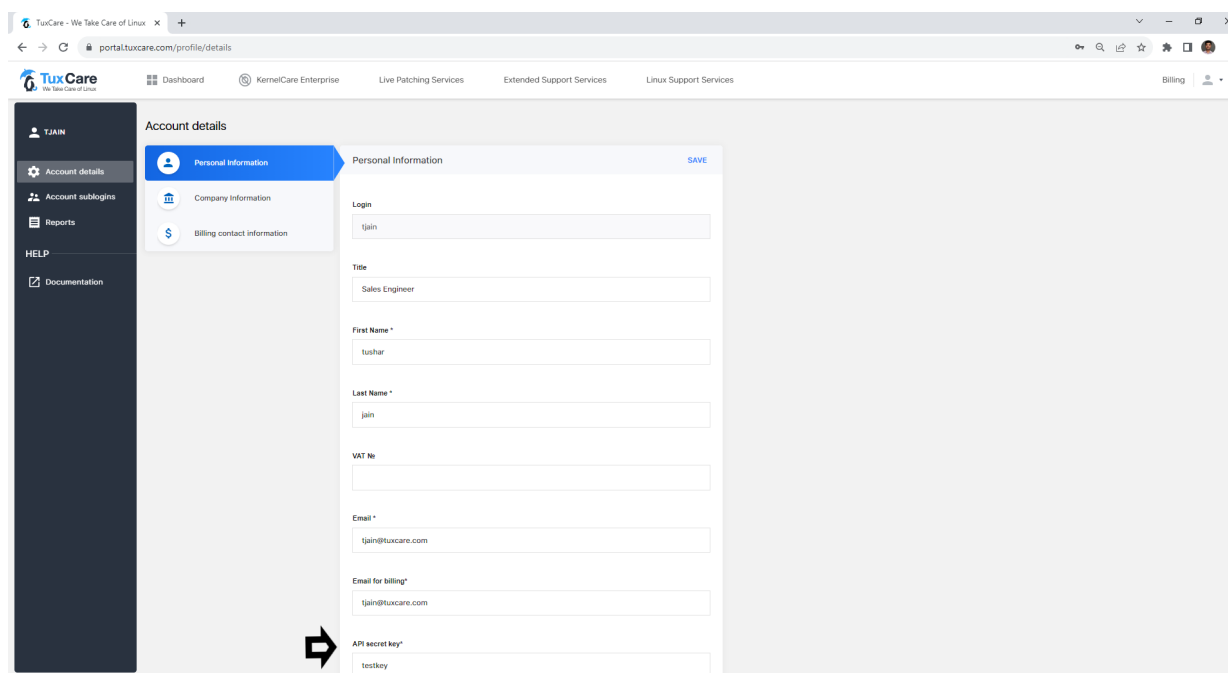
All date/time values returned from API will be formatted as a string in ISO-8601:
"yyyy-MM-ddTHH:mmZ" = 04-30T11:26-0400

Secret Key

Step 1 → To generate a secret key, log in to portal.tuxcare.com with your credentials.
Step 2 → Click on Settings Tab:




Step 3 → Under Personal Information, inside the API secret key tab, configure a secret key:



The screenshot shows the TuxCare portal's 'Account details' page. The 'Personal Information' tab is selected, and the 'API secret key' field is highlighted with a red arrow. The field contains the value 'testkey'.

sha(secret_key + timestamp)

Use one of the sha online tools <http://www.sha1-online.com/> to generate the hash of the combination of (secret key + timestamp).



The screenshot shows the SHA1 online generator website. The input field contains the text 'testkey1692699188'. The 'hash' button is clicked, and the result for sha1 is displayed as '35b8466e0235170a6354bb71c11cc35b4dca5e2b'.

NOTE: The Authentication Token generated is only valid for 30 minutes. After that, the above process needs to be repeated to generate a new hash with the current POSIX time.

ELS Products API

API root URL: <https://portal.tuxcare.com/api/els>

API for interactions with Extended Life Support products (CentOS/Ubuntu/Oracle).

ELS product codes

ELS key has **code** to define its product:

- CELS - "CentOS 6 ELS"
- OELS - "Oracle ELS"
- UELS16 - "Ubuntu 16 ELS"
- CELS_8 - "CentOS 8 ELS"
- UELS18 - "Ubuntu 18 ELS"
- PHP_ELS - "PHP ELS"
- PYTHON_ELS - "PYTHON ELS"
- ELS_FOR_CENTOS6_PLUS_SLA - "CentOS 6 ELS with SLA"
- ELS_FOR_ORACLE_LINUX_PLUS_SLA - "Oracle ELS with SLA"
- ELS_FOR_CENTOS8_PLUS_SLA - "CentOS 8 ELS with SLA"
- UELS18_SLA - "Ubuntu 18 ELS with SLA"
- UELS16_SLA - "Ubuntu 16 ELS with SLA"
- ELS_FOR_CENTOS7 - "CentOS 7 ELS"
- ELS_FOR_CENTOS7_PLUS_SLA - "CentOS 7 ELS with ELS"
- EXTENDED_SECURITY_UPDATES - "Extended Security Updates"

Create License Key /key/create.json

URL:

https://portal.tuxcare.com/api/els/key/create.json?token=AUTH_TOKEN&productType=ProductCode&limit=2¬e=12

POST arguments:

- **token** - authorization token
- **code** - ELS license code (see description above)
- **note** - key note
- **limit** - key limit (max servers per key) 0 for unlimited key

Update License Key /key/update.json

URL:

https://portal.tuxcare.com/api/els/key/update.json?token=AUTH_TOKEN&key=CELS-key&limit=4&no

PUT arguments:

- **token** - authorization token
- **key** - key to update
- **limit** - key limit (max servers per key) 0 for unlimited key
- **note** - key note

Remove License Key /key/remove.json

URL:

https://portal.tuxcare.com/api/els/key/remove.json?token=AUTH_TOKEN&key=CELS-key

DELETE arguments:

- **token** - authorization token
- **key** - ELS key to remove

List All ELS Keys /key/list.json

URL:

https://portal.tuxcare.com/api/els/key/list.json?token=AUTH_TOKEN&productType=ProductCode

GET arguments:

- **token** - authorization token
- **code** – product code

List all ELS keys owned by the customer.

Success response data (json): Returns list of key objects

List Servers Registered to Key /srv/list.json

URL:

https://portal.tuxcare.com/api/els/srv/list.json?token=AUTH_TOKEN&productType=ProductCode

GET arguments:

- **token** - authorization token
- **productType** – ELS license code (see description above)

List all ELS servers under the specific key.

Success response data (json): Returns a list of server objects or an empty list

Remove Server Registered to Key /srv/remove.json

URL:

https://portal.tuxcare.com/api/els/srv/remove.json?token=AUTH_TOKEN&id=server_id

DELETE arguments:

- **token** - authorization token
- **id** - server id to remove

AlmaLinux API

API root URL: <https://portal.tuxcare.com/api/els>

API for interactions with AlmaLinux products.

AlmaLinux product codes

AlmaLinux key has a **code** to define its product:

- EXTENDED_SECURITY_UPDATES - "Extended Security Updates"

/key/create.json

URL:

https://portal.tuxcare.com/api/els/key/create.json?token=AUTH_TOKEN&code=ESU&limit=2¬e=12

GET arguments:

- **token** - authorization token
- **code** - AlmaLinux license code (see description above)
- **note** - key note
- **limit** - key limit (max servers per key) 0 for unlimited key

Create AlmaLinux key.

Success response data (json): Returns newly generated AlmaLinux key object

/key/update.json

```
{
  "key": "ESU-yUDxBd5ethmGO5d3hnAZEau0",
  "productCode": "ESU"
  "usageLimit": "2"
  "description": "123"
}
```

URL:

https://portal.tuxcare.com/api/els/key/update.json?token=AUTH_TOKEN&key=ESU-key&limit=4¬e=

GET arguments:

- **token** - authorization token
- **key** - key to update
- **limit** - key limit (max servers per key) 0 for unlimited key
- **note** - key note

Update AlmaLinux key properties.

Success response data (json): Returns updated AlmaLinux key object

```
{
  "key": "ESU-yUDxBd5ethmGO5d3hnAZEau0",
  "productCode": "ESU", "usageLimit":
  "4"
  "description": "321"
}
```

/key/remove.json

URL:

https://portal.tuxcare.com/api/els/key/remove.json?token=AUTH_TOKEN&key=ESU-key

GET arguments:

- **token** - authorization token
- **key** - AlmaLinux key to remove

Remove AlmaLinux registration key with all servers.

Success response data (json): Returns list of server objects removed with the current key.

```
[
  {"key": "ESU-yUDxBd5ethmGO5d3hnAZEau0"}
]
```

/key/list.json

URL: https://portal.tuxcare.com/api/els/key/list.json?token=AUTH_TOKEN&code=ESU

GET arguments:

- **token** - authorization token
- **code** – product code

List all AlmaLinux keys owned by customer.

Success response data (json): Returns a list of key objects.

/srv/list.json

```
[
  {"key": "ESU-yUDxBd5ethmGO5d3hnAZEau0", "productCode": "ESU",
  "usageLimit": "2", "description": 123}
  {"key": "ESU-kdiRMfJ4yhmF7je4ldptndYk", "productCode": "ESU",
  "usageLimit": "5", "description": null}
]
```

URL: https://portal.tuxcare.com/api/els/srv/list.json?token=AUTH_TOKEN&code=ESU

GET arguments:

- token - authorization token
- code – product code

List all AlmaLinux servers under specific key

Success response data (json): Returns list of server objects or an empty list

```
[
  {"id": "12345", "key": "ESU-value", "ip": "1.1.1.1", "hostname":
  "test"}
  {"id": "12346", "key": "ESU-value", "ip": "1.1.1.1", "hostname":
  "test2"}
]
```

/srv/remove.json

URL:

https://portal.tuxcare.com/api/els/key/remove.json?token=AUTH_TOKEN&key=ESU-value

GET arguments:

- **token** - authorization token
- **id** - server id to remove

Remove AlmaLinux server by its ID.

Success response data (json): Returns removed server object

```
{ "id": "12345", "key": "ESU-value", "ip": "1.1.1.1", "hostname": "test" }
```

IP-based licenses API

API root URL: <https://portal.tuxcare.com/api/ipl>

You can manage an IP-based license over this API.

IP license product types

To bind an IP license with a particular product you must provide a valid product type:

- 1 - "CloudLinux OS"
- 4 - "CloudLinux Solo"
- 5 - "CloudLinux Admin"
- 1002 - "CloudLinux OS Shared Pro"
- 16 - "KernelCare Enterprise"
- 17 - "KernelCare Enterprise Plus"
- 40 - "ImunifyAV+"
- 41 - "Imunify360 single user"
- 42 - "Imunify360 up to 30 users"
- 43 - "Imunify360 up to 250 users"
- 49 - "Imunify360 unlimited"

/availability.json

URL example:

https://portal.tuxcare.com/api/ipl/availability.json?ip=1.1.1.1&token=AUTH_TOKEN

Will return information about what kind of license types are available for registration and what types are used by current account.

GET arguments:

- **token** - authorization token
- **ip** - IP address to check

Success response data:

```
{ "available": [1,41,42,43,49], "owned":[16] }
```

- **available(int[])** - list of types that can be used to register new IP license
- **owned(int[])** - list of types that already registered(owned) by this account

As you can see if somebody owns a license than that license type will not be in **available** list.

/check.json

URL example:

https://portal.tuxcare.com/api/ipl/check.json?ip=1.1.1.1&token=AUTH_TOKEN

Check if the IP license is registered by any customer.

GET arguments:

- **token** - authorization token
- **ip** - IP address to check

Success response data (list of integers):

```
[1,16] OR [41] OR []
```

Will return a list of registered license types or empty list if provided IP is not registered yet.

/register.json

URL example:

https://portal.tuxcare.com/api/ipl/register.json?ip=1.1.1.1&type=1&token=AUTH_TOKEN

Will register IP-based license for an authorized user.

GET arguments:

- **token** - authorization token
- **ip** - IP address to register
- **type** - IP license type (1,16,41,42,43,49)
- **els_allowed** - allow CL6 registration

On success response **returns** information about created or already registered license.

Success response data (json object):

```
{"ip": "1.1.1.1", "type": ,16 "registered": false, "created":  
"2014-04-30T11:26-0400"}
```

- **ip(string)**
- **type(int)** - license type (1,16,41,42,43,49)
- **registered(boolean)** - true if the server was registered in CLN with this license (CLN licenses only).
- **created(string)** - license creation time

Will return a non-successful response in any other cases.

/remove.json

URL example:

https://portal.tuxcare.com/ipl/remove.json?ip=1.1.1.1&type=16&token=AUTH_TOKEN

Will remove an IP-based license from the authorized user licenses.

GET arguments:

- **token** - authorization token
- **ip** - IP address to remove
- **type**(optional) - IP license type. If empty - will remove licenses with all types

Success response data (boolean):

- **true** when IP license was removed
- **false** if IP license was not found OR not owned by user

/list.json

URL example: https://portal.tuxcare.com/api/ipl/list.json?token=AUTH_TOKEN

Return all IP licenses owned by an authorized user.

GET arguments:

- **token** - authorization token

Success response data (json objects list):

```
[{"ip": "1.1.1.1", "type": ,16 "registered": false, "created":  
"2014-04-30T11:26- 0400"}, ...]
```

- **ip(string)**
- **type(int)** - license type (1,16,41,42,43,49)
- **registered(boolean)** - true if the server was registered in CLN with this license (CLN licenses only).
- **created(string)** - license creation time

/server.json

URL example:

https://portal.tuxcare.com/api/ipl/server.json?token=AUTH_TOKEN&ip=1.1.1.1

Return all IP licenses owned by authorized user filtered by ip.

GET arguments:

- **token** - authorization token
- **ip** - IP address to fetch data

Success response data (json objects list):

```
[{"server_info":"cloudlinux-release-5 (2.6.18-294.26.1.el5.lve0.8.18)","ip":  
"1.1.1.1", "type": 1, "registered": false, "created": "2014-04-30T11:26-0400",  
"last_checkin": "2014-04-30T11:26-0400"}, ...]
```

- **server_info(string)** - info about os, version and running kernel of server, registered by the IP license available)
- **ip(string)**
- **type(int)** - license type (1,16,41,42,43,49)
- **registered(boolean)** - true if server was registered in CLN with this license (CLN licenses only).
- **created(string)** - license creation time
- **last_checkin(string)** - license last_checkin time

/update.json

URL example:

https://portal.tuxcare.com/api/ip/update.json?ip=1.1.1.1&type=16&token=AUTH_TOKEN

Update the IP-based license from the authorized user licenses.

GET arguments:

- **token** - authorization token
- **ip** - IP address to fetch data
- **els_allowed** - allow CL6 registration
- **type (optional)** - IP license type. If empty - will remove licenses with all types

Success response data (boolean):

- **true** - when IP license was updated
- **false** - if IP license was not found OR not owned by user

KernelCare API

API root URL: <https://portal.tuxcare.com/api/kcare>

You can manage KC licenses and keys over this API.

Create License Key /key/create.json

URL: [https://portal.tuxcare.com/api/kcare/key/create.json?
token=AUTH_TOKEN&limit=2¬e=Key+description](https://portal.tuxcare.com/api/kcare/key/create.json?token=AUTH_TOKEN&limit=2¬e=Key+description)

Will generate a new KC key for an authorized user.

GET arguments:

- **token** - authorization token
- **limit** - key servers limit (0 for unlimited key)
- **note** (optional) - key description up to 100 characters

Success response data (string): returns newly generated KC key

Delete License Key /key/delete.json

URL:

https://portal.tuxcare.com/api/kcare/key/delete.json?token=AUTH-TOKEN&key=key_to_delete

Will delete KC key owned by the authorized user.

GET arguments:

- **token** - authorization token
- **key** - KC key to delete

Success response data (boolean): returns true if the key was deleted or false if the key was not found.

List License Key /key/list.json

URL: https://portal.tuxcare.com/api/kcare/key/list.json?token=AUTH_TOKEN

Return list of all KC keys registered by an authorized user.

GET arguments:

- **token** - authorization token

Success response data (json objects list):

```
[{
  "key": "WsTs821nSAtiastD", // key identifier "enabled": false,
  "created": "2014-04-30T11:26-0400", // key creation time "limit": 2,
  // key servers max limit 0 for unlimited key "note": "Some custom
  keynote"
}, // Will return an empty list if a user has no keys
]
```

/key/servers.json

URL:

<https://portal.tuxcare.com/api/kcare/key/servers.json?token=AUTH-TOKEN&key=CEGaUo vC9Bi8ppH9>

Return list of servers registered with key owned by authorized user.

GET arguments:

- **token** - authorization token
- **key** - KC key linked to servers

Add CIDR List to KC Key /key/set_cidr.json

URL:

https://portal.tuxcare.com/api/kcare/key/set_cidr.json?token=AUTH_TOKEN&key=KEY&cidr=cidr+list

Will add a list of CIDRs for the authorized user.

GET arguments:

- **token** - authorization token
- **key** - key associated with CIDR
- **cidr** - allowed IP range in CIDR format. Multiple CIDRs can be separated with spaces, commas and sem

Success response data (json object):

```
{
  "code": 0 // CIDR adding status code
}
```

- code(int):
 - 0 - success (all CIDRs have been added)
 - 1 - irregular IP address in one of CIDRs
 - 2 - irregular prefix in one of CIDRs
 - 3 - internal error

Remove Server Registered too Key /srv/remove.json

URL:

https://portal.tuxcare.com/api/kcare/srv/remove.json?server_id={value}&token={value}

GET arguments:

- **token** - authorization token
- **server_id** - server id to remove

Success response data (json): Server found and deleted

```
{
  "message": null,
  "type": null,
  "success": true,
  "data": {
    "server_id": "",
    "key": ""
  }
}
```

Success response data (json): Server not found

```
{
  "message": "Server not found", "type":
  null,
  "success": true,
  "data": null
}
```

Error response (json): authorization failed

```
{
  "message": "Authorization failed", "type":
  null,
  "success": false, "data":
  null
}
```

KernelCare Nagios API

/nagios/register_key.plain

URL: https://portal.tuxcare.com/api/kcare/nagios/register_key.plain?key=WsTs821nSAtiastD
API to register a custom monitoring key for the IP license. The key will be registered for the remote client IP.

GET arguments:

- **key** - Monitoring key from 16 to 32 alphanumeric characters (small & capital letters)

Success response example:

```
success:true code:0
```

Any nonzero code value means key creation error!

- **code(int):**
 - 0 - success (key created/updated)
 - 1 - wrong key size/for mat
 - 2 - no KC IP license available

Server Error example:

```
success:false
```

/nagios

URL: https://portal.tuxcare.com/api/kcare/nagios/{key_id}

Will check status all servers registered by the key

- **key_id** - Server id to check

Return success response if all servers are up to date.

Success response example:

```
KernelCare OK - all servers are up to date!
```

Error response will be returned if any server is out of date, unsupported or inactive.

/nagios-res

URL: <https://portal.tuxcare.com/api/kcare/nagios-res/{login}/{token}>

Will check the status of all servers registered to reseller

- **login** - Login of reseller
- **token** - authorization token

Will return success response if all servers are up to date.

Success response example:

```
KernelCare OK - all servers are up to date!
```

Error response will be returned if any server is out of date, unsupported or inactive.

Server Groups for KernelCare API

API root URL: <https://portal.tuxcare.com/api/group>

API to manage server groups for KernelCare products.

/create.json

METHOD: POST

URL:

https://portal.tuxcare.com/api/group/create.json?token=AUTH_TOKEN&name=NAME&description=DESCRIPTION

Will create a new group for the authorized user.

Query parameters:

- **token** - authorization token
- **name** - group name
- **description** (optional) – group description

Success response data (json): Returns newly generated group

```
{
  "message": null,
  "type": null,
  "success": true,
  "data": {
    "name": "NAME",
    "description": "DESCRIPTION"
  }
}
```

/list.json

METHOD: GET

URL: https://portal.tuxcare.com/api/group/list.json?token=AUTH_TOKEN

Will receive all groups for authorized user.

Query parameter:

- **token** - authorization token

Success response data (json): Returns all groups


```
{
  "message": null,
  "type": null,
  "success": true,
  "data": [
    {
      "name": "NAME",
      "description": "DESCRIPTION"
    }
  ]
}
```

/servers/list.json

METHOD: GET

URL:

https://portal.tuxcare.com/api/group/servers/list.json?token=AUTH_TOKEN&name=NAME

Will return a list of the server objects or an empty list.

Query parameters:

- **token** - authorization token
- **name** - group name

Success response data (json): Returns server objects

```
{
  "message": null,
  "type": null,
  "success": true,
  "data": [
    {
      "id": "SERVER-1",
      "hostName": "10.000.0.000",
      "ip": "10.000.0.000",
      "type": "IMUNIFY"
    }
  ]
}
```

/servers/put.json

METHOD: PUT

URL:

[https://portal.tuxcare.com/api/group/servers/put.json?token=AUTH_TOKEN&name=NAME
&server_ids=SERVER-1,SERVER-2](https://portal.tuxcare.com/api/group/servers/put.json?token=AUTH_TOKEN&name=NAME&server_ids=SERVER-1,SERVER-2)

Will add servers to a group.

Query parameters:

- **token** - authorization token
- **name** - group name
- **server_ids** – array of server ids

Success response data (json): Returns a list of added servers

```
{
  "message": null,
  "type": null,
  "success": true,
  "data": [
    {
      "id": "SERVER-1",
      "hostName": "10.000.0.000",
      "ip": "10.000.0.000",
      "type": "IMUNIFY"
    },
    {
      "id": "SERVER-1",
      "hostName": "10.000.0.000",
      "ip": "10.000.0.000",
      "type": "IMUNIFY"
    }
  ]
}
```

/servers/remove.json

METHOD: DELETE

URL:

https://portal.portal.tuxcare.com/api/group/servers/remove.json?token=AUTH_TOKEN&name=NAME&server_ids=SERVER-1,SERVER-2

Will remove servers from a group

Query parameters:

- **token** - authorization token
- **name** - group name
- **server_ids** – array of server ids

Success response data (json): Returns a list of removed servers

```
{
  "message": null,
  "type": null,
  "success": true,
  "data": [
    {
      "id": "SERVER-1",
      "hostName": "10.000.0.000",
      "ip": "10.000.0.000",
      "type": "IMUNIFY"
    },
    {
      "id": "SERVER-1",
      "hostName": "10.000.0.000",
      "ip": "10.000.0.000",
      "type": "IMUNIFY"
    }
  ]
}
```