pydactyl

Release 2.0.3

Jul 27, 2023

Contents

An easy to use Python wrapper for the Pterodactyl Panel API.

CHAPTER 1

Documentation

This documentation is generated from docstrings so the class names do not usually match the way they are accessed when using PterodactylClient.

For example the *get_server()* method documented in the *pydactyl.api.client.servers.base.ServersBase* class is accessed by calling *client.servers.get_server()*.

The majority of names in PterodactylClient match their path minus the class name, so *pydactyl.api.client.account.Account.get_server()* is *client.account.get_server()*. Classes named or ending with Base are imported into the parent namespace.

More example usage can be found can be found in the README at https://github.com/iamkubi/pydactyl.

CHAPTER 2

Client API

The Client API are endpoints available to any Pterodactyl user account and require a Client API key generated in the account settings.

2.1 client.account

class pydactyl.api.client.account.**Account**

Class for interacting with the Pterodactyl Client API.

Methods in this class appear in the base **client.account** namespace when using PterodactylClient.

api_key_create(*description: str*, *allowed_ips: list*)

Create a client API key.

Parameters

- **description** (*str*) – Note for the API key
- **allowed_ips** (*iter*) – List of allowed IPs

api_key_delete(*identifier*)

Delete a client API key.

Parameters **identifier** (*str*) – API key identifier

api_key_list()

List client's API keys.

disable_2fa(*password: str*)

Disables 2FA on the account.

Parameters **password** (*str*) – User's account password

enable_2fa(*code: str*)

Enables TOTP 2FA.

Takes a TOTP code generated by a client configured using the code from get_2fa_setup_code(). This does not take the code directly and you must first setup an authenticator app such as Google Authenticator to generate the code provided to this method.

Parameters **code** (*str*) – TOTP code generated by authenticator app

get_2fa_setup_code()
Generates a TOTP QR code image to allow the setup of 2FA.

get_account()
List details of the account belonging to the client API key.

update_email(*email: str*, *password: str*)
Updates the email address of the account.

Parameters

- **email** (*str*) – New email address
- **password** (*str*) – User's account password

update_password(*current: str*, *new: str*, *new_confirm: str*)
Updates the password of the account.

Parameters

- **current** (*str*) – User's current password
- **new** (*str*) – New password for the account, must match new_confirm
- **new_confirm** (*str*) – New password for the account, must match new

2.2 client.servers

class pydactyl.api.client.servers.base.**ServersBase**
Pterodactyl Client Server Base API.

Methods in this class appear in the base **client.servers** namespace when using PterodactylClient.

get_server(*server_id*, *detail=False*, *includes=None*, *params=None*)
Get information for the specified server.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **detail** (*bool*) – If True includes the object type and a nested data structure. This is not particularly useful.
- **includes** (*iter*) – List of includes, e.g. ('egg', 'subusers')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

get_server_utilization(*server_id*, *detail=False*)
Get resource utilization information for the specified server.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **detail** (*bool*) – If True includes the object type and a nested data structure. This is not particularly useful.

get_websocket(*server_id*)
Generates credentials to connect to the server's websocket.

Parameters **server_id** (*str*) – Server identifier (abbreviated UUID)

Returns Response with token and websocket address

Return type response(dict)

list_permissions()
Retries all available permissions.

This is used by the frontend. I have no idea what this does.

list_servers(*includes=None*, *params=None*)
List all servers the client has access to.

Parameters

- **includes** (*iter*) – List of includes, e.g. ('egg', 'subusers')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

send_console_command(*server_id*, *cmd*)
Sends a console command to the specified server.

The server must be online, otherwise API will return a HTTP 412 error. If successful, there will be an empty response body.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **cmd** (*str*) – Console command to send to the server

send_power_action(*server_id*, *signal*)
Sends a console command to the specified server.

The server must be online, otherwise API will return a HTTP 412 error. If successful, there will be an empty response body.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **signal** (*str*) – Power signal to send to the server. Valid options: start - Sends the startup command to the server. stop - Sends the stop command to the server. restart - Stops the server then immediately starts it. kill - Instantly ends all processes and marks the server as

  stopped. The kill signal can corrupt server files and should only be used as a last resort.

2.3 client.servers.backups

class pydactyl.api.client.servers.backups.**Backups**
Pterodactyl Client Server Backups API.

create_backup(*server_id: str*)
Create a new backup of the specified server.

Parameters **server_id** (*str*) – Server identifier (abbreviated UUID)

delete_backup(*server_id: str*, *backup_id: str*)
Deletes the specified backup.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **backup_id** (*str*) – Backup identifier (long UUID)

get_backup_detail (*server_id: str*, *backup_id: str*)
Retrieves information about the specified backup.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **backup_id** (*str*) – Backup identifier (long UUID)

get_backup_download (*server_id: str*, *backup_id: str*)
Generates a download link for the specified backup.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **backup_id** (*str*) – Backup identifier (long UUID)

list_backups (*server_id: str*)
List files belonging to specified server.

Optionally specify a directory and only return results in the specified directory. Directory is relative to the server's root.

Parameters **server_id** (*str*) – Server identifier (abbreviated UUID)

2.4 client.servers.databases

class pydactyl.api.client.servers.databases.**Databases**
Pterodactyl Client Server Databases API.

create_database (*server_id: str*, *name: str*, *remote: str = '%'*)
Create a new database for the specified server.

Limits connections to the address specified in remote, defaulting to allowing from all.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **remote** (*str*) – Remote address to allow connections from (e.g. 1.2.3.4)

delete_database (*server_id: str*, *database_id: str*)
Deletes the specified database.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **database_id** (*str*) – Database identifier (long UUID)

list_databases (*server_id: str*, *include_passwords: bool = False*, *includes: list = []*, *params: dict = None*)
List all databases for a server.

Optionally includes the database user passwords.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)

- **include_passwords** (*bool*) – True to include database user passwords
- **includes** (*iter*) – List of includes, e.g. ('password')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

rotate_database_password(*server_id: str*, *database_id: str*)
Changes the password of the specified database.

Generates a new password and returns it in the response.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **database_id** (*str*) – Database identifier (long UUID)

2.5 client.servers.files

class pydactyl.api.client.servers.files.**Files**
Class for interacting with the Pterodactyl Client API.

compress_files(*server_id: str*, *files: iter*, *path: str = '/'*)
Creates a compressed archive.

Creates a tar.gz compressed file containing the listed files.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **files** (*iter*) – List of files to add to the archive
- **path** (*str*) – Root path to create the archive from

copy_file(*server_id: str*, *path: str*)
Makes a copy of a file.

This is primarily used by the file manager.

Makes a copy of the file with a unique name. You cannot specify the new name of the file, it just picks one for you. For example 'test.txt' will have a copy created named 'test copy.txt'. Running it again will create 'test copy 1.txt'.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **path** (*str*) – URL encoded path to desired file (e.g. 'eula.txt')

create_folder(*server_id: str*, *name: str*, *path: str = '/'*)
Creates the specified folder in the specified directory.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **name** (*str*) – Name of the directory to create
- **path** (*str*) – Root path to create the directory in

decompress_file(*server_id: str*, *file: str*, *path: str = '/'*)
Decompresses an archive.

Decompresses a compressed file to the specified path.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **file** (*str*) – Name of the archive file to decompress
- **path** (*str*) – Root path to decompress in

delete_files (*server_id: str*, *files: iter*, *path: str = '/'*)
Deletes the specified file(s) or directory(s).

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **files** (*iter*) – List of files to delete
- **path** (*str*) – Root path look for files in

download_file (*server_id: str*, *path: str*) → str
Get a download link for the specified file on the specified server.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **path** (*str*) – URL encoded path to desired file (e.g. 'eula.txt')

Returns Signed URL to download file

Return type response(str)

get_file_contents (*server_id: str*, *path: str*) → str
Get contents of the specified file on the specified server.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **path** (*str*) – URL encoded path to desired file (e.g. 'eula.txt')

get_upload_file_url (*server_id: str*) → str
Returns a signed URL used to upload files to the server using POST.

Parameters **server_id** (*str*) – Server identifier (abbreviated UUID)

list_files (*server_id: str*, *path: str = None*)
List files belonging to specified server.

Optionally specify a directory and only return results in the specified directory. Directory is relative to the server's root.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **path** (*str*) – Path to list in (e.g. 'save_game')

rename_file (*server_id: str*, *old_name: str*, *new_name: str*, *root: str = '/'*)
Rename a file.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **old_name** (*str*) – Name of existing file to rename
- **new_name** (*str*) – New filename

- **root** (*str*) – Path to files, relative to server root

write_file(*server_id: str*, *path: str*, *contents: str*)
Writes contents to a file.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **path** (*str*) – Path to desired file (e.g. 'eula.txt')
- **contents** (*str*) – Contents to write to the file.

2.6 client.servers.network

class pydactyl.api.client.servers.network.**Network**
Pterodactyl Client Server Network API.

assign_allocation(*server_id: str*)
Assigns an allocation to the server.

Automatically assigns a new allocation if auto-assign is enabled on the instance.

Parameters **server_id** (*str*) – Server identifier (abbreviated UUID)

list_allocations(*server_id: str*)
Retrieves network information for the specified server.

Parameters **server_id** (*str*) – Server identifier (abbreviated UUID)

set_allocation_note(*server_id: str*, *allocation_id: int*, *note: str*)
Sets the note on an allocation.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **allocation_id** (*int*) – Allocation identifier (e.g. 2)
- **note** (*str*) – Contents of the note

set_primary_allocation(*server_id: str*, *allocation_id: int*)
Sets the specified allocation as the primary allocation.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **allocation_id** (*int*) – Allocation identifier (e.g. 2)

unassign_allocation(*server_id: str*, *allocation_id: int*)
Deletes the specified non-primary allocation.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **allocation_id** (*int*) – Allocation identifier (e.g. 2)

2.7 client.servers.settings

class pydactyl.api.client.servers.settings.**Settings**
Pterodactyl Client Server Settings API.

reinstall_server(*server_id: str*)
Reinstalls the specified server.

Parameters **server_id** (*str*) – Server identifier (abbreviated UUID)

rename_server(*server_id: str*, *name: str*)
Renames the server.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **name** (*str*) – New name for the server

2.8 client.servers.startup

class pydactyl.api.client.servers.startup.**Startup**
Pterodactyl Client Server Startup API.

list_variables(*server_id: str*)
Lists all variables on the server.

Parameters **server_id** (*str*) – Server identifier (abbreviated UUID)

update_variable(*server_id: str*, *name: str*, *value: str*)
Updates the specified server variable.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **name** (*str*) – Variable name to update
- **value** (*str*) – Value to assign to the updated variable

2.9 client.servers.users

class pydactyl.api.client.servers.users.**Users**
Pterodactyl Client Server Backups API.

create_user(*server_id: str*, *email: str*, *permissions: iter*, *username: str = None*)
Adds a new user to the server.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **email** (*str*) – Email address of the new user
- **permissions** (*iter*) – List of permissions to assign to the user
- **username** (*str*) – Username to assign, randomized if not specified

delete_user(*server_id: str*, *user_id: str*)
Deletes the specified user.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **user_id** (*str*) – User identifier (long UUID)

get_user(*server_id: str*, *user_id: str*)
Retrieves details about the specified user.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **user_id** (*str*) – User identifier (long UUID)

list_users(*server_id: str*)
List all users added to the server.

Includes user details and permissions assigned to them.

Parameters **server_id** (*str*) – Server identifier (abbreviated UUID)

update_user(*server_id: str*, *user_id: str*, *permissions*)
Updates the specified user.

This probably has more options than the documentation list.

Parameters

- **server_id** (*str*) – Server identifier (abbreviated UUID)
- **user_id** (*str*) – User identifier (long UUID)
- **permissions** (*iter*) – List of permissions to assign to the user

CHAPTER 3

Application API

Application API endpoints are only available to Pterodactyl Administrators and require an Application API key generated in the Admin panel.

3.1 locations

class pydactyl.api.locations.**Locations**
Class for interacting with the Pterdactyl Locations API.

create_location(*shortcode*, *description*)
Creates a new location.

Parameters

- **shortcode** (*str*) – Short identifier between 1 and 60 characters, e.g. us.nyc.lvl3
- **description** (*str*) – A long description of the location. Max 255 characters.

delete_location(*location_id*)
Delete an existing location.

Parameters **location_id** (*int*) – Pterodactyl Location ID.

edit_location(*location_id*, *shortcode=None*, *description=None*)
Modify an existing location.

Parameters

- **location_id** (*int*) – Pterodactyl Location ID.
- **shortcode** (*str*) – Short identifier between 1 and 60 characters, e.g. us.nyc.lvl3
- **description** (*str*) – A long description of the location. Max 255 characters.

get_location_info(*location_id*, *includes=None*, *params=None*)
Get detailed info for the specified location.

Parameters

- **location_id** (*int*) – Pterodactyl Location ID.
- **includes** (*iter*) – List of includes, e.g. ('nodes', 'servers')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

list_locations(*includes=None*, *params=None*)
List all locations.

Parameters

- **includes** (*iter*) – List of includes, e.g. ('nodes', 'servers')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

3.2 nests

class pydactyl.api.nests.**Nests**
Class for interacting with the Pterdactyl Nests API.

get_egg_info(*nest_id*, *egg_id*, *includes=None*, *params=None*)
Get detailed info for the specified egg.

Parameters

- **nest_id** (*int*) – Pterodactyl Nest ID.
- **egg_id** (*int*) – Pterodactyl Egg ID.
- **includes** (*iter*) – List of includes, e.g. ('config', 'nest', 'script')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

get_eggs_in_nest(*nest_id*, *includes=None*, *params=None*)
Get detailed info for the specified nest.

Parameters

- **nest_id** (*int*) – Pterodactyl Nest ID.
- **includes** (*iter*) – List of includes, e.g. ('config', 'nest', 'script')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

get_nest_info(*nest_id*, *includes=None*, *params=None*)
Get detailed info for the specified nest.

Parameters

- **nest_id** (*int*) – Pterodactyl Nest ID.
- **includes** (*iter*) – List of includes, e.g. ('eggs', 'servers')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

list_nests(*includes=None*, *params=None*)
List all nests.

Parameters

- **includes** (*iter*) – List of includes, e.g. ('eggs', 'servers')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

3.3 nodes

class pydactyl.api.nodes.**Nodes**
Class for interacting with the Pterdactyl Nodes API.

create_allocations(*node_id*, *ip*, *ports*, *alias=None*)
Create one or more allocations.

Parameters

- **node_id** (*int*) – Pterodactyl Node ID.
- **ip** (*str*) – The IP address to create the allocation on.
- **ports** (*iter*) – List of strings representing strings. Can use ranges as supported by Pterodactyl, e.g. ["4000", "4003-4005"]
- **alias** (*str*) – Optional IP alias. Used if you want to display a different IP address to Panel users, for example to display the external IP when behind a NAT.

create_node(*name*, *description*, *location_id*, *fqdn*, *memory*, *disk*, *memory_overallocate=0*, *disk_overallocate=0*, *use_ssl=True*, *behind_proxy=False*, *daemon_base='/srv/daemon-data'*, *daemon_sftp=2022*, *daemon_listen=8080*, *upload_size=100*, *public=True*, *maintenance_mode=False*)
Creates a new node.

Parameters

- **name** (*str*) – Node Name, 1-100 characters, valid characters: **a-zA-Z0-9_**.-[Space]
- **description** (*str*) – A long description of the node.
- **location_id** (*int*) – A valid Location ID
- **fqdn** (*str*) – Domain name used to connect to the daemon. Alternatively an IP address if not using SSL.
- **memory** (*int*) – Memory in MB that is available to the daemon for allocation to servers.
- **memory_overallocate** (*int*) – Percentage of memory that can be overallocated, e.g. 150
- **disk** (*int*) – Disk space in MB that is available to the daemon for allocation to servers.
- **disk_overallocate** (*int*) – Percentage of disk space that can be overallocated, e.g. 150
- **use_ssl** (*bool*) – True to enable SSL, false for insecure HTTP
- **behind_proxy** (*bool*) – Set to True if running behind a proxy like CloudFlare. Skips certificate check on boot.
- **daemon_base** (*str*) – Directory where server files will be stored.
- **daemon_sftp** (*int*) – Port used by daemon for SFTP.
- **daemon_listen** (*int*) – Port used by the daemon.
- **public** (*bool*) – Set to False to prevent servers from being created on this node.
- **maintenance_mode** (*bool*) – Set to True to disable the node or something.

delete_allocation(*node_id*, *allocation_id*)
Deletes the specified allocation on the specified node.

The Pterodactyl API currently limits delete_allocation to a single allocation per API call.

See: https://github.com/pterodactyl/panel/issues/4373

Parameters

- **node_id** (*int*) – Pterodactyl Node ID.
- **allocation_id** (*int*) – Pterodactyl Allocation ID. This is the internal ID assigned to the allocation, not the port number.

delete_node(*node_id*)
Delete an existing node.

Parameters **node_id** (*int*) – Pterodactyl Node ID.

edit_node(*node_id*, *name=None*, *description=None*, *location_id=None*, *fqdn=None*, *use_ssl=None*, *behind_proxy=None*, *maintenance_mode=None*, *memory=None*, *memory_overallocate=None*, *disk=None*, *disk_overallocate=None*, *upload_size=None*, *daemon_sftp=None*, *daemon_listen=None*)
Update the configuration for an existing node.

Modifies an existing node identified by node_id and updates any parameters that are passed.

*** WARNING *** This endpoint currently requires that you specify all parameters in order to edit a node. This will be updated in the future to automatically fetch existing values, however since multiple endpoints require this functionality it will be added in a common location.

Parameters

- **node_id** (*int*) – Pterodactyl Node ID.
- **name** (*str*) – Node name
- **description** (*str*) – A long description of the node. Max 255 characters.
- **location_id** (*int*) – Location ID
- **fqdn** (*str*) – Fully qualified domain name of the node
- **use_ssl** (*bool*) – True to enable SSL, false for insecure HTTP
- **behind_proxy** (*bool*) – Sets the node's behind_proxy
- **maintenance_mode** (*bool*) – Set the node's maintenance_mode
- **memory** (*int*) – Total memory available for the node in MB
- **memory_overallocate** (*int*) – A percentage to overallocate, e.g. 20 for 120%
- **disk** (*int*) – Total disk available for the node in MB
- **disk_overallocate** (*int*) – A percentage to overallocate, e.g. 20 for 120%
- **upload_size** (*int*) – Maximum size of uploads in file manager in MB
- **daemon_sftp** (*int*) – Node's SFTP port (default 2022)
- **daemon_listen** (*int*) – Wings listen port (default 8080)

get_node_config(*node_id*)
Get the Wings configuration for the specified node.

Parameters **node_id** (*int*) – Pterodactyl Node ID.

get_node_details(*node_id*, *includes=None*, *params=None*)
Get detailed info for the specified node.

Parameters

- **node_id** (*int*) – Pterodactyl Node ID.

- **includes** (*iter*) – List of includes, e.g. ('allocations', 'servers')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

get_node_info(*node_id*)
DEPRECATED: Use get_node_details

list_node_allocations(*node_id*, *includes=None*, *params=None*)
Retrieves all allocations for a specified node.

Parameters

- **node_id** (*int*) – Pterodactyl Node ID.
- **includes** (*iter*) – List of includes, e.g. ('node', 'server')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

Returns Iterable response that fetches pages as required.

Return type obj

list_nodes(*includes=None*, *params=None*)
List all nodes.

Parameters

- **includes** (*iter*) – List of includes, e.g. ('allocations', 'servers')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

3.4 servers

class pydactyl.api.servers.**Servers**
Class for interacting with the Pterdactyl Servers API.

create_server(*name*, *user_id*, *nest_id*, *egg_id*, *memory_limit*, *swap_limit*, *disk_limit*, *location_ids=[]*, *port_range=[]*, *environment={}*, *cpu_limit=0*, *io_limit=500*, *database_limit=0*, *allocation_limit=0*, *backup_limit=0*, *docker_image=None*, *startup_cmd=None*, *dedicated_ip=False*, *start_on_completion=True*, *oom_disabled=True*, *default_allocation=None*, *additional_allocations=None*, *external_id=None*, *description=None*)
Creates one or more servers in the specified locations.

Creates server instance(s) and begins the install process using the specified configurations and limits. If more than one value is specified for location_ids then identical servers will be created in each location.

Parameters

- **name** (*str*) – Name of the server to display in the panel.
- **user_id** (*int*) – User ID that will own the server.
- **nest_id** (*int*) – Nest ID for the created server.
- **egg_id** (*int*) – Egg ID for the created server.
- **memory_limit** (*int*) – Memory limit in MB for the Docker container. To allow unlimited memory limit set to 0.
- **swap_limit** (*int*) – Swap limit in MB for the Docker container. To not assign any swap set to 0. For unlimited swap set to -1.

- **disk_limit** (*int*) – Disk limit in MB for the Docker container. To allow unlimited disk space set to 0.
- **environment** (*dict*) – Key value pairs of Service Variables to set. Every variable from the egg must be set or the API will return an error. Default values will be pulled from the egg config or set to None.
- **location_ids** (*iter*) – List of location_ids where the server(s) will be created. If more than one location is specified identical servers will be created at each.
- **port_range** (*iter*) – List of ports or port ranges to use when selecting an allocation. If empty, all ports will be considered. If set, only ports appearing in the list or range will be used. e.g. [20715, '20815-20915']
- **cpu_limit** (*int*) – CPU limit for the Docker container. To allow unlimited CPU usage set to 0. To limit to one core set to 100. For four cores set to 400.
- **io_limit** (*int*) – Block IO weight for the Docker container. Must be between 10 and 1000.
- **database_limit** (*int*) – Maximum number of databases that can be assigned to this server.
- **allocation_limit** (*int*) – Maximum number of allocations that can be assigned to this server.
- **backup_limit** (*int*) – Maximum number of backups that can be created for this server.
- **docker_image** (*str*) – Name or URL of the Docker server to use. e.g. quay.io/pterodactyl/core:java-glibc
- **startup_cmd** (*str*) – Startup command, if specified replaces the egg's default value.
- **dedicated_ip** (*bool*) – Limit allocations to IPs without any existing allocations.
- **start_on_completion** (*bool*) – Start server after install completes.
- **oom_disabled** (*bool*) – Disables OOM-killer on the Docker container.
- **default_allocation** (*int*) – Specify allocation(s) instead of using the Pterodactyl deployment service. Uses the allocation's internal ID and not the port number.
- **additional_allocations** (*iter*) – Additional allocations on top of default_allocation.
- **description** (*str*) – A description of the server if needed

create_server_database(*server_id*)

Create a database for the specified server.

Parameters **server_id** (*int*) – Pterodactyl Server ID.

delete_server(*server_id*, *force=False*)

Delete the server with the specified internal ID.

Attempts to delete the server from both the panel and daemon. By default if either one reports an error the action will be cancelled.

Parameters

- **server_id** (*int*) – Pterodactyl Server ID.
- **force** (*bool*) – If True the delete action will continue if the panel or daemon reports an error.

delete_server_database(*server_id*, *database_id*)
Delete the specified database for the specified server.

Parameters

- **server_id** (*int*) – Pterodactyl Server ID.
- **database_id** (*int*) – Database ID for specified server.

get_server_database_info(*server_id*, *database_id*, *detail=False*, *includes=None*, *params=None*)
Get information about the specified database on the specified server.

Parameters

- **server_id** (*int*) – Pterodactyl Server ID.
- **database_id** (*int*) – Database ID for specified server.
- **detail** (*bool*) – If True includes the object type and a nested data structure.
- **includes** (*iter*) – List of includes, e.g. ('password', 'host')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

get_server_info(*server_id=None*, *external_id=None*, *detail=False*, *includes=None*, *params=None*)
Get detailed info for the specified server.

Parameters

- **server_id** (*int*) – Pterodactyl Server ID.
- **external_id** (*int*) – Server ID from an external system like WHMCS
- **detail** (*bool*) – If True includes created and updated timestamps.
- **includes** (*iter*) – List of includes, e.g. ('egg', 'allocations')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

list_server_databases(*server_id*, *includes=None*, *params=None*)
List the database servers assigned to the specified server ID.

Parameters

- **server_id** (*int*) – Pterodactyl Server ID.
- **includes** (*iter*) – List of includes, e.g. ('password', 'host')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

list_servers(*includes=None*, *params=None*)
List all servers.

Parameters

- **includes** (*iter*) – List of includes, e.g. ('allocations', 'node')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

rebuild_server(*server_id*)
Rebuild the server with the specified internal ID.

Parameters **server_id** (*int*) – Pterodactyl Server ID.

reinstall_server(*server_id*)
Reinstall the server with the specified internal ID.

Parameters **server_id** (*int*) – Pterodactyl Server ID.

reset_server_database_password(*server_id*, *database_id*)
Resets the password for the specified server database.

Parameters

- **server_id** (*int*) – Pterodactyl Server ID.
- **database_id** (*int*) – Database ID for specified server.

suspend_server(*server_id*)
Suspend the server with the specified internal ID.

Parameters **server_id** (*int*) – Pterodactyl Server ID.

unsuspend_server(*server_id*)
Suspend the server with the specified internal ID.

Parameters **server_id** (*int*) – Pterodactyl Server ID.

update_server_build(*server_id*, *allocation_id*, *memory_limit=None*, *swap_limit=None*, *disk_limit=None*, *cpu_limit=None*, *io_limit=None*, *database_limit=None*, *allocation_limit=None*, *backup_limit=None*, *add_allocations=None*, *remove_allocations=None*, *oom_disabled=None*)
Updates the build configuration for an existing server.

Modifies an existing server identified by allocation_id and updates any parameters that are passed.

*** WARNING *** This endpoint has a lot of requirements and it doesn't always surface helpful errors. Sometimes they're in the panel logs. I plan to automate some of the painful parts so you can specify only the fields you want to update, however currently you must satisfy the API's requirements by passing in everything.

Example of a working set of parameters:

update_server_build(server_id=12, allocation_id=81, memory_limit=2048, swap_limit=2048, disk_limit=5120, cpu_limit=100, io_limit=500, database_limit=1, allocation_limit=2, backup_limit=4, add_allocations=None, remove_allocations=None, oom_disabled=True)

Parameters

- **server_id** (*int*) – Internal server ID, e.g. 12
- **allocation_id** (*int*) – Base allocation of the server to modify.
- **memory_limit** (*int*) – Memory limit in MB for the Docker container. To allow unlimited memory limit set to 0.
- **swap_limit** (*int*) – Swap limit in MB for the Docker container. To not assign any swap set to 0. For unlimited swap set to -1.
- **disk_limit** (*int*) – Disk limit in MB for the Docker container. To allow unlimited disk space set to 0.
- **cpu_limit** (*int*) – CPU limit for the Docker container. To allow unlimited CPU usage set to 0. To limit to one core set to 100. For four cores set to 400.
- **io_limit** (*int*) – Block IO weight for the Docker container. Must be between 10 and 1000.
- **database_limit** (*int*) – Maximum number of databases that can be assigned to this server.

- **allocation_limit** (*int*) – Maximum number of allocations that can be assigned to this server.
- **backup_limit** (*int*) – Maximum number of backups that can be assigned to this server.
- **add_allocations** (*iter*) – List of allocation IDs to add to the server.
- **remove_allocations** (*iter*) – List of allocation IDs to remove from the server.
- **oom_disabled** (*bool*) – Disables OOM-killer on the Docker container.

update_server_details(*server_id*, *name*, *user_id*, *external_id=None*, *description=None*)
Updates the details of an existing server.

Modifies an existing server details identified by its Pterodactyl id.

Example of a working set of parameters: update_server_details(server_id=10, name='My awesome Server', external_id='some_id2389234', description='This is really an awesome server !'

Parameters

- **server_id** (*int*) – Internal server ID, e.g. 12
- **name** (*str*) – Name of the server to display in the panel.
- **user_id** (*int*) – User ID that will own the server.
- **external_id** (*int*) – Server ID from an external system like WHMCS
- **description** (*str*) – A description of the server if needed

update_server_startup(*server_id*, *egg_id=None*, *environment={}*, *docker_image=None*, *startup_cmd=None*, *skip_scripts=None*)
Updates the startup config for the specified server.

Modifies the startup config of an existing server replacing any specified values. Unspecified values will not be changed.

Parameters

- **egg_id** (*int*) – Egg ID to update on the server.
- **environment** (*dict*) – Key value pairs of Service Variables to set. Every variable from the egg must be set or the API will return an error. Any keys specified will be overwritten in the existing environment list. Unspecified keys will not be modified. Extra keys will be dropped.
- **docker_image** (*str*) – Name or URL of the Docker server to use. e.g. quay.io/pterodactyl/core:java-glibc
- **startup_cmd** (*str*) – Startup command, if specified replaces the egg's default value.
- **skip_scripts** (*bool*) – True to skip egg scripts.

3.5 user

class pydactyl.api.user.**User**
Class for interacting with the Pterdactyl Client API.

create_user(*username*, *email*, *first_name*, *last_name*, *external_id=None*, *password=None*, *root_admin=False*, *language='en'*)
Creates a primary user on the Pterodactyl panel.

Parameters

- **username** (*str*) – Username, must be unique, required.
- **email** (*str*) – Full email, required.
- **first_name** (*str*) – User's first name, required.
- **last_name** (*str*) – User's last name, required.
- **external_id** (*str*) – User ID from external system like WHMCS.
- **password** (*str*) – Optionally specify a password. If not specified user will receive an email to setup their password.
- **root_admin** (*bool*) – Whether the account is an admin.
- **language** (*str*) – Used to specify the default language for an account.

delete_user(*user_id=None*, *external_id=None*, *detail=True*)
Deletes the specified user.

Will look up by external_id if provided. This action cannot be reversed so use with care.

Parameters

- **user_id** (*int*) – Pterodactyl user ID
- **external_id** (*int*) – User ID from external system like WHMCS
- **detail** (*bool*) – If True includes created and updated timestamps.

edit_user(*user_id*, *username*, *email*, *first_name*, *last_name*, *external_id=None*, *password=None*, *root_admin=False*, *language='en'*)
Edits an existing user on the Pterodactyl panel.

The user_id is used to find the existing user. All other parameters will be updated on that user_id.

Parameters

- **user_id** (*int*) – Pterodactyl User ID for the user to update.
- **username** (*str*) – Username, must be unique, required.
- **email** (*str*) – Full email, required.
- **first_name** (*str*) – User's first name, required.
- **last_name** (*str*) – User's last name, required.
- **external_id** (*str*) – User ID from external system like WHMCS.
- **password** (*str*) – Optionally specify a password. If not specified user will receive an email to setup their password.
- **root_admin** (*bool*) – Whether the account is an admin.
- **language** (*str*) – Used to specify the default language for an account.

get_user_info(*user_id=None*, *external_id=None*, *detail=True*, *includes=None*, *params=None*)
List detailed user information for specified user_id.

Parameters

- **user_id** (*int*) – Pterodactyl user ID

- **external_id** (*int*) – User ID from external system like WHMCS
- **detail** (*bool*) – If True includes created and updated timestamps.
- **includes** (*iter*) – List of includes, e.g. ('servers')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

list_users(*email=None*, *uuid=None*, *username=None*, *external_id=None*, *includes=None*, *params=None*)

List all users.

Accepts optional filter parameters. If multiple filters are specified only results that match all filters will be returned.

Parameters

- **email** (*str*) – Filter by email
- **uuid** (*str*) – Filter by uuid
- **username** (*str*) – Filter by username
- **external_id** (*int*) – Filter by external_id
- **includes** (*iter*) – List of includes, e.g. ('servers')
- **params** (*dict*) – Extra parameters to pass, e.g. {'per_page': 300}

CHAPTER 4

Indices and tables

- genindex
- modindex

Python Module Index

p

Index

E

F

G

L

N

P

R

S

U

W