PlanetLab Central API Documentation

PlanetLab Central API Documentation

Table of Contents

Chapter 1. Introduction

The PlanetLab Central API (PLCAPI) is the interface through which the PlanetLab Central database should be accessed and maintained. The API is used by the website, by nodes, by automated scripts, and by users to access and update information about users, nodes, sites, slices, and other entities maintained by the database.

1.1. Authentication

The API should be accessed via XML-RPC over HTTPS. The API supports the standard introspection calls system.listMethods, system.methodSignature, and system.methodHelp, and the standard batching call system.multicall. With the exception of these calls, all PLCAPI calls take an authentication structure as their first argument. All authentication structures require the specification of *`AuthMethod`*. If the documentation for a call does not further specify the authentication structure, then any of (but only) the following authentication structures may be used:

- Session authentication. User sessions are typically valid for 24 hours. Node sessions are valid until the next reboot. Obtain a session key with GetSession using another form of authentication, such as password or GnuPG authentication.

AuthMethod	`session`
session	Session key

- Password authentication.

AuthMethod	`password`
Username	Username, typically an e-mail address
AuthString	Authentication string, typically a password

- GnuPG authentication. Users may upload a GPG public key using AddPersonKey. Peer GPG keys should be added with AddPeer or UpdatePeer.

AuthMethod	`gpg`
name	Peer or user name

signature	GnuPG signature of the canonicalized (http://www.w3.org/TR/xml-c14n) XML-RPC (http://www.xmlrpc.com/spec) representation of the rest of the arguments to the call.

- Anonymous authentication.

AuthMethod	`anonymous`

1.2. Roles

Some functions may only be called by users with certain roles (see GetRoles), and others may return different information to different callers depending on the role(s) of the caller.

The `node` and `anonymous` roles are pseudo-roles. A function that allows the `node` role may be called by automated scripts running on a node, such as the Boot and Node Managers. A function that allows the `anonymous` role may be called by anyone; an API authentication structure must still be specified (see Section 1.1).

1.3. Filters

Most of the `Get` functions take a filter argument. Filters may be arrays of integer (and sometimes string) identifiers, or a struct representing a filter on the attributes of the entities being queried. For example,

```
# plcsh code fragment (see below)
GetNodes([1,2,3])
GetNodes({'node_id': [1,2,3]})
GetNodes({'node_id': 1}) + GetNodes({'node_id': 2}) + GetNodes({'node_id': 3})
```

Would all be equivalent queries. Attributes that are themselves arrays (such as `nodenetwork_ids` and `slice_ids` for nodes) cannot be used in filters.

1.4. PlanetLab shell

A command-line program called **plcsh** simplifies authentication structure handling, and is useful for scripting. This program is distributed as a Linux RPM called PLCAPI and requires Python ≥2.4.

```
usage: plcsh [options]

options:
  -f CONFIG, --config=CONFIG
                        PLC configuration file
  -h URL, --url=URL     API URL
  -c CACERT, --cacert=CACERT
                        API SSL certificate
  -k INSECURE, --insecure=INSECURE
                        Do not check SSL certificate
  -m METHOD, --method=METHOD
                        API authentication method
  -s SESSION, --session=SESSION
                        API session key
  -u USER, --user=USER  API user name
  -p PASSWORD, --password=PASSWORD
                        API password
  -r ROLE, --role=ROLE  API role
  -x, --xmlrpc          Use XML-RPC interface
  --help                show this help message and exit
```

Specify at least the API URL and your user name:

```
plcsh --url https://www.planet-lab.org/PLCAPI/ -u user@site.edu
```

You will be presented with a prompt. From here, you can invoke API calls and omit the authentication structure, as it will be filled in automatically.

```
user@site.edu connected using password authentication
Type "system.listMethods()" or "help(method)" for more information.
[user@site.edu]>>> AuthCheck()
1
[user@site.edu]>>> GetNodes([121], ['node_id', 'hostname'])
[{'node_id': 121, 'hostname': 'planetlab-1.cs.princeton.edu'}]
```

As this program is actually a Python interpreter, you may create variables, execute for loops, import other packages, etc., directly on the command line as you would using the regular Python shell.

To use **plcsh** programmatically, import the `PLC.Shell` module:

```
#!/usr/bin/python
```

```
import sys

# Default location that the PLCAPI RPM installs the PLC class
sys.path.append('/usr/share/plc_api')

# Initialize shell environment. Shell() will define all PLCAPI methods
# in the specified namespace (specifying globals() will define them
# globally).
from PLC.Shell import Shell
plc = Shell(globals(),
            url = "https://www.planet-lab.org/PLCAPI/",
            user = "user@site.edu",
            password = "password")

# Both are equivalent
nodes = GetNodes([121], ['node_id', 'hostname'])
nodes = plc.GetNodes([121], ['node_id', 'hostname'])
```

Chapter 2. PlanetLab API Methods

2.1. AddAddressType

Prototype:

AddAddressType (auth, address_type_fields)

Description:

Adds a new address type. Fields specified in address_type_fields are used.

Returns the new address_type_id (> 0) if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *address_type_fields* : struct
  - *name* : string, Address type
  - *description* : string, Address type description

Returns:

- int, New address_type_id (> 0) if successful

2.2. AddAddressTypeToAddress

Prototype:

AddAddressTypeToAddress (auth, address_type_id_or_name, address_id)

Description:

Adds an address type to the specified address.

PIs may only update addresses of their own sites.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *address_type_id_or_name* : int or string
  - int, Address type identifier
  - string, Address type
- *address_id* : int, Address identifier

Returns:

- int, 1 if successful

2.3. AddBootState

Prototype:

AddBootState (auth, name)

Description:

Adds a new node boot state.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *name* : string, Boot state

Returns:

- int, 1 if successful

2.4. AddConfFile

Prototype:

AddConfFile (auth, conf_file_fields)

Description:

Adds a new node configuration file. Any fields specified in conf_file_fields are used, otherwise defaults are used.

Returns the new conf_file_id (> 0) if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *conf_file_fields* : struct
  - *file_owner* : string, chown(1) owner
  - *postinstall_cmd* : string, Shell command to execute after installing
  - *error_cmd* : string, Shell command to execute if any error occurs
  - *preinstall_cmd* : string, Shell command to execute prior to installing

- *dest* : string, Absolute path where file should be installed
- *ignore_cmd_errors* : boolean, Install file anyway even if an error occurs
- *enabled* : boolean, Configuration file is active
- *file_permissions* : string, chmod(1) permissions
- *source* : string, Relative path on the boot server where file can be downloaded
- *always_update* : boolean, Always attempt to install file even if unchanged
- *file_group* : string, chgrp(1) owner

Returns:

- int, New conf_file_id (> 0) if successful

2.5. AddConfFileToNodeGroup

Prototype:

AddConfFileToNodeGroup (auth, conf_file_id, nodegroup_id_or_name)

Description:

Adds a configuration file to the specified node group. If the node group is already linked to the configuration file, no errors are returned.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *conf_file_id* : int, Configuration file identifier
- *nodegroup_id_or_name* : int or string
  - int, Node group identifier
  - string, Node group name

Returns:

- int, 1 if successful

2.6. AddConfFileToNode

Prototype:

AddConfFileToNode (auth, conf_file_id, node_id_or_hostname)

Description:

Adds a configuration file to the specified node. If the node is already linked to the configuration file, no errors are returned.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *conf_file_id* : int, Configuration file identifier
- *node_id_or_hostname* : int or string
  - int, Node identifier
  - string, Fully qualified hostname

Returns:

- int, 1 if successful

2.7. AddKeyType

Prototype:

AddKeyType (auth, name)

Description:

Adds a new key type.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *name* : string, Key type

Returns:

- int, 1 if successful

2.8. AddMessage

Prototype:

AddMessage (auth, message_fields)

Description:

Adds a new message template. Any values specified in message_fields are used, otherwise defaults are used.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *message_fields* : struct
  - *enabled* : boolean, Message is enabled
  - *message_id* : string, Message identifier
  - *template* : string, Message template
  - *subject* : string, Message summary

Returns:

- int, 1 if successful

2.9. AddNetworkMethod

Prototype:

AddNetworkMethod (auth, name)

Description:

Adds a new network method.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *name* : string, Network method

Returns:

- int, 1 if successful

2.10. AddNetworkType

Prototype:

AddNetworkType (auth, name)

Description:

Adds a new network type.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *name* : string, Network type

Returns:

- int, 1 if successful

2.11. AddNodeGroup

Prototype:

AddNodeGroup (auth, nodegroup_fields)

Description:

Adds a new node group. Any values specified in nodegroup_fields are used, otherwise defaults are used.

Returns the new nodegroup_id (> 0) if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *nodegroup_fields* : struct
  - *name* : string, Node group name
  - *description* : string, Node group description

Returns:

- int, New nodegroup_id (> 0) if successful

2.12. AddNodeNetwork

Prototype:

AddNodeNetwork (auth, node_id_or_hostname, nodenetwork_fields)

Description:

Adds a new network for a node. Any values specified in nodenetwork_fields are used, otherwise defaults are used. Acceptable values for method may be retrieved via GetNetworkMethods. Acceptable values for type may be retrieved via GetNetworkTypes.

If type is static, ip, gateway, network, broadcast, netmask, and dns1 must all be specified in nodenetwork_fields. If type is dhcp, these parameters, even if specified, are ignored.

PIs and techs may only add networks to their own nodes. Admins may add networks to any node.

Returns the new nodenetwork_id (> 0) if successful, faults otherwise.

Allowed Roles:

admin, pi, tech

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *node_id_or_hostname* : int or string
  - int, Node identifier
  - string, Fully qualified hostname

- *nodenetwork_fields* : struct
  - *network* : string, Subnet address
  - *is_primary* : boolean, Is the primary interface for this node
  - *dns1* : string, IP address of primary DNS server
  - *hostname* : string, (Optional) Hostname
  - *mac* : string, MAC address
  - *bwlimit* : int, Bandwidth limit
  - *broadcast* : string, Network broadcast address
  - *method* : string, Addressing method (e.g., 'static' or 'dhcp')
  - *netmask* : string, Subnet mask
  - *dns2* : string, IP address of secondary DNS server
  - *ip* : string, IP address
  - *type* : string, Address type (e.g., 'ipv4')
  - *gateway* : string, IP address of primary gateway

Returns:

- int, New nodenetwork_id (> 0) if successful

2.13. AddNode

Prototype:

AddNode (auth, site_id_or_login_base, node_fields)

Description:

Adds a new node. Any values specified in node_fields are used, otherwise defaults are used.

PIs and techs may only add nodes to their own sites. Admins may add nodes to any site.

Returns the new node_id (> 0) if successful, faults otherwise.

Allowed Roles:

admin, pi, tech

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *site_id_or_login_base* : int or string
  - int, Site identifier
  - string, Site slice prefix
- *node_fields* : struct
  - *boot_state* : string, Boot state
  - *model* : string, Make and model of the actual machine
  - *version* : string, Apparent Boot CD version
  - *hostname* : string, Fully qualified hostname

Returns:

- int, New node_id (> 0) if successful

2.14. AddNodeToNodeGroup

Prototype:

AddNodeToNodeGroup (auth, node_id_or_hostname, nodegroup_id_or_name)

Description:

Add a node to the specified node group. If the node is already a member of the nodegroup, no errors are returned.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *node_id_or_hostname* : int or string
  - int, Node identifier
  - string, Fully qualified hostname

- *nodegroup_id_or_name* : int or string
  - int, Node group identifier
  - string, Node group name

Returns:

- int, 1 if successful

2.15. AddNodeToPCU

Prototype:

AddNodeToPCU (auth, node_id_or_hostname, pcu_id, port)

Description:

Adds a node to a port on a PCU. Faults if the node has already been added to the PCU or if the port is already in use.

Non-admins may only update PCUs at their sites.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, tech

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *node_id_or_hostname* : int or string
  - int, Node identifier
  - string, Fully qualified hostname

- *pcu_id* : int, PCU identifier
- *port* : int, PCU port number

Returns:

- int, 1 if successful

2.16. AddPCU

Prototype:

AddPCU (auth, site_id_or_login_base, pcu_fields)

Description:

Adds a new power control unit (PCU) to the specified site. Any fields specified in pcu_fields are used, otherwise defaults are used.

PIs and technical contacts may only add PCUs to their own sites.

Returns the new pcu_id (> 0) if successful, faults otherwise.

Allowed Roles:

admin, pi, tech

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *site_id_or_login_base* : int or string
  - int, Site identifier
  - string, Site slice prefix

- *pcu_fields* : struct
  - *username* : string, PCU username
  - *protocol* : string, PCU protocol, e.g. ssh, https, telnet
  - *ip* : string, PCU IP address
  - *notes* : string, Miscellaneous notes
  - *hostname* : string, PCU hostname
  - *model* : string, PCU model string
  - *password* : string, PCU username

Returns:

- int, New pcu_id (> 0) if successful

2.17. AddPeer

Prototype:

AddPeer (auth, peer_fields)

Description:

Adds a new peer.

Returns the new peer_id (> 0) if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *peer_fields* : struct
  - *key* : string, Peer GPG public key
  - *cacert* : string, Peer SSL public certificate
  - *peername* : string, Peer name
  - *peer_url* : string, Peer API URL

Returns:

- int, New peer_id (> 0) if successful

2.18. AddPersonKey

Prototype:

AddPersonKey (auth, person_id_or_email, key_fields)

Description:

Adds a new key to the specified account.

Non-admins can only modify their own keys.

Returns the new key_id (> 0) if successful, faults otherwise.

Allowed Roles:

admin, pi, tech, user

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *person_id_or_email* : int or string
  - int, User identifier
  - string, Primary e-mail address

- *key_fields* : struct
  - *key_type* : string, Key type
  - *key* : string, Key value

Returns:

- int, New key_id (> 0) if successful

2.19. AddPerson

Prototype:

AddPerson (auth, person_fields)

Description:

Adds a new account. Any fields specified in person_fields are used, otherwise defaults are used.

Accounts are disabled by default. To enable an account, use UpdatePerson().

Returns the new person_id (> 0) if successful, faults otherwise.

Allowed Roles:

admin, pi

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *person_fields* : struct
  - *bio* : string, Biography
  - *first_name* : string, Given name
  - *last_name* : string, Surname
  - *title* : string, Title
  - *url* : string, Home page
  - *phone* : string, Telephone number

- *password* : string, Account password in crypt() form
- *email* : string, Primary e-mail address

Returns:

- int, New person_id (> 0) if successful

2.20. AddPersonToSite

Prototype:

AddPersonToSite (auth, person_id_or_email, site_id_or_login_base)

Description:

Adds the specified person to the specified site. If the person is already a member of the site, no errors are returned. Does not change the person's primary site.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *person_id_or_email* : int or string
  - int, User identifier
  - string, Primary e-mail address
- *site_id_or_login_base* : int or string
  - int, Site identifier
  - string, Site slice prefix

Returns:

- int, 1 if successful

2.21. AddPersonToSlice

Prototype:

AddPersonToSlice (auth, person_id_or_email, slice_id_or_name)

Description:

Adds the specified person to the specified slice. If the person is already a member of the slice, no errors are returned.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *person_id_or_email* : int or string
  - int, User identifier
  - string, Primary e-mail address
- *slice_id_or_name* : int or string
  - int, Slice identifier
  - string, Slice name

Returns:

- int, 1 if successful

2.22. AddRole

Prototype:

AddRole (auth, role_id, name)

Description:

Adds a new role.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *role_id* : int, Role identifier
- *name* : string, Role

Returns:

- int, 1 if successful

2.23. AddRoleToPerson

Prototype:

AddRoleToPerson (auth, role_id_or_name, person_id_or_email)

Description:

Grants the specified role to the person.

PIs can only grant the tech and user roles to users and techs at their sites. Admins can grant any role to any user.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *role_id_or_name* : int or string
  - int, Role identifier
  - string, Role

- *person_id_or_email* : int or string
  - int, User identifier
  - string, Primary e-mail address

Returns:

- int, 1 if successful

2.24. AddSiteAddress

Prototype:

AddSiteAddress (auth, site_id_or_login_base, address_fields)

Description:

Adds a new address to a site. Fields specified in address_fields are used; some are not optional.

PIs may only add addresses to their own sites.

Returns the new address_id (> 0) if successful, faults otherwise.

Allowed Roles:

admin, pi

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *site_id_or_login_base* : int or string
  - int, Site identifier
  - string, Site slice prefix

- *address_fields* : struct
  - *city* : string, City
  - *country* : string, Country
  - *line3* : string, Address line 3
  - *line2* : string, Address line 2
  - *line1* : string, Address line 1
  - *state* : string, State or province
  - *postalcode* : string, Postal code

Returns:

- int, New address_id (> 0) if successful

2.25. AddSite

Prototype:

AddSite (auth, site_fields)

Description:

Adds a new site, and creates a node group for that site. Any fields specified in site_fields are used, otherwise defaults are used.

Returns the new site_id (> 0) if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *site_fields* : struct
  - *name* : string, Full site name
  - *url* : string, URL of a page that describes the site
  - *enabled* : boolean, Has been enabled
  - *longitude* : double, Decimal longitude of the site
  - *latitude* : double, Decimal latitude of the site
  - *max_slices* : int, Maximum number of slices that the site is able to create
  - *login_base* : string, Site slice prefix
  - *max_slivers* : int, Maximum number of slivers that the site is able to create
  - *is_public* : boolean, Publicly viewable site
  - *abbreviated_name* : string, Abbreviated site name

Returns:

- int, New site_id (> 0) if successful

2.26. AddSliceAttribute

Prototype:

AddSliceAttribute (auth, slice_id_or_name, attribute_type_id_or_name, value, node_id_or_hostname)

Description:

Sets the specified attribute of the slice (or sliver, if node_id_or_hostname is specified) to the specified value.

Attributes may require the caller to have a particular role in order to be set or changed. Users may only set attributes of slices or slivers of which they are members. PIs may only set attributes of slices or slivers at their sites, or of which they are members. Admins may set attributes of any slice or sliver.

Returns the new slice_attribute_id (> 0) if successful, faults otherwise.

Allowed Roles:

admin, pi, user

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *slice_id_or_name* : int or string
  - int, Slice identifier
  - string, Slice attribute type name

- *attribute_type_id_or_name* : int or string
  - int, Slice attribute type identifier
  - string, Slice attribute type name

- *value* : string, Slice attribute value
- *node_id_or_hostname* : int or string
  - int, Node identifier
  - string, Fully qualified hostname

Returns:

- int, New slice_attribute_id (> 0) if successful

2.27. AddSliceAttributeType

Prototype:

AddSliceAttributeType (auth, attribute_type_fields)

Description:

Adds a new type of slice attribute. Any fields specified in attribute_type_fields are used, otherwise defaults are used.

Returns the new attribute_type_id (> 0) if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *attribute_type_fields* : struct
  - *min_role_id* : int, Minimum (least powerful) role that can set or change this attribute
  - *name* : string, Slice attribute type name
  - *description* : string, Slice attribute type description

Returns:

- int, New attribute_id (> 0) if successful

2.28. AddSliceInstantiation

Prototype:

AddSliceInstantiation (auth, name)

Description:

Adds a new slice instantiation state.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *name* : string, Slice instantiation state

Returns:

- int, 1 if successful

2.29. AddSlice

Prototype:

AddSlice (auth, slice_fields)

Description:

Adds a new slice. Any fields specified in slice_fields are used, otherwise defaults are used.

Valid slice names are lowercase and begin with the login_base (slice prefix) of a valid site, followed by a single underscore. Thereafter, only letters, numbers, or additional underscores may be used.

PIs may only add slices associated with their own sites (i.e., slice prefixes must always be the login_base of one of their sites).

Returns the new slice_id (> 0) if successful, faults otherwise.

Allowed Roles:

admin, pi

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *slice_fields* : struct
  - *url* : string, URL further describing this slice
  - *max_nodes* : int, Maximum number of nodes that can be assigned to this slice
  - *instantiation* : string, Slice instantiation state
  - *name* : string, Slice name
  - *description* : string, Slice description

Returns:

- int, New slice_id (> 0) if successful

2.30. AddSliceToNodes

Prototype:

AddSliceToNodes (auth, slice_id_or_name, node_id_or_hostname_list)

Description:

Adds the specified slice to the specified nodes. Nodes may be either local or foreign nodes.

If the slice is already associated with a node, no errors are returned.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, user

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *slice_id_or_name* : int or string
  - int, Slice identifier
  - string, Slice name

- *node_id_or_hostname_list* : array of int or string
  - int, Node identifier
  - string, Fully qualified hostname

Returns:

- int, 1 if successful

2.31. AuthCheck

Prototype:

AuthCheck (auth)

Description:

Returns 1 if the user or node authenticated successfully, faults otherwise.

Allowed Roles:

admin, pi, user, tech, node

Parameters:

- `auth` : struct, API authentication structure
  - `AuthMethod` : string, Authentication method to use

Returns:

- int, 1 if successful

2.32. BlacklistKey

Prototype:

BlacklistKey (auth, key_id)

Description:

Blacklists a key, disassociating it and all others identical to it from all accounts and preventing it from ever being added again.

WARNING: Identical keys associated with other accounts with also be blacklisted.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *key_id* : int, Key identifier

Returns:

- int, 1 if successful

2.33. BootGetNodeDetails

Prototype:

BootGetNodeDetails (auth)

Description:

Returns a set of details about the calling node, including a new node session value.

Allowed Roles:

node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use, always 'hmac'
  - *value* : string, HMAC of node key and method call
  - *node_id* : int, Node identifier

Returns:

- struct
  - *boot_state* : string, Boot state
  - *model* : string, Make and model of the actual machine
  - *hostname* : string, Fully qualified hostname
  - *networks* : array of struct
    - *broadcast* : string, Network broadcast address

- *is_primary* : boolean, Is the primary interface for this node
- *network* : string, Subnet address
- *ip* : string, IP address
- *dns1* : string, IP address of primary DNS server
- *hostname* : string, (Optional) Hostname
- *netmask* : string, Subnet mask
- *gateway* : string, IP address of primary gateway
- *nodenetwork_id* : int, Node interface identifier
- *mac* : string, MAC address
- *node_id* : int, Node associated with this interface
- *dns2* : string, IP address of secondary DNS server
- *bwlimit* : int, Bandwidth limit
- *type* : string, Address type (e.g., 'ipv4')
- *method* : string, Addressing method (e.g., 'static' or 'dhcp')

- *session* : string, Session key

2.34. BootNotifyOwners

Prototype:

BootNotifyOwners (auth, message_id, include_pis, include_techs, include_support)

Description:

Notify the owners of the node, and/or support about an event that happened on the machine.

Returns 1 if successful.

Allowed Roles:

node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use, always 'hmac'

- *value* : string, HMAC of node key and method call
- *node_id* : int, Node identifier

- *message_id* : string, Message identifier
- *include_pis* : int, Notify PIs
- *include_techs* : int, Notify technical contacts
- *include_support* : int, Notify support

Returns:

- int, 1 if successful

2.35. BootUpdateNode

Prototype:

BootUpdateNode (auth, node_fields)

Description:

Allows the calling node to update its own record. Only the primary network can be updated, and the node IP cannot be changed.

Returns 1 if updated successfully.

Allowed Roles:

node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use, always 'hmac'
  - *value* : string, HMAC of node key and method call
  - *node_id* : int, Node identifier

- *node_fields* : struct
  - *boot_state* : string, Boot state
  - *primary_network* : struct

- *network* : string, Subnet address
- *dns2* : string, IP address of secondary DNS server
- *dns1* : string, IP address of primary DNS server
- *netmask* : string, Subnet mask
- *method* : string, Addressing method (e.g., 'static' or 'dhcp')
- *broadcast* : string, Network broadcast address
- *mac* : string, MAC address
- *gateway* : string, IP address of primary gateway

- *ssh_host_key* : string, Last known SSH host key

Returns:

- int, 1 if successful

2.36. DeleteAddress

Prototype:

DeleteAddress (auth, address_id)

Description:

Deletes an address.

PIs may only delete addresses from their own sites.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *address_id* : int, Address identifier

Returns:

- int, 1 if successful

2.37. DeleteAddressTypeFromAddress

Prototype:

DeleteAddressTypeFromAddress (auth, address_type_id_or_name, address_id)

Description:

Deletes an address type from the specified address.

PIs may only update addresses of their own sites.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *address_type_id_or_name* : int or string
  - int, Address type identifier
  - string, Address type

- *address_id* : int, Address identifier

Returns:

- int, 1 if successful

2.38. DeleteAddressType

Prototype:

DeleteAddressType (auth, address_type_id_or_name)

Description:

Deletes an address type.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *address_type_id_or_name* : int or string
  - int, Address type identifier
  - string, Address type

Returns:

- int, 1 if successful

2.39. DeleteBootState

Prototype:

DeleteBootState (auth, name)

Description:

Deletes a node boot state.

WARNING: This will cause the deletion of all nodes in this boot state.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *name* : string, Boot state

Returns:

- int, 1 if successful

2.40. DeleteConfFileFromNodeGroup

Prototype:

DeleteConfFileFromNodeGroup (auth, conf_file_id, nodegroup_id_or_name)

Description:

Deletes a configuration file from the specified nodegroup. If the nodegroup is not linked to the configuration file, no errors are returned.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *conf_file_id* : int, Configuration file identifier

- *nodegroup_id_or_name* : int or string
  - int, Node group identifier
  - string, Node group name

Returns:

- int, 1 if successful

2.41. DeleteConfFileFromNode

Prototype:

DeleteConfFileFromNode (auth, conf_file_id, node_id_or_hostname)

Description:

Deletes a configuration file from the specified node. If the node is not linked to the configuration file, no errors are returned.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *conf_file_id* : int, Configuration file identifier
- *node_id_or_hostname* : int or string
  - int, Node identifier
  - string, Fully qualified hostname

Returns:

- int, 1 if successful

2.42. DeleteConfFile

Prototype:

DeleteConfFile (auth, conf_file_id)

Description:

Returns an array of structs containing details about node configuration files. If conf_file_ids is specified, only the specified configuration files will be queried.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *conf_file_id* : int, Configuration file identifier

Returns:

- int, 1 if successful

2.43. DeleteKey

Prototype:

DeleteKey (auth, key_id)

Description:

Deletes a key.

Non-admins may only delete their own keys.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, tech, user

Parameters:

- `auth` : struct, API authentication structure
  - `AuthMethod` : string, Authentication method to use
- `key_id` : int, Key identifier

Returns:

- int, 1 if successful

2.44. DeleteKeyType

Prototype:

DeleteKeyType (auth, name)

Description:

Deletes a key type.

WARNING: This will cause the deletion of all keys of this type.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- `auth` : struct, API authentication structure
  - `AuthMethod` : string, Authentication method to use
- `name` : string, Key type

Returns:

- int, 1 if successful

2.45. DeleteMessage

Prototype:

DeleteMessage (auth, message_id)

Description:

Deletes a message template.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *message_id* : string, Message identifier

Returns:

- int, 1 if successful

2.46. DeleteNetworkMethod

Prototype:

DeleteNetworkMethod (auth, name)

Description:

Deletes a network method.

WARNING: This will cause the deletion of all network interfaces that use this method.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *name* : string, Network method

Returns:

- int, 1 if successful

2.47. DeleteNetworkType

Prototype:

DeleteNetworkType (auth, name)

Description:

Deletes a network type.

WARNING: This will cause the deletion of all network interfaces that use this type.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *name* : string, Network type

Returns:

- int, 1 if successful

2.48. DeleteNodeFromNodeGroup

Prototype:

DeleteNodeFromNodeGroup (auth, node_id_or_hostname, nodegroup_id_or_name)

Description:

Removes a node from the specified node group.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *node_id_or_hostname* : int or string
  - int, Node identifier
  - string, Fully qualified hostname
- *nodegroup_id_or_name* : int or string
  - int, Node group identifier
  - string, Node group name

Returns:

- int, 1 if successful

2.49. DeleteNodeFromPCU

Prototype:

DeleteNodeFromPCU (auth, node_id_or_hostname, pcu_id)

Description:

Deletes a node from a PCU.

Non-admins may only update PCUs at their sites.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, tech

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *node_id_or_hostname* : int or string
  - int, Node identifier
  - string, Fully qualified hostname
- *pcu_id* : int, PCU identifier

Returns:

- int, 1 if successful

2.50. DeleteNodeGroup

Prototype:

DeleteNodeGroup (auth, node_group_id_or_name)

Description:

Delete an existing Node Group.

ins may delete any node group

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *node_group_id_or_name* : int or string
  - int, Node group identifier
  - string, Node group name

Returns:

- int, 1 if successful

2.51. DeleteNodeNetwork

Prototype:

DeleteNodeNetwork (auth, nodenetwork_id)

Description:

Deletes an existing node network interface.

Admins may delete any node network. PIs and techs may only delete node network interfaces associated with nodes at their sites.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, tech

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *nodenetwork_id* : int, Node interface identifier

Returns:

- int, 1 if successful

2.52. DeleteNode

Prototype:

DeleteNode (auth, node_id_or_hostname)

Description:

Mark an existing node as deleted.

PIs and techs may only delete nodes at their own sites. ins may delete nodes at any site.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, tech

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *node_id_or_hostname* : int or string
  - int, Node identifier
  - string, Fully qualified hostname

Returns:

- int, 1 if successful

2.53. DeletePCU

Prototype:

DeletePCU (auth, pcu_id)

Description:

Deletes a PCU.

Non-admins may only delete PCUs at their sites.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, tech

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *pcu_id* : int, PCU identifier

Returns:

- int, 1 if successful

2.54. DeletePeer

Prototype:

DeletePeer (auth, peer_id_or_name)

Description:

Mark an existing peer as deleted. All entities (e.g., slices, keys, nodes, etc.) for which this peer is authoritative will also be deleted or marked as deleted.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *peer_id_or_name* : int or string
  - int, Peer identifier
  - string, Peer name

Returns:

- int, 1 if successful

2.55. DeletePersonFromSite

Prototype:

DeletePersonFromSite (auth, person_id_or_email, site_id_or_login_base)

Description:

Removes the specified person from the specified site. If the person is not a member of the specified site, no error is returned.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure

- *AuthMethod* : string, Authentication method to use

- *person_id_or_email* : int or string
  - int, User identifier
  - string, Primary e-mail address

- *site_id_or_login_base* : int or string
  - int, Site identifier
  - string, Site slice prefix

Returns:

- int, 1 if successful

2.56. DeletePersonFromSlice

Prototype:

DeletePersonFromSlice (auth, person_id_or_email, slice_id_or_name)

Description:

Deletes the specified person from the specified slice. If the person is not a member of the slice, no errors are returned.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *person_id_or_email* : int or string
  - int, User identifier

- string, Primary e-mail address

- *slice_id_or_name* : int or string
  - int, Slice identifier
  - string, Slice name

Returns:

- int, 1 if successful

2.57. DeletePerson

Prototype:

DeletePerson (auth, person_id_or_email)

Description:

Mark an existing account as deleted.

Users and techs can only delete themselves. PIs can only delete themselves and other non-PIs at their sites. ins can delete anyone.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, user, tech

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *person_id_or_email* : int or string
  - int, User identifier
  - string, Primary e-mail address

Returns:

- int, 1 if successful

2.58. DeleteRoleFromPerson

Prototype:

DeleteRoleFromPerson (auth, role_id_or_name, person_id_or_email)

Description:

Deletes the specified role from the person.

PIs can only revoke the tech and user roles from users and techs at their sites. ins can revoke any role from any user.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *role_id_or_name* : int or string
  - int, Role identifier
  - string, Role

- *person_id_or_email* : int or string
  - int, User identifier
  - string, Primary e-mail address

Returns:

- int, 1 if successful

2.59. DeleteRole

Prototype:

DeleteRole (auth, role_id_or_name)

Description:

Deletes a role.

WARNING: This will remove the specified role from all accounts that possess it, and from all node and slice attributes that refer to it.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *role_id_or_name* : int or string
  - int, Role identifier
  - string, Role

Returns:

- int, 1 if successful

2.60. DeleteSession

Prototype:

DeleteSession (auth)

Description:

Invalidates the current session.

Returns 1 if successful.

Allowed Roles:

admin, pi, user, tech, node

Parameters:

- *auth* : struct, API authentication structure
  - *session* : string, Session key
  - *AuthMethod* : string, Authentication method to use, always 'session'

Returns:

- int, 1 if successful

2.61. DeleteSite

Prototype:

DeleteSite (auth, site_id_or_login_base)

Description:

Mark an existing site as deleted. The accounts of people who are not members of at least one other non-deleted site will also be marked as deleted. Nodes, PCUs, and slices associated with the site will be deleted.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *site_id_or_login_base* : int or string

- int, Site identifier
- string, Site slice prefix

Returns:

- int, 1 if successful

2.62. DeleteSliceAttribute

Prototype:

DeleteSliceAttribute (auth, slice_attribute_id)

Description:

Deletes the specified slice or sliver attribute.

Attributes may require the caller to have a particular role in order to be deleted. Users may only delete attributes of slices or slivers of which they are members. PIs may only delete attributes of slices or slivers at their sites, or of which they are members. Admins may delete attributes of any slice or sliver.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, user

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *slice_attribute_id* : int, Slice attribute identifier

Returns:

- int, 1 if successful

2.63. DeleteSliceAttributeType

Prototype:

DeleteSliceAttributeType (auth, attribute_type_id_or_name)

Description:

Deletes the specified slice attribute.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *attribute_type_id_or_name* : int or string
  - int, Slice attribute type identifier
  - string, Slice attribute type name

Returns:

- int, 1 if successful

2.64. DeleteSliceFromNodes

Prototype:

DeleteSliceFromNodes (auth, slice_id_or_name, node_id_or_hostname_list)

Description:

Deletes the specified slice from the specified nodes. If the slice is not associated with a node, no errors are returned.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, user

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *slice_id_or_name* : int or string
  - int, Slice identifier
  - string, Slice name

- *node_id_or_hostname_list* : array of int or string
  - int, Node identifier
  - string, Fully qualified hostname

Returns:

- int, 1 if successful

2.65. DeleteSliceInstantiation

Prototype:

DeleteSliceInstantiation (auth, instantiation)

Description:

Deletes a slice instantiation state.

WARNING: This will cause the deletion of all slices of this instantiation.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *instantiation* : string, Slice instantiation state

Returns:

- int, 1 if successful

2.66. DeleteSlice

Prototype:

DeleteSlice (auth, slice_id_or_name)

Description:

Deletes the specified slice.

Users may only delete slices of which they are members. PIs may delete any of the slices at their sites, or any slices of which they are members. Admins may delete any slice.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, user

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *slice_id_or_name* : int or string
  - int, Slice identifier
  - string, Slice name

Returns:

- int, 1 if successful

2.67. GetAddresses

Prototype:

GetAddresses (auth, address_filter, return_fields)

Description:

Returns an array of structs containing details about addresses. If address_filter is specified and is an array of address identifiers, or a struct of address attributes, only addresses matching the filter will be returned. If return_fields is specified, only the specified details will be returned.

Allowed Roles:

admin, pi, user, tech, node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *address_filter* : array of int or struct
  - array of int, Address identifier
  - struct, Attribute filter
    - *city* : string or array of string
      - string, City
      - array of string, City

    - *address_id* : int or array of int
      - int, Address identifier
      - array of int, Address identifier

    - *country* : string or array of string
      - string, Country
      - array of string, Country

    - *line3* : string or array of string
        - string, Address line 3
        - array of string, Address line 3

    - *line2* : string or array of string
        - string, Address line 2
        - array of string, Address line 2

    - *line1* : string or array of string
        - string, Address line 1
        - array of string, Address line 1

    - *state* : string or array of string
        - string, State or province
        - array of string, State or province

    - *postalcode* : string or array of string
        - string, Postal code
        - array of string, Postal code

- *return_fields* : array, List of fields to return
    - string

Returns:

- array of struct
    - *city* : string, City
    - *address_id* : int, Address identifier
    - *country* : string, Country
    - *line3* : string, Address line 3
    - *line2* : string, Address line 2
    - *line1* : string, Address line 1

- *address_type_ids* : array, Address type identifiers
  - int

- *state* : string, State or province
- *postalcode* : string, Postal code
- *address_types* : array, Address types
  - string

2.68. GetAddressTypes

Prototype:

GetAddressTypes (auth, address_type_filter, return_fields)

Description:

Returns an array of structs containing details about address types. If address_type_filter is specified and is an array of address type identifiers, or a struct of address type attributes, only address types matching the filter will be returned. If return_fields is specified, only the specified details will be returned.

Allowed Roles:

admin, pi, user, tech, node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *address_type_filter* : array of int or string or struct
  - array of int or string
    - int, Address type identifier
    - string, Address type

  - struct, Attribute filter
    - *name* : string or array of string

    - string, Address type
    - array of string, Address type

  - *address_type_id* : int or array of int
    - int, Address type identifier
    - array of int, Address type identifier

  - *description* : string or array of string
    - string, Address type description
    - array of string, Address type description

- *return_fields* : array, List of fields to return
  - string

Returns:

- array of struct
  - *name* : string, Address type
  - *address_type_id* : int, Address type identifier
  - *description* : string, Address type description

2.69. GetBootStates

Prototype:

GetBootStates (auth)

Description:

Returns an array of all valid node boot states.

Allowed Roles:

admin, pi, user, tech, node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

Returns:

- array of string, Boot state

2.70. GetConfFiles

Prototype:

GetConfFiles (auth, conf_file_filter, return_fields)

Description:

Returns an array of structs containing details about configuration files. If conf_file_filter is specified and is an array of configuration file identifiers, or a struct of configuration file attributes, only configuration files matching the filter will be returned. If return_fields is specified, only the specified details will be returned.

Allowed Roles:

admin, node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *conf_file_filter* : array of int or struct
  - array of int, Configuration file identifier
  - struct, Attribute filter
    - *file_owner* : string or array of string
      - string, chown(1) owner
      - array of string, chown(1) owner

    - *postinstall_cmd* : string or array of string
      - string, Shell command to execute after installing

    - array of string, Shell command to execute after installing

- *error_cmd* : string or array of string
    - string, Shell command to execute if any error occurs
    - array of string, Shell command to execute if any error occurs

- *preinstall_cmd* : string or array of string
    - string, Shell command to execute prior to installing
    - array of string, Shell command to execute prior to installing

- *node_ids* : int or array of int
    - int, List of nodes linked to this file
    - array of int, List of nodes linked to this file

- *dest* : string or array of string
    - string, Absolute path where file should be installed
    - array of string, Absolute path where file should be installed

- *ignore_cmd_errors* : boolean or array of boolean
    - boolean, Install file anyway even if an error occurs
    - array of boolean, Install file anyway even if an error occurs

- *enabled* : boolean or array of boolean
    - boolean, Configuration file is active
    - array of boolean, Configuration file is active

- *conf_file_id* : int or array of int
    - int, Configuration file identifier
    - array of int, Configuration file identifier

- *file_permissions* : string or array of string
    - string, chmod(1) permissions
    - array of string, chmod(1) permissions

- *source* : string or array of string

    - string, Relative path on the boot server where file can be downloaded
    - array of string, Relative path on the boot server where file can be downloaded

  - *nodegroup_ids* : int or array of int
    - int, List of node groups linked to this file
    - array of int, List of node groups linked to this file

  - *always_update* : boolean or array of boolean
    - boolean, Always attempt to install file even if unchanged
    - array of boolean, Always attempt to install file even if unchanged

  - *file_group* : string or array of string
    - string, chgrp(1) owner
    - array of string, chgrp(1) owner

- *return_fields* : array, List of fields to return
  - string

Returns:

- array of struct
  - *file_owner* : string, chown(1) owner
  - *postinstall_cmd* : string, Shell command to execute after installing
  - *error_cmd* : string, Shell command to execute if any error occurs
  - *preinstall_cmd* : string, Shell command to execute prior to installing
  - *node_ids* : int, List of nodes linked to this file
  - *dest* : string, Absolute path where file should be installed
  - *ignore_cmd_errors* : boolean, Install file anyway even if an error occurs
  - *enabled* : boolean, Configuration file is active
  - *conf_file_id* : int, Configuration file identifier
  - *file_permissions* : string, chmod(1) permissions
  - *source* : string, Relative path on the boot server where file can be downloaded
  - *nodegroup_ids* : int, List of node groups linked to this file

- *always_update* : boolean, Always attempt to install file even if unchanged
- *file_group* : string, chgrp(1) owner

2.71. GetEvents

Prototype:

GetEvents (auth, event_filter, return_fields)

Description:

Returns an array of structs containing details about events and faults. If event_filter is specified and is an array of event identifiers, or a struct of event attributes, only events matching the filter will be returned. If return_fields is specified, only the specified details will be returned.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *event_filter* : array of int or struct
  - array of int, Event identifier
  - struct, Attribute filter
    - *fault_code* : int or array of int
      - int, Event fault code
      - array of int, Event fault code

    - *event_id* : int or array of int
      - int, Event identifier
      - array of int, Event identifier

    - *object_type* : string or array of string
      - string, What type of object is this event affecting

    - array of string, What type of object is this event affecting

  - *node_id* : int or array of int
    - int, Identifier of node responsible for event, if any
    - array of int, Identifier of node responsible for event, if any

  - *call* : string or array of string
    - string, Call responsible for this event, including paramters
    - array of string, Call responsible for this event, including paramters

  - *time* : int or array of int
    - int, Date and time that the event took place, in seconds since UNIX epoch
    - array of int, Date and time that the event took place, in seconds since UNIX epoch

  - *person_id* : int or array of int
    - int, Identifier of person responsible for event, if any
    - array of int, Identifier of person responsible for event, if any

  - *message* : string or array of string
    - string, High level description of this event
    - array of string, High level description of this event

  - *runtime* : double or array of double
    - double, Runtime of event
    - array of double, Runtime of event

  - *call_name* : string or array of string
    - string, Call responsible for this event
    - array of string, Call responsible for this event

- *return_fields* : array, List of fields to return
  - string

Returns:

- array of struct
  - *fault_code* : int, Event fault code
  - *event_id* : int, Event identifier
  - *object_type* : string, What type of object is this event affecting
  - *object_ids* : array, IDs of objects affected by this event
    - int
  - *node_id* : int, Identifier of node responsible for event, if any
  - *call* : string, Call responsible for this event, including paramters
  - *time* : int, Date and time that the event took place, in seconds since UNIX epoch
  - *person_id* : int, Identifier of person responsible for event, if any
  - *message* : string, High level description of this event
  - *runtime* : double, Runtime of event
  - *call_name* : string, Call responsible for this event

2.72. GetKeys

Prototype:

GetKeys (auth, key_filter, return_fields)

Description:

Returns an array of structs containing details about keys. If key_filter is specified and is an array of key identifiers, or a struct of key attributes, only keys matching the filter will be returned. If return_fields is specified, only the specified details will be returned.

Admin may query all keys. Non-admins may only query their own keys.

Allowed Roles:

admin, pi, user, tech, node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *key_filter* : array of int or struct
  - array of int
    - int, Key identifier
  - struct, Attribute filter
    - *peer_key_id* : int or array of int
      - int, Foreign key identifier at peer
      - array of int, Foreign key identifier at peer
    - *key_type* : string or array of string
      - string, Key type
      - array of string, Key type
    - *key* : string or array of string
      - string, Key value
      - array of string, Key value
    - *person_id* : int or array of int
      - int, User to which this key belongs
      - array of int, User to which this key belongs
    - *key_id* : int or array of int
      - int, Key identifier
      - array of int, Key identifier
    - *peer_id* : int or array of int
      - int, Peer to which this key belongs
      - array of int, Peer to which this key belongs
- *return_fields* : array, List of fields to return
  - string

Returns:

- array of struct
  - *peer_id* : int, Peer to which this key belongs
  - *key_type* : string, Key type
  - *key* : string, Key value
  - *person_id* : int, User to which this key belongs
  - *key_id* : int, Key identifier
  - *peer_key_id* : int, Foreign key identifier at peer

2.73. GetKeyTypes

Prototype:

GetKeyTypes (auth)

Description:

Returns an array of all valid key types.

Allowed Roles:

admin, pi, user, tech, node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

Returns:

- array of string, Key type

2.74. GetMessages

Prototype:

GetMessages (auth, message_filter, return_fields)

Description:

Returns an array of structs containing details about message templates. If message template_filter is specified and is an array of message template identifiers, or a struct of message template attributes, only message templates matching the filter will be returned. If return_fields is specified, only the specified details will be returned.

Allowed Roles:

admin, node

Parameters:

- `auth` : struct, API authentication structure
  - `AuthMethod` : string, Authentication method to use

- `message_filter` : array of string or struct
  - array of string, Message identifier
  - struct, Attribute filter
    - `enabled` : boolean or array of boolean
      - boolean, Message is enabled
      - array of boolean, Message is enabled

    - `message_id` : string or array of string
      - string, Message identifier
      - array of string, Message identifier

    - `template` : string or array of string
      - string, Message template
      - array of string, Message template

    - `subject` : string or array of string
      - string, Message summary
      - array of string, Message summary

- `return_fields` : array, List of fields to return

- string

Returns:

- array of struct
  - `enabled` : boolean, Message is enabled
  - `message_id` : string, Message identifier
  - `template` : string, Message template
  - `subject` : string, Message summary

2.75. GetNetworkMethods

Prototype:

GetNetworkMethods (auth)

Description:

Returns a list of all valid network methods.

Allowed Roles:

admin, pi, user, tech, node

Parameters:

- `auth` : struct, API authentication structure
  - `AuthMethod` : string, Authentication method to use

Returns:

- array of string, Network method

2.76. GetNetworkTypes

Prototype:

GetNetworkTypes (auth)

Description:

Returns a list of all valid network types.

Allowed Roles:

admin, pi, user, tech, node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

Returns:

- array of string, Network type

2.77. GetNodeGroups

Prototype:

GetNodeGroups (auth, nodegroup_filter, return_fields)

Description:

Returns an array of structs containing details about node groups. If nodegroup_filter is specified and is an array of node group identifiers or names, or a struct of node group attributes, only node groups matching the filter will be returned. If return_fields is specified, only the specified details will be returned.

Allowed Roles:

admin, pi, user, tech, node, anonymous

Parameters:

- *auth* : struct, API authentication structure

- *AuthMethod* : string, Authentication method to use

- *nodegroup_filter* : array of int or string or struct
  - array of int or string
    - int, Node group identifier
    - string, Node group name

  - struct, Attribute filter
    - *nodegroup_id* : int or array of int
      - int, Node group identifier
      - array of int, Node group identifier

    - *name* : string or array of string
      - string, Node group name
      - array of string, Node group name

    - *description* : string or array of string
      - string, Node group description
      - array of string, Node group description

- *return_fields* : array, List of fields to return
  - string

Returns:

- array of struct
  - *node_ids* : array, List of nodes in this node group
    - int

  - *nodegroup_id* : int, Node group identifier
  - *name* : string, Node group name
  - *conf_file_ids* : array, List of configuration files specific to this node group
    - int

- `description` : string, Node group description

2.78. GetNodeNetworks

Prototype:

GetNodeNetworks (auth, nodenetwork_filter, return_fields)

Description:

Returns an array of structs containing details about node network interfacess. If nodenetworks_filter is specified and is an array of node network identifiers, or a struct of node network attributes, only node network interfaces matching the filter will be returned. If return_fields is specified, only the specified details will be returned.

Allowed Roles:

admin, pi, user, tech, node, anonymous

Parameters:

- `auth` : struct, API authentication structure
  - `AuthMethod` : string, Authentication method to use

- `nodenetwork_filter` : array of int or struct
  - array of int, Node interface identifier
  - struct, Attribute filter
    - `nodenetwork_id` : int or array of int
      - int, Node interface identifier
      - array of int, Node interface identifier

    - `network` : string or array of string
      - string, Subnet address
      - array of string, Subnet address

    - `is_primary` : boolean or array of boolean
      - boolean, Is the primary interface for this node

    - array of boolean, Is the primary interface for this node

- *dns1* : string or array of string
    - string, IP address of primary DNS server
    - array of string, IP address of primary DNS server

- *hostname* : string or array of string
    - string, (Optional) Hostname
    - array of string, (Optional) Hostname

- *mac* : string or array of string
    - string, MAC address
    - array of string, MAC address

- *bwlimit* : int or array of int
    - int, Bandwidth limit
    - array of int, Bandwidth limit

- *broadcast* : string or array of string
    - string, Network broadcast address
    - array of string, Network broadcast address

- *method* : string or array of string
    - string, Addressing method (e.g., 'static' or 'dhcp')
    - array of string, Addressing method (e.g., 'static' or 'dhcp')

- *netmask* : string or array of string
    - string, Subnet mask
    - array of string, Subnet mask

- *node_id* : int or array of int
    - int, Node associated with this interface
    - array of int, Node associated with this interface

- *dns2* : string or array of string

    - string, IP address of secondary DNS server
    - array of string, IP address of secondary DNS server

  - *ip* : string or array of string
    - string, IP address
    - array of string, IP address

  - *type* : string or array of string
    - string, Address type (e.g., 'ipv4')
    - array of string, Address type (e.g., 'ipv4')

  - *gateway* : string or array of string
    - string, IP address of primary gateway
    - array of string, IP address of primary gateway

- *return_fields* : array, List of fields to return
  - string

Returns:

- array of struct
  - *broadcast* : string, Network broadcast address
  - *is_primary* : boolean, Is the primary interface for this node
  - *network* : string, Subnet address
  - *ip* : string, IP address
  - *dns1* : string, IP address of primary DNS server
  - *hostname* : string, (Optional) Hostname
  - *netmask* : string, Subnet mask
  - *gateway* : string, IP address of primary gateway
  - *nodenetwork_id* : int, Node interface identifier
  - *mac* : string, MAC address
  - *node_id* : int, Node associated with this interface
  - *dns2* : string, IP address of secondary DNS server

- *bwlimit* : int, Bandwidth limit
- *type* : string, Address type (e.g., 'ipv4')
- *method* : string, Addressing method (e.g., 'static' or 'dhcp')

2.79. GetNodes

Prototype:

GetNodes (auth, node_filter, return_fields)

Description:

Returns an array of structs containing details about nodes. If node_filter is specified and is an array of node identifiers or hostnames, or a struct of node attributes, only nodes matching the filter will be returned. If return_fields is specified, only the specified details will be returned.

Some fields may only be viewed by admins.

Allowed Roles:

admin, pi, user, tech, node, anonymous

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *node_filter* : array of int or string or struct
  - array of int or string
    - int, Node identifier
    - string, Fully qualified hostname

  - struct, Attribute filter
    - *last_updated* : int or array of int
      - int, Date and time when node entry was created
      - array of int, Date and time when node entry was created

    - *version* : string or array of string

  - string, Apparent Boot CD version
  - array of string, Apparent Boot CD version

- *boot_state* : string or array of string
  - string, Boot state
  - array of string, Boot state

- *peer_node_id* : int or array of int
  - int, Foreign node identifier at peer
  - array of int, Foreign node identifier at peer

- *hostname* : string or array of string
  - string, Fully qualified hostname
  - array of string, Fully qualified hostname

- *site_id* : int or array of int
  - int, Site at which this node is located
  - array of int, Site at which this node is located

- *boot_nonce* : string or array of string
  - string, (Admin only) Random value generated by the node at last boot
  - array of string, (Admin only) Random value generated by the node at last boot

- *session* : string or array of string
  - string, (Admin only) Node session value
  - array of string, (Admin only) Node session value

- *ssh_rsa_key* : string or array of string
  - string, Last known SSH host key
  - array of string, Last known SSH host key

- *key* : string or array of string
  - string, (Admin only) Node key
  - array of string, (Admin only) Node key

- *date_created* : int or array of int
  - int, Date and time when node entry was created
  - array of int, Date and time when node entry was created

- *model* : string or array of string
  - string, Make and model of the actual machine
  - array of string, Make and model of the actual machine

- *peer_id* : int or array of int
  - int, Peer to which this node belongs
  - array of int, Peer to which this node belongs

- *node_id* : int or array of int
  - int, Node identifier
  - array of int, Node identifier

- *return_fields* : array, List of fields to return
  - string

Returns:

- array of struct
  - *session* : string, (Admin only) Node session value
  - *pcu_ids* : array, List of PCUs that control this node
    - int

  - *nodegroup_ids* : array, List of node groups that this node is in
    - int

  - *last_updated* : int, Date and time when node entry was created
  - *version* : string, Apparent Boot CD version
  - *nodenetwork_ids* : array, List of network interfaces that this node has
    - int

- *boot_state* : string, Boot state
- *peer_node_id* : int, Foreign node identifier at peer
- *hostname* : string, Fully qualified hostname
- *site_id* : int, Site at which this node is located
- *ports* : array, List of PCU ports that this node is connected to
  - int
- *slice_ids* : array, List of slices on this node
  - int
- *boot_nonce* : string, (Admin only) Random value generated by the node at last boot
- *node_id* : int, Node identifier
- *key* : string, (Admin only) Node key
- *date_created* : int, Date and time when node entry was created
- *model* : string, Make and model of the actual machine
- *peer_id* : int, Peer to which this node belongs
- *conf_file_ids* : array, List of configuration files specific to this node
  - int
- *ssh_rsa_key* : string, Last known SSH host key

2.80. GetPCUs

Prototype:

GetPCUs (auth, pcu_filter, return_fields)

Description:

Returns an array of structs containing details about power control units (PCUs). If pcu_filter is specified and is an array of PCU identifiers, or a struct of PCU attributes, only PCUs matching the filter will be returned. If return_fields is specified, only the specified details will be returned.

Admin may query all PCUs. Non-admins may only query the PCUs at their sites.

Allowed Roles:

admin, pi, tech, node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *pcu_filter* : array of int or struct
  - array of int, PCU identifier
  - struct, Attribute filter
    - *username* : string or array of string
      - string, PCU username
      - array of string, PCU username

    - *protocol* : string or array of string
      - string, PCU protocol, e.g. ssh, https, telnet
      - array of string, PCU protocol, e.g. ssh, https, telnet

    - *ip* : string or array of string
      - string, PCU IP address
      - array of string, PCU IP address

    - *pcu_id* : int or array of int
      - int, PCU identifier
      - array of int, PCU identifier

    - *hostname* : string or array of string
      - string, PCU hostname
      - array of string, PCU hostname

    - *site_id* : int or array of int
      - int, Identifier of site where PCU is located
      - array of int, Identifier of site where PCU is located

- *model* : string or array of string
  - string, PCU model string
  - array of string, PCU model string

- *password* : string or array of string
  - string, PCU username
  - array of string, PCU username

- *notes* : string or array of string
  - string, Miscellaneous notes
  - array of string, Miscellaneous notes

- *return_fields* : array, List of fields to return
  - string

Returns:

- array of struct
  - *username* : string, PCU username
  - *protocol* : string, PCU protocol, e.g. ssh, https, telnet
  - *node_ids* : array, List of nodes that this PCU controls
    - int

  - *ip* : string, PCU IP address
  - *pcu_id* : int, PCU identifier
  - *hostname* : string, PCU hostname
  - *site_id* : int, Identifier of site where PCU is located
  - *ports* : array, List of the port numbers that each node is connected to
    - int

  - *model* : string, PCU model string
  - *password* : string, PCU username
  - *notes* : string, Miscellaneous notes

2.81. GetPeerData

Prototype:

GetPeerData (auth)

Description:

Returns lists of local objects that a peer should cache in its database as foreign objects. Also returns the list of foreign nodes in this database, for which the calling peer is authoritative, to assist in synchronization of slivers.

See the implementation of RefreshPeer for how this data is used.

Allowed Roles:

admin, peer

Parameters:

- `auth` : struct, API authentication structure
  - `AuthMethod` : string, Authentication method to use

Returns:

- struct
  - `Slices` : array, List of local slices
    - struct
  - `Keys` : array, List of local keys
    - struct
  - `Sites` : array, List of local sites
    - struct
  - `Persons` : array, List of local users
    - struct

- `Nodes` : array, List of local nodes
  - struct

- `db_time` : double, (Debug) Database fetch time

2.82. GetPeerName

Prototype:

GetPeerName (auth)

Description:

Returns this peer's name, as defined in the config as PLC_NAME

Allowed Roles:

admin, peer, node

Parameters:

- `auth` : struct, API authentication structure
  - `AuthMethod` : string, Authentication method to use

Returns:

- string, Peer name

2.83. GetPeers

Prototype:

GetPeers (auth, peer_filter, return_fields)

Description:

Returns an array of structs containing details about peers. If person_filter is specified and is an array of peer identifiers or peer names, or a struct of peer attributes, only peers matching the filter will be returned. If return_fields is specified, only the specified details will be returned.

Allowed Roles:

admin, node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *peer_filter* : array of int or string or struct
  - array of int or string
    - int, Peer identifier
    - string, Peer name

  - struct, Attribute filter
    - *key* : string or array of string
      - string, Peer GPG public key
      - array of string, Peer GPG public key

    - *cacert* : string or array of string
      - string, Peer SSL public certificate
      - array of string, Peer SSL public certificate

    - *peer_id* : int or array of int
      - int, Peer identifier
      - array of int, Peer identifier

    - *peername* : string or array of string
      - string, Peer name
      - array of string, Peer name

    - *peer_url* : string or array of string
      - string, Peer API URL

    - array of string, Peer API URL

- `return_fields` : array, List of fields to return
  - string

Returns:

- array of struct
  - `node_ids` : array, List of nodes for which this peer is authoritative
    - int
  - `key_ids` : array, List of keys for which this peer is authoritative
    - int
  - `person_ids` : array, List of users for which this peer is authoritative
    - int
  - `peername` : string, Peer name
  - `peer_url` : string, Peer API URL
  - `slice_ids` : array, List of slices for which this peer is authoritative
    - int
  - `key` : string, Peer GPG public key
  - `cacert` : string, Peer SSL public certificate
  - `site_ids` : array, List of sites for which this peer is authoritative
    - int
  - `peer_id` : int, Peer identifier

2.84. GetPersons

Prototype:

GetPersons (auth, person_filter, return_fields)

Description:

Returns an array of structs containing details about users. If person_filter is specified and is an array of user identifiers or usernames, or a struct of user attributes, only users matching the filter will be returned. If return_fields is specified, only the specified details will be returned.

Users and techs may only retrieve details about themselves. PIs may retrieve details about themselves and others at their sites. Admins and nodes may retrieve details about all accounts.

Allowed Roles:

admin, pi, user, tech, node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *person_filter* : array of int or string or struct
  - array of int or string
    - int, User identifier
    - string, Primary e-mail address

  - struct, Attribute filter
    - *bio* : string or array of string
      - string, Biography
      - array of string, Biography

    - *first_name* : string or array of string
      - string, Given name
      - array of string, Given name

    - *last_name* : string or array of string
      - string, Surname

    - array of string, Surname

- *last_updated* : int or array of int
    - int, Date and time of last update
    - array of int, Date and time of last update

- *title* : string or array of string
    - string, Title
    - array of string, Title

- *url* : string or array of string
    - string, Home page
    - array of string, Home page

- *verification_key* : string or array of string
    - string, Reset password key
    - array of string, Reset password key

- *enabled* : boolean or array of boolean
    - boolean, Has been enabled
    - array of boolean, Has been enabled

- *phone* : string or array of string
    - string, Telephone number
    - array of string, Telephone number

- *peer_person_id* : int or array of int
    - int, Foreign user identifier at peer
    - array of int, Foreign user identifier at peer

- *password* : string or array of string
    - string, Account password in crypt() form
    - array of string, Account password in crypt() form

- *person_id* : int or array of int

    - int, User identifier
    - array of int, User identifier

  - *date_created* : int or array of int
    - int, Date and time when account was created
    - array of int, Date and time when account was created

  - *peer_id* : int or array of int
    - int, Peer to which this user belongs
    - array of int, Peer to which this user belongs

  - *verification_expires* : int or array of int
    - int, Date and time when verification_key expires
    - array of int, Date and time when verification_key expires

  - *email* : string or array of string
    - string, Primary e-mail address
    - array of string, Primary e-mail address

- *return_fields* : array, List of fields to return
  - string

Returns:

- array of struct
  - *bio* : string, Biography
  - *first_name* : string, Given name
  - *last_name* : string, Surname
  - *last_updated* : int, Date and time of last update
  - *roles* : array, List of roles
    - string

  - *title* : string, Title

- *url* : string, Home page
- *key_ids* : array, List of key identifiers
  - int
- *enabled* : boolean, Has been enabled
- *slice_ids* : array, List of slice identifiers
  - int
- *phone* : string, Telephone number
- *peer_person_id* : int, Foreign user identifier at peer
- *role_ids* : array, List of role identifiers
  - int
- *person_id* : int, User identifier
- *date_created* : int, Date and time when account was created
- *site_ids* : array, List of site identifiers
  - int
- *peer_id* : int, Peer to which this user belongs
- *email* : string, Primary e-mail address

2.85. GetRoles

Prototype:

GetRoles (auth)

Description:

Get an array of structs containing details about all roles.

Allowed Roles:

admin, pi, user, tech, node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

Returns:

- array of struct
  - *name* : string, Role
  - *role_id* : int, Role identifier

2.86. GetSession

Prototype:

GetSession (auth)

Description:

Returns a new session key if a user or node authenticated successfully, faults otherwise.

Allowed Roles:

admin, pi, user, tech, node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

Returns:

- string, Session key

2.87. GetSites

Prototype:

GetSites (auth, site_filter, return_fields)

Description:

Returns an array of structs containing details about sites. If site_filter is specified and is an array of site identifiers or hostnames, or a struct of site attributes, only sites matching the filter will be returned. If return_fields is specified, only the specified details will be returned.

Allowed Roles:

admin, pi, user, tech, node, anonymous

Parameters:

- `auth` : struct, API authentication structure
  - `AuthMethod` : string, Authentication method to use

- `site_filter` : array of int or string or struct
  - array of int or string
    - int, Site identifier
    - string, Site slice prefix

  - struct, Attribute filter
    - `last_updated` : int or array of int
      - int, Date and time when site entry was last updated, in seconds since UNIX epoch
      - array of int, Date and time when site entry was last updated, in seconds since UNIX epoch

    - `name` : string or array of string
      - string, Full site name
      - array of string, Full site name

    - `url` : string or array of string
      - string, URL of a page that describes the site
      - array of string, URL of a page that describes the site

    - `site_id` : int or array of int
      - int, Site identifier
      - array of int, Site identifier

    - `enabled` : boolean or array of boolean

  - boolean, Has been enabled
  - array of boolean, Has been enabled

- *longitude* : double or array of double
  - double, Decimal longitude of the site
  - array of double, Decimal longitude of the site

- *latitude* : double or array of double
  - double, Decimal latitude of the site
  - array of double, Decimal latitude of the site

- *max_slices* : int or array of int
  - int, Maximum number of slices that the site is able to create
  - array of int, Maximum number of slices that the site is able to create

- *login_base* : string or array of string
  - string, Site slice prefix
  - array of string, Site slice prefix

- *is_public* : boolean or array of boolean
  - boolean, Publicly viewable site
  - array of boolean, Publicly viewable site

- *max_slivers* : int or array of int
  - int, Maximum number of slivers that the site is able to create
  - array of int, Maximum number of slivers that the site is able to create

- *date_created* : int or array of int
  - int, Date and time when site entry was created, in seconds since UNIX epoch
  - array of int, Date and time when site entry was created, in seconds since UNIX epoch

- *peer_site_id* : int or array of int
  - int, Foreign site identifier at peer
  - array of int, Foreign site identifier at peer

- *peer_id* : int or array of int
  - int, Peer to which this site belongs
  - array of int, Peer to which this site belongs

- *abbreviated_name* : string or array of string
  - string, Abbreviated site name
  - array of string, Abbreviated site name

- *return_fields* : array, List of fields to return
  - string

Returns:

- array of struct
  - *address_ids* : array, List of address identifiers
    - int

  - *pcu_ids* : array, List of PCU identifiers
    - int

  - *last_updated* : int, Date and time when site entry was last updated, in seconds since UNIX epoch
  - *name* : string, Full site name
  - *node_ids* : array, List of site node identifiers
    - int

  - *url* : string, URL of a page that describes the site
  - *enabled* : boolean, Has been enabled
  - *person_ids* : array, List of account identifiers
    - int

  - *site_id* : int, Site identifier
  - *longitude* : double, Decimal longitude of the site
  - *slice_ids* : array, List of slice identifiers

- int

- *max_slivers* : int, Maximum number of slivers that the site is able to create
- *max_slices* : int, Maximum number of slices that the site is able to create
- *login_base* : string, Site slice prefix
- *date_created* : int, Date and time when site entry was created, in seconds since UNIX epoch
- *latitude* : double, Decimal latitude of the site
- *is_public* : boolean, Publicly viewable site
- *peer_site_id* : int, Foreign site identifier at peer
- *peer_id* : int, Peer to which this site belongs
- *abbreviated_name* : string, Abbreviated site name

2.88. GetSliceAttributes

Prototype:

GetSliceAttributes (auth, slice_attribute_filter, return_fields)

Description:

Returns an array of structs containing details about slice and sliver attributes. An attribute is a sliver attribute if the node_id field is set. If slice_attribute_filter is specified and is an array of slice attribute identifiers, or a struct of slice attribute attributes, only slice attributes matching the filter will be returned. If return_fields is specified, only the specified details will be returned.

Users may only query attributes of slices or slivers of which they are members. PIs may only query attributes of slices or slivers at their sites, or of which they are members. Admins may query attributes of any slice or sliver.

Allowed Roles:

admin, pi, user, node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *slice_attribute_filter* : array of int or struct

- array of int, Slice attribute identifier
- struct, Attribute filter
  - *name* : string or array of string
    - string, Slice attribute type name
    - array of string, Slice attribute type name
  - *slice_id* : int or array of int
    - int, Slice identifier
    - array of int, Slice identifier
  - *slice_attribute_id* : int or array of int
    - int, Slice attribute identifier
    - array of int, Slice attribute identifier
  - *value* : string or array of string
    - string, Slice attribute value
    - array of string, Slice attribute value
  - *attribute_type_id* : int or array of int
    - int, Slice attribute type identifier
    - array of int, Slice attribute type identifier
  - *node_id* : int or array of int
    - int, Node identifier, if a sliver attribute
    - array of int, Node identifier, if a sliver attribute
  - *min_role_id* : int or array of int
    - int, Minimum (least powerful) role that can set or change this attribute
    - array of int, Minimum (least powerful) role that can set or change this attribute
  - *description* : string or array of string
    - string, Slice attribute type description
    - array of string, Slice attribute type description

- *return_fields* : array, List of fields to return
  - string

Returns:

- array of struct
  - *name* : string, Slice attribute type name
  - *slice_id* : int, Slice identifier
  - *slice_attribute_id* : int, Slice attribute identifier
  - *value* : string, Slice attribute value
  - *attribute_type_id* : int, Slice attribute type identifier
  - *node_id* : int, Node identifier, if a sliver attribute
  - *min_role_id* : int, Minimum (least powerful) role that can set or change this attribute
  - *description* : string, Slice attribute type description

2.89. GetSliceAttributeTypes

Prototype:

GetSliceAttributeTypes (auth, attribute_type_filter, return_fields)

Description:

Returns an array of structs containing details about slice attribute types. If attribute_type_filter is specified and is an array of slice attribute type identifiers, or a struct of slice attribute type attributes, only slice attribute types matching the filter will be returned. If return_fields is specified, only the specified details will be returned.

Allowed Roles:

admin, pi, user, tech, node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *attribute_type_filter* : array of int or string or struct
  - array of int or string
    - int, Slice attribute type identifier
    - string, Slice attribute type name
  - struct, Attribute filter
    - *attribute_type_id* : int or array of int
      - int, Slice attribute type identifier
      - array of int, Slice attribute type identifier
    - *min_role_id* : int or array of int
      - int, Minimum (least powerful) role that can set or change this attribute
      - array of int, Minimum (least powerful) role that can set or change this attribute
    - *name* : string or array of string
      - string, Slice attribute type name
      - array of string, Slice attribute type name
    - *description* : string or array of string
      - string, Slice attribute type description
      - array of string, Slice attribute type description
- *return_fields* : array, List of fields to return
  - string

Returns:

- array of struct
  - *attribute_type_id* : int, Slice attribute type identifier
  - *min_role_id* : int, Minimum (least powerful) role that can set or change this attribute
  - *name* : string, Slice attribute type name
  - *description* : string, Slice attribute type description

2.90. GetSliceInstantiations

Prototype:

GetSliceInstantiations (auth)

Description:

Returns an array of all valid slice instantiation states.

Allowed Roles:

admin, pi, user, tech, node

Parameters:

- `auth` : struct, API authentication structure
  - `AuthMethod` : string, Authentication method to use

Returns:

- array of string, Slice instantiation state

2.91. GetSlices

Prototype:

GetSlices (auth, slice_filter, return_fields)

Description:

Returns an array of structs containing details about slices. If slice_filter is specified and is an array of slice identifiers or slice names, or a struct of slice attributes, only slices matching the filter will be returned. If return_fields is specified, only the specified details will be returned.

Users may only query slices of which they are members. PIs may query any of the slices at their sites. Admins and nodes may query any slice. If a slice that cannot be queried is specified in slice_filter, details about that slice will not be returned.

Allowed Roles:

admin, pi, user, node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *slice_filter* : array of int or string or struct
  - array of int or string
    - int, Slice identifier
    - string, Slice name

  - struct, Attribute filter
    - *creator_person_id* : int or array of int
      - int, Identifier of the account that created this slice
      - array of int, Identifier of the account that created this slice

    - *instantiation* : string or array of string
      - string, Slice instantiation state
      - array of string, Slice instantiation state

    - *name* : string or array of string
      - string, Slice name
      - array of string, Slice name

    - *slice_id* : int or array of int
      - int, Slice identifier
      - array of int, Slice identifier

    - *created* : int or array of int
      - int, Date and time when slice was created, in seconds since UNIX epoch
      - array of int, Date and time when slice was created, in seconds since UNIX epoch

    - *url* : string or array of string
      - string, URL further describing this slice
      - array of string, URL further describing this slice

- *max_nodes* : int or array of int
  - int, Maximum number of nodes that can be assigned to this slice
  - array of int, Maximum number of nodes that can be assigned to this slice

- *expires* : int or array of int
  - int, Date and time when slice expires, in seconds since UNIX epoch
  - array of int, Date and time when slice expires, in seconds since UNIX epoch

- *site_id* : int or array of int
  - int, Identifier of the site to which this slice belongs
  - array of int, Identifier of the site to which this slice belongs

- *peer_slice_id* : int or array of int
  - int, Foreign slice identifier at peer
  - array of int, Foreign slice identifier at peer

- *peer_id* : int or array of int
  - int, Peer to which this slice belongs
  - array of int, Peer to which this slice belongs

- *description* : string or array of string
  - string, Slice description
  - array of string, Slice description

- *return_fields* : array, List of fields to return
  - string

Returns:

- array of struct
  - *creator_person_id* : int, Identifier of the account that created this slice
  - *instantiation* : string, Slice instantiation state
  - *slice_attribute_ids* : array, List of slice attributes

    - int

- *name* : string, Slice name
- *slice_id* : int, Slice identifier
- *created* : int, Date and time when slice was created, in seconds since UNIX epoch
- *url* : string, URL further describing this slice
- *max_nodes* : int, Maximum number of nodes that can be assigned to this slice
- *person_ids* : array, List of accounts that can use this slice
    - int

- *expires* : int, Date and time when slice expires, in seconds since UNIX epoch
- *site_id* : int, Identifier of the site to which this slice belongs
- *peer_slice_id* : int, Foreign slice identifier at peer
- *node_ids* : array, List of nodes in this slice
    - int

- *peer_id* : int, Peer to which this slice belongs
- *description* : string, Slice description

2.92. GetSliceTicket

Prototype:

GetSliceTicket (auth, slice_id_or_name)

Description:

Returns a ticket for, or signed representation of, the specified slice. Slice tickets may be used to manually instantiate or update a slice on a node. Present this ticket to the local Node Manager interface to redeem it.

If the slice has not been added to a node with AddSliceToNodes, and the ticket is redeemed on that node, it will be deleted the next time the Node Manager contacts the API.

Users may only obtain tickets for slices of which they are members. PIs may obtain tickets for any of the slices at their sites, or any slices of which they are members. Admins may obtain tickets for any slice.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, user, peer

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *slice_id_or_name* : int or string
  - int, Slice identifier
  - string, Slice name

Returns:

- string, Signed slice ticket

2.93. GetSlivers

Prototype:

GetSlivers (auth, node_id_or_hostname)

Description:

Returns a struct containing information about the specified node (or calling node, if called by a node and node_id_or_hostname is not specified), including the current set of slivers bound to the node.

All of the information returned by this call can be gathered from other calls, e.g. GetNodes, GetNodeNetworks, GetSlices, etc. This function exists almost solely for the benefit of Node Manager.

Allowed Roles:

admin, node

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *node_id_or_hostname* : int or string
  - int, Node identifier
  - string, Fully qualified hostname

Returns:

- struct
  - *timestamp* : int, Timestamp of this call, in seconds since UNIX epoch
  - *hostname* : string, Fully qualified hostname
  - *conf_files* : array of struct
    - *file_owner* : string, chown(1) owner
    - *postinstall_cmd* : string, Shell command to execute after installing
    - *error_cmd* : string, Shell command to execute if any error occurs
    - *preinstall_cmd* : string, Shell command to execute prior to installing
    - *node_ids* : int, List of nodes linked to this file
    - *dest* : string, Absolute path where file should be installed
    - *ignore_cmd_errors* : boolean, Install file anyway even if an error occurs
    - *enabled* : boolean, Configuration file is active
    - *conf_file_id* : int, Configuration file identifier
    - *file_permissions* : string, chmod(1) permissions
    - *source* : string, Relative path on the boot server where file can be downloaded
    - *nodegroup_ids* : int, List of node groups linked to this file
    - *always_update* : boolean, Always attempt to install file even if unchanged
    - *file_group* : string, chgrp(1) owner

  - *node_id* : int, Node identifier
  - *groups* : array of string, Node group name
  - *networks* : array of struct
    - *broadcast* : string, Network broadcast address
    - *is_primary* : boolean, Is the primary interface for this node
    - *network* : string, Subnet address
    - *ip* : string, IP address
    - *dns1* : string, IP address of primary DNS server
    - *hostname* : string, (Optional) Hostname

- *netmask* : string, Subnet mask
- *gateway* : string, IP address of primary gateway
- *nodenetwork_id* : int, Node interface identifier
- *mac* : string, MAC address
- *node_id* : int, Node associated with this interface
- *dns2* : string, IP address of secondary DNS server
- *bwlimit* : int, Bandwidth limit
- *type* : string, Address type (e.g., 'ipv4')
- *method* : string, Addressing method (e.g., 'static' or 'dhcp')

- *slivers* : array of struct
  - *instantiation* : string, Slice instantiation state
  - *name* : string, Slice name
  - *slice_id* : int, Slice identifier
  - *keys* : array of struct
    - *key_type* : string, Key type
    - *key* : string, Key value

  - *expires* : int, Date and time when slice expires, in seconds since UNIX epoch
  - *attributes* : array of struct
    - *name* : string, Slice attribute type name
    - *value* : string, Slice attribute value

2.94. NotifyPersons

Prototype:

NotifyPersons (auth, person_filter, subject, body)

Description:

Sends an e-mail message to the specified users. If person_filter is specified and is an array of user identifiers or usernames, or a struct of user attributes, only users matching the filter will receive the message.

Returns 1 if successful.

Allowed Roles:

admin, node

Parameters:

- `auth` : struct, API authentication structure
  - `AuthMethod` : string, Authentication method to use

- `person_filter` : array of int or string or struct
  - array of int or string
    - int, User identifier
    - string, Primary e-mail address

  - struct, Attribute filter
    - `bio` : string or array of string
      - string, Biography
      - array of string, Biography

    - `first_name` : string or array of string
      - string, Given name
      - array of string, Given name

    - `last_name` : string or array of string
      - string, Surname
      - array of string, Surname

    - `last_updated` : int or array of int
      - int, Date and time of last update
      - array of int, Date and time of last update

    - `title` : string or array of string
      - string, Title

    - array of string, Title

- *url* : string or array of string
    - string, Home page
    - array of string, Home page

- *verification_key* : string or array of string
    - string, Reset password key
    - array of string, Reset password key

- *enabled* : boolean or array of boolean
    - boolean, Has been enabled
    - array of boolean, Has been enabled

- *phone* : string or array of string
    - string, Telephone number
    - array of string, Telephone number

- *peer_person_id* : int or array of int
    - int, Foreign user identifier at peer
    - array of int, Foreign user identifier at peer

- *password* : string or array of string
    - string, Account password in crypt() form
    - array of string, Account password in crypt() form

- *person_id* : int or array of int
    - int, User identifier
    - array of int, User identifier

- *date_created* : int or array of int
    - int, Date and time when account was created
    - array of int, Date and time when account was created

- *peer_id* : int or array of int

- int, Peer to which this user belongs
- array of int, Peer to which this user belongs

- *verification_expires* : int or array of int
  - int, Date and time when verification_key expires
  - array of int, Date and time when verification_key expires

- *email* : string or array of string
  - string, Primary e-mail address
  - array of string, Primary e-mail address

- *subject* : string, E-mail subject
- *body* : string, E-mail body

Returns:

- int, 1 if successful

2.95. RebootNode

Prototype:

RebootNode (auth, node_id_or_hostname)

Description:

Sends the specified node a specially formatted UDP packet which should cause it to reboot immediately.

Admins can reboot any node. Techs and PIs can only reboot nodes at their site.

Returns 1 if the packet was successfully sent (which only whether the packet was sent, not whether the reboot was successful).

Allowed Roles:

admin, pi, tech

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *node_id_or_hostname* : int or string
  - int, Node identifier
  - string, Fully qualified hostname

Returns:

- int, 1 if successful

2.96. RefreshPeer

Prototype:

RefreshPeer (auth, peer_id_or_peername)

Description:

Fetches node and slice data from the specified peer and caches it locally; also deletes stale entries. Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *peer_id_or_peername* : int or string
  - int, Peer identifier
  - string, Peer name

Returns:

- int, 1 if successful

2.97. ResetPassword

Prototype:

ResetPassword (auth, person_id_or_email, verification_key, verification_expires)

Description:

If verification_key is not specified, then a new verification_key will be generated and stored with the user's account. The key will be e-mailed to the user in the form of a link to a web page.

The web page should verify the key by calling this function again and specifying verification_key. If the key matches what has been stored in the user's account, a new random password will be e-mailed to the user.

Returns 1 if verification_key was not specified, or was specified and is valid, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *person_id_or_email* : int or string
  - int, User identifier
  - string, Primary e-mail address

- *verification_key* : string, Reset password key
- *verification_expires* : int, Date and time when verification_key expires

Returns:

- int, 1 if verification_key is valid

2.98. SetPersonPrimarySite

Prototype:

SetPersonPrimarySite (auth, person_id_or_email, site_id_or_login_base)

Description:

Makes the specified site the person's primary site. The person must already be a member of the site.

Admins may update anyone. All others may only update themselves.

Allowed Roles:

admin, pi, user, tech

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *person_id_or_email* : int or string
  - int, User identifier
  - string, Primary e-mail address
- *site_id_or_login_base* : int or string
  - int, Site identifier
  - string, Site slice prefix

Returns:

- int, 1 if successful

2.99. UpdateAddress

Prototype:

UpdateAddress (auth, address_id, address_fields)

Description:

Updates the parameters of an existing address with the values in address_fields.

PIs may only update addresses of their own sites.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *address_id* : int, Address identifier
- *address_fields* : struct
  - *city* : string, City
  - *country* : string, Country
  - *line3* : string, Address line 3
  - *line2* : string, Address line 2
  - *line1* : string, Address line 1
  - *state* : string, State or province
  - *postalcode* : string, Postal code

Returns:

- int, 1 if successful

2.100. UpdateAddressType

Prototype:

UpdateAddressType (auth, address_type_id_or_name, address_type_fields)

Description:

Updates the parameters of an existing address type with the values in address_type_fields.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *address_type_id_or_name* : int or string
  - int, Address type identifier
  - string, Address type

- *address_type_fields* : struct
  - *name* : string, Address type
  - *description* : string, Address type description

Returns:

- int, 1 if successful

2.101. UpdateConfFile

Prototype:

UpdateConfFile (auth, conf_file_id, conf_file_fields)

Description:

Updates a node configuration file. Only the fields specified in conf_file_fields are updated, all other fields are left untouched.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *conf_file_id* : int, Configuration file identifier
- *conf_file_fields* : struct
  - *file_owner* : string, chown(1) owner
  - *postinstall_cmd* : string, Shell command to execute after installing
  - *error_cmd* : string, Shell command to execute if any error occurs
  - *preinstall_cmd* : string, Shell command to execute prior to installing
  - *dest* : string, Absolute path where file should be installed
  - *ignore_cmd_errors* : boolean, Install file anyway even if an error occurs
  - *enabled* : boolean, Configuration file is active
  - *file_permissions* : string, chmod(1) permissions
  - *source* : string, Relative path on the boot server where file can be downloaded
  - *always_update* : boolean, Always attempt to install file even if unchanged
  - *file_group* : string, chgrp(1) owner

Returns:

- int, 1 if successful

2.102. UpdateKey

Prototype:

UpdateKey (auth, key_id, key_fields)

Description:

Updates the parameters of an existing key with the values in key_fields.

Non-admins may only update their own keys.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, tech, user

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *key_id* : int, Key identifier
- *key_fields* : struct
  - *key_type* : string, Key type
  - *key* : string, Key value

Returns:

- int, 1 if successful

2.103. UpdateMessage

Prototype:

UpdateMessage (auth, message_id, message_fields)

Description:

Updates the parameters of an existing message template with the values in message_fields.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *message_id* : string, Message identifier
- *message_fields* : struct
  - *enabled* : boolean, Message is enabled
  - *template* : string, Message template

Returns:

- int, 1 if successful

2.104. UpdateNodeGroup

Prototype:

UpdateNodeGroup (auth, nodegroup_id_or_name, nodegroup_fields)

Description:

Updates a custom node group.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *nodegroup_id_or_name* : int or string
  - int, Node group identifier
  - string, Node group name

- *nodegroup_fields* : struct
  - *name* : string, Node group name
  - *description* : string, Node group description

Returns:

- int, 1 if successful

2.105. UpdateNodeNetwork

Prototype:

UpdateNodeNetwork (auth, nodenetwork_id, nodenetwork_fields)

Description:

Updates an existing node network. Any values specified in nodenetwork_fields are used, otherwise defaults are used. Acceptable values for method are dhcp and static. If type is static, then ip, gateway, network, broadcast, netmask, and dns1 must all be specified in nodenetwork_fields. If type is dhcp, these parameters, even if specified, are ignored.

PIs and techs may only update networks associated with their own nodes. Admins may update any node network.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, tech

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *nodenetwork_id* : int, Node interface identifier
- *nodenetwork_fields* : struct
  - *network* : string, Subnet address
  - *is_primary* : boolean, Is the primary interface for this node
  - *dns1* : string, IP address of primary DNS server
  - *hostname* : string, (Optional) Hostname
  - *mac* : string, MAC address
  - *bwlimit* : int, Bandwidth limit

- *broadcast* : string, Network broadcast address
- *method* : string, Addressing method (e.g., 'static' or 'dhcp')
- *netmask* : string, Subnet mask
- *dns2* : string, IP address of secondary DNS server
- *ip* : string, IP address
- *type* : string, Address type (e.g., 'ipv4')
- *gateway* : string, IP address of primary gateway

Returns:

- int, 1 if successful

2.106. UpdateNode

Prototype:

UpdateNode (auth, node_id_or_hostname, node_fields)

Description:

Updates a node. Only the fields specified in node_fields are updated, all other fields are left untouched.

PIs and techs can update only the nodes at their sites. Only admins can update the key, session, and boot_nonce fields.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, tech

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *node_id_or_hostname* : int or string
  - int, Node identifier
  - string, Fully qualified hostname

- *node_fields* : struct
  - *version* : string, Apparent Boot CD version
  - *boot_state* : string, Boot state
  - *hostname* : string, Fully qualified hostname
  - *boot_nonce* : string, (Admin only) Random value generated by the node at last boot
  - *session* : string, (Admin only) Node session value
  - *key* : string, (Admin only) Node key
  - *model* : string, Make and model of the actual machine

Returns:

- int, 1 if successful

2.107. UpdatePCU

Prototype:

UpdatePCU (auth, pcu_id, pcu_fields)

Description:

Updates the parameters of an existing PCU with the values in pcu_fields.

Non-admins may only update PCUs at their sites.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, tech

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *pcu_id* : int, PCU identifier

- *pcu_fields* : struct
  - *username* : string, PCU username
  - *protocol* : string, PCU protocol, e.g. ssh, https, telnet
  - *node_ids* : array, List of nodes that this PCU controls
    - int
  - *ip* : string, PCU IP address
  - *notes* : string, Miscellaneous notes
  - *hostname* : string, PCU hostname
  - *model* : string, PCU model string
  - *password* : string, PCU username
  - *ports* : array, List of the port numbers that each node is connected to
    - int

Returns:

- int, 1 if successful

2.108. UpdatePeer

Prototype:

UpdatePeer (auth, peer_id_or_name, peer_fields)

Description:

Updates a peer. Only the fields specified in peer_fields are updated, all other fields are left untouched.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure

    - *AuthMethod* : string, Authentication method to use

- *peer_id_or_name* : int or string
    - int, Peer identifier
    - string, Peer name

- *peer_fields* : struct
    - *key* : string, Peer GPG public key
    - *cacert* : string, Peer SSL public certificate
    - *peername* : string, Peer name
    - *peer_url* : string, Peer API URL

Returns:

- int, 1 if successful

2.109. UpdatePerson

Prototype:

UpdatePerson (auth, person_id_or_email, person_fields)

Description:

Updates a person. Only the fields specified in person_fields are updated, all other fields are left untouched.

Users and techs can only update themselves. PIs can only update themselves and other non-PIs at their sites. Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, user, tech

Parameters:

- *auth* : struct, API authentication structure
    - *AuthMethod* : string, Authentication method to use

- *person_id_or_email* : int or string
  - int, User identifier
  - string, Primary e-mail address

- *person_fields* : struct
  - *bio* : string, Biography
  - *first_name* : string, Given name
  - *last_name* : string, Surname
  - *title* : string, Title
  - *url* : string, Home page
  - *enabled* : boolean, Has been enabled
  - *phone* : string, Telephone number
  - *password* : string, Account password in crypt() form
  - *email* : string, Primary e-mail address

Returns:

- int, 1 if successful

2.110. UpdateSite

Prototype:

UpdateSite (auth, site_id_or_login_base, site_fields)

Description:

Updates a site. Only the fields specified in update_fields are updated, all other fields are left untouched.

PIs can only update sites they are a member of. Only admins can update max_slices, max_slivers, and login_base.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *site_id_or_login_base* : int or string
  - int, Site identifier
  - string, Site slice prefix

- *site_fields* : struct
  - *name* : string, Full site name
  - *url* : string, URL of a page that describes the site
  - *enabled* : boolean, Has been enabled
  - *longitude* : double, Decimal longitude of the site
  - *latitude* : double, Decimal latitude of the site
  - *max_slices* : int, Maximum number of slices that the site is able to create
  - *login_base* : string, Site slice prefix
  - *max_slivers* : int, Maximum number of slivers that the site is able to create
  - *is_public* : boolean, Publicly viewable site
  - *abbreviated_name* : string, Abbreviated site name

Returns:

- int, 1 if successful

2.111. UpdateSliceAttribute

Prototype:

UpdateSliceAttribute (auth, slice_attribute_id, value)

Description:

Updates the value of an existing slice or sliver attribute.

Users may only update attributes of slices or slivers of which they are members. PIs may only update attributes of slices or slivers at their sites, or of which they are members. Admins may update attributes of any slice or sliver.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, user

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use
- *slice_attribute_id* : int, Slice attribute identifier
- *value* : string, Slice attribute value

Returns:

- int, 1 if successful

2.112. UpdateSliceAttributeType

Prototype:

UpdateSliceAttributeType (auth, attribute_type_id_or_name, attribute_type_fields)

Description:

Updates the parameters of an existing attribute with the values in attribute_type_fields.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *attribute_type_id_or_name* : int or string
  - int, Slice attribute type identifier
  - string, Slice attribute type name

- *attribute_type_fields* : struct
  - *min_role_id* : int, Minimum (least powerful) role that can set or change this attribute
  - *name* : string, Slice attribute type name
  - *description* : string, Slice attribute type description

Returns:

- int, 1 if successful

2.113. UpdateSlice

Prototype:

UpdateSlice (auth, slice_id_or_name, slice_fields)

Description:

Updates the parameters of an existing slice with the values in slice_fields.

Users may only update slices of which they are members. PIs may update any of the slices at their sites, or any slices of which they are members. Admins may update any slice.

Only PIs and admins may update max_nodes. Slices cannot be renewed (by updating the expires parameter) more than 8 weeks into the future.

Returns 1 if successful, faults otherwise.

Allowed Roles:

admin, pi, user

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *slice_id_or_name* : int or string
  - int, Slice identifier
  - string, Slice name

- *slice_fields* : struct
  - *url* : string, URL further describing this slice
  - *max_nodes* : int, Maximum number of nodes that can be assigned to this slice
  - *instantiation* : string, Slice instantiation state
  - *expires* : int, Date and time when slice expires, in seconds since UNIX epoch
  - *description* : string, Slice description

Returns:

- int, 1 if successful

2.114. VerifyPerson

Prototype:

VerifyPerson (auth, person_id_or_email, verification_key, verification_expires)

Description:

Verify a new (must be disabled) user's e-mail address and registration.

If verification_key is not specified, then a new verification_key will be generated and stored with the user's account. The key will be e-mailed to the user in the form of a link to a web page.

The web page should verify the key by calling this function again and specifying verification_key. If the key matches what has been stored in the user's account, then an e-mail will be sent to the user's PI (and support if the user is requesting a PI role), asking the PI (or support) to enable the account.

Returns 1 if the verification key if valid.

Allowed Roles:

admin

Parameters:

- *auth* : struct, API authentication structure
  - *AuthMethod* : string, Authentication method to use

- *person_id_or_email* : int or string
  - int, User identifier
  - string, Primary e-mail address

- *verification_key* : string, Reset password key
- *verification_expires* : int, Date and time when verification_key expires

Returns:

- int, 1 if verification_key is valid

2.115. system.listMethods

Prototype:

system.listMethods ()

Description:

This method lists all the methods that the XML-RPC server knows how to dispatch.

Allowed Roles:

Parameters:

- None

Returns:

- array, List of methods

2.116. system.methodHelp

Prototype:

system.methodHelp (method)

Description:

Returns help text if defined for the method passed, otherwise returns an empty string.

Allowed Roles:

Parameters:

- `method` : string, Method name

Returns:

- string, Method help

2.117. system.methodSignature

Prototype:

system.methodSignature (method)

Description:

Returns an array of known signatures (an array of arrays) for the method name passed. If no signatures are known, returns a none-array (test for type != array to detect missing signature).

Allowed Roles:

Parameters:

- `method` : string, Method name

Returns:

- array of array, Method signature
  - string

2.118. system.multicall

Prototype:

system.multicall (calls)

Description:

Process an array of calls, and return an array of results. Calls should be structs of the form

{'methodName': string, 'params': array}

Each result will either be a single-item array containg the result value, or a struct of the form

{'faultCode': int, 'faultString': string}

This is useful when you need to make lots of small calls without lots of round trips.

Allowed Roles:

Parameters:

- `calls` : array of struct
  - `params` : array, Method arguments
  - `methodName` : string, Method name

Returns:

- array of mixed or struct
  - array of mixed
  - struct
    - `faultCode` : int, XML-RPC fault code
    - `faultString` : int, XML-RPC fault detail