

---

**dathost**

***Release 1.0.1***

**WardPearce**

**Oct 03, 2021**



# CONTENTS

<b>1</b>	<b>Install</b>	<b>3</b>
<b>2</b>	<b>Documentation Contents</b>	<b>5</b>
2.1	Intro . . . . .	5
2.1.1	Non-context managers . . . . .	5
2.1.2	Context managers . . . . .	6
2.2	Examples . . . . .	6
2.2.1	Creating a server . . . . .	6
2.2.2	Creating a match . . . . .	7
2.3	API . . . . .	7
2.3.1	Awaiting . . . . .	7
2.3.2	Blocking . . . . .	12
2.4	Settings . . . . .	16
2.4.1	ServerSettings . . . . .	16
2.4.2	MatchSettings . . . . .	18
2.5	Models . . . . .	19
2.5.1	Account . . . . .	19
2.5.2	Backup . . . . .	20
2.5.3	File . . . . .	20
2.5.4	Match . . . . .	21
2.5.5	Metric . . . . .	23
2.5.6	Server . . . . .	24
2.6	Exceptions . . . . .	28
<b>3</b>	<b>Indices and tables</b>	<b>29</b>
	<b>Index</b>	<b>31</b>



This is a unofficial asynchronous & synchronous wrapper for Dathost's API.

**Features:**

- Full API coverage.
- Asynchronous & synchronous support.
- Easy to use with an object oriented design.
- 100% test coverage.



## INSTALL

Pip:

```
pip3 install dathost
```

Git:

```
pip3 install git+https://github.com/WardPearce/dathost.git
```





## DOCUMENTATION CONTENTS

### 2.1 Intro

This wrapper has both asynchronous & synchronous support, this intro will cover the basic of both. Luckily for you the API for asynchronous (awaiting) & synchronous (blocking) is identical.

#### 2.1.1 Non-context managers

##### Awaiting client

```
import dathost

client = dathost.Awaiting(
    email="wardpearce@protonmail.com",
    password="..."
)

# A client should always be closed after being used!
await client.close()
```

##### Blocking client

```
import dathost

client = dathost.Blocking(
    email="wardpearce@protonmail.com",
    password="..."
)

# A client should always be closed after being used!
client.close()
```

## 2.1.2 Context managers

### Blocking

```
import dathost

with dathost.Blocking(EMAIL, PASSWORD) as client:
    pass
```

### Awaiting

```
import dathost

async with dathost.Awaiting(EMAIL, PASSWORD) as client:
    pass
```

## 2.2 Examples

Here are some simple examples on how to use this wrapper. This is written using the blocking wrapper, but still applies to the awaiting wrapper.

Assume that “client” has been initialized.

### 2.2.1 Creating a server

```
from dathost.settings import ServerSettings

data, server = client.create_server(
    ServerSettings(
        name="CS: GO server",
        location="sydney",
    ).csgo(
        slots=5,
        game_token="",
        tickrate=128,
        rcon_password=""
    )
)

server.start()
print(data.slots)
```

## 2.2.2 Creating a match

```

from dathost.settings import MatchSettings

data, match = server.create_match(
    MatchSettings(
        connection_time=60,
    ).team_1(
        [
            "[U:1:116962485]",
            76561198017567105,
            "STEAM_0:1:186064092"
        ]
    ).team_2(
        [
            "[U:1:320762620]",
            "STEAM_0:1:83437164",
            76561198214871324
        ]
    ).spectators(
        [
            "[U:1:320762620]",
            "STEAM_0:1:83437164",
            76561198214871324
        ]
    )
)

# Don't worry about steam IDs, the wrapper
# ensures they're all converted correctly.

```

## 2.3 API

### 2.3.1 Awaiting

**class** dathost.Awaiting

**await** **account()** → *dathost.models.account.AccountModel*

Gets account details

**Returns** Holds data on a account.

**Return type** *AccountModel*

**await** **close()** → None

Closes sessions

**await** **create\_server**(*settings: dathost.settings.ServerSettings*) →

*Tuple[dathost.models.server.ServerModel, dathost.server.awaiting.ServerAwaiting]*

Creates a new server.

**Parameters** **settings** (*ServerSettings*) – Used to configure server.

**Returns**

- *ServerModel* – Holds data on server.
- *ServerAwaiting* – Used to interact with the created server.

**async for ... in domains()** → AsyncGenerator[str, None]

Used to list domains.

**Returns** List of domains.

**Return type** list

**match(match\_id: str)** → *dathost.match.awaiting.AwaitingMatch*

Used to interact with a match.

**Parameters** **match\_id** (str) – Dathost Match ID.

**Returns**

**Return type** *AwaitingMatch*

**server(server\_id: str)** → *dathost.server.awaiting.ServerAwaiting*

Used for interacting with a server.

**Parameters** **server\_id** (str) – Dathost server ID.

**Returns** Used to interact with the server.

**Return type** *ServerAwaiting*

**async for ... in servers()** → AsyncGenerator[Tuple[*dathost.models.server.ServerModel*,  
*dathost.server.awaiting.ServerAwaiting*], None]

Used to list servers.

**Yields** *ServerModel* – Holds data on server.

## Server

**class** *dathost.server.awaiting.ServerAwaiting*

**backup(backup\_name: str)** → *dathost.server.awaiting.backup.AwaitingBackup*

Used to interact with a backup.

**Parameters** **backup\_name** (str) – Name of backup.

**Returns**

**Return type** *AwaitingBackup*

**async for ... in backups()** → AsyncGenerator[Tuple[*dathost.models.backup.BackupModel*,  
*dathost.server.awaiting.backup.AwaitingBackup*], None]

Used to list backups a server has.

**Yields**

- *BackupModel* – Holds details on backup.
- *AwaitingBackup* – Used for interacting with a backup.

**await console\_retrieve(lines: int = 1000)** → list

Used to retrieve lines from the console.

**Parameters** **lines** (int, optional) – Amount of lines to retrieve, by default 1000

**Returns** List of strings.

**Return type** list

**Raises** *InvalidConsoleLine* – Raised when console lines below 1 or above 100000.

**await console\_send**(*line: str*) → None

Used to send a rcon command to console.

**Parameters** *line* (*str*) – Console command.

**await create\_match**(*match\_settings: dathost.settings.MatchSettings*) →

Tuple[*dathost.models.match.MatchModel*, *dathost.match.awaiting.AwaitingMatch*]

Creates a match.

**Parameters** *match\_settings* (*MatchSettings*) – Holds details on the match.

**Returns**

- *MatchModel* – Holds match details.
- *AwaitingMatch* – Used to interact with a match.

**await delete**() → None

Used to delete a sever.

**await duplicate**(*sync: bool = False*) → Tuple[*dathost.models.server.ServerModel*,  
*dathost.server.awaiting.ServerAwaiting*]

Used to duplicate a server.

**Parameters** *sync* (*bool*) – Used to force update server cache, by default False

**Returns**

- *ServerModel* – Holds server data.
- *ServerAwaiting* – Used to interact with server.

**file**(*pathway: str*) → *dathost.server.awaiting.file.AwaitingFile*

Used to interact with a file on the server.

**Parameters** *pathway* (*str*) – Pathway of file on server.

**Returns**

**Return type** *AwaitingFile*

**async for ... in files**(*hide\_default: bool = False*, *path: Optional[str] = None*, *file\_sizes: bool = False*,  
*deleted\_files: bool = False*) →

AsyncGenerator[Tuple[*dathost.models.file.FileModel*,  
*dathost.server.awaiting.file.AwaitingFile*], None]

Used to list files.

**Parameters**

- *hide\_default* (*bool*, *optional*) – by default False
- *path* (*str*, *optional*) – Path to use as root, by default None
- *file\_sizes* (*bool*, *optional*) – by default False
- *deleted\_files* (*bool*, *optional*) – Include deleted files in list, by default False

**Yields**

- *FileModel* – Holds details on a file.
- *AwaitingFile* – Used to interact with a file.

**await ftp\_reset()** → None  
Resets the FRP password.

**await get()** → *dathost.models.server.ServerModel*  
Used to get details on server.

**Returns** Holds data on server.

**Return type** *ServerModel*

**await metrics()** → *dathost.models.metrics.MetricsModel*  
Used to get server metrics.

**Returns** Holds details on server metrics.

**Return type** *MetricsModel*

**await reset()** → None  
Used to reset the server.

**await start(allow\_host\_reassignment: bool = True)** → None  
Used to start the server.

**Parameters** **allow\_host\_reassignment** (*bool*, *optional*) – By default True

**await stop()** → None  
Used to stop the server.

**await sync()** → None  
Used to sync files from server to cache.

**await update(settings: dathost.settings.ServerSettings)** → None  
Update servers paramters.

**Parameters** **settings** (*ServerSettings*) – Used to configure server.

## Match

**class** dathost.match.awaiting.AwaitingMatch

**await get()** → *dathost.models.match.MatchModel*  
Gets details on a match

**Returns** Holds match details.

**Return type** *MatchModel*

## Backup

**class** dathost.server.awaiting.backup.AwaitingBackup

**await restore()** → None  
Used to restore a backup.

## File

**class** dathost.server.awaiting.file.**AwaitingFile**

**await delete()** → None

Deletes file.

**await download()** → bytes

Used to download a file into memory.

**Returns**

**Return type** bytes

### Notes

Its recommended to use `download_iterate` for large files.

**async for ... in download\_iterate()** → AsyncGenerator[bytes, None]

Asynchronously downloads data into memory.

**Yields** bytes

**await move(destination: str)** → None

Used for moving a file.

**Parameters destination (str)** –

### Notes

When called the `file_path` changes to the given destination.

**await save(local\_pathway: str)** → None

Saves file to local pathway.

**Parameters local\_pathway (str)** – Pathway to save file to.

**await unzip(destination: str)** → None

Used to unzip a file.

**Parameters destination (str)** –

**await upload(data: Optional[bytes] = None)** → None

Used for uploading raw bytes.

**Parameters data (bytes)** – Data to upload.

**await upload\_file(local\_pathway: str)** → None

Used to upload a local file.

**Parameters local\_pathway (str)** – Local file to upload.

## 2.3.2 Blocking

**class** dathost.Blocking

**account**() → *dathost.models.account.AccountModel*

Gets account details

**Returns** Holds data on a account.

**Return type** *AccountModel*

**close**() → None

Closes sessions

**create\_server**(*settings*: dathost.settings.ServerSettings) → Tuple[*dathost.models.server.ServerModel*,  
*dathost.server.blocking.ServerBlocking*]

Creates a new server.

**Parameters** **settings** (*ServerSettings*) – Used to configure server.

**Returns**

- *ServerModel* – Holds data on server.
- *ServerBlocking* – Used to interact with the created server.

**for ... in domains**() → Generator[str, None, None]

Used to list domains.

**Returns** List of domains.

**Return type** list

**match**(*match\_id*: str) → *dathost.match.blocking.BlockingMatch*

Used to interact with a match.

**Parameters** **match\_id** (*str*) – Dathost Match ID.

**Returns**

**Return type** *BlockingMatch*

**server**(*server\_id*: str) → *dathost.server.blocking.ServerBlocking*

Used for interacting with a server.

**Parameters** **server\_id** (*str*) – Dathost server ID.

**Returns** Used to interact with the server.

**Return type** *ServerBlocking*

**for ... in servers**() → Generator[Tuple[*dathost.models.server.ServerModel*,  
*dathost.server.blocking.ServerBlocking*], None, None]

Used to list servers.

**Yields**

- *ServerModel* – Holds data on server.
- *ServerBlocking* – Used to interact with server.



## Server

**class** dathost.server.blocking.**ServerBlocking**

**backup**(*backup\_name: str*) → *dathost.server.blocking.backup.BlockingBackup*

Used to interact with a backup.

**Parameters** **backup\_name** (*str*) – Name of backup.

**Returns**

**Return type** *BlockingBackup*

**for ... in backups**() → Generator[Tuple[*dathost.models.backup.BackupModel*,  
*dathost.server.blocking.backup.BlockingBackup*], None, None]

Used to list backups a server has.

**Yields**

- *BackupModel* – Holds details on backup.
- *Backup* – Used for interacting with a backup.

**console\_retrieve**(*lines: int = 1000*) → list

Used to retrieve lines from the console.

**Parameters** **lines** (*int*, *optional*) – Amount of lines to retrieve, by default 1000

**Returns** List of strings.

**Return type** list

**Raises** *InvalidConsoleLine* – Raised when console lines below 1 or above 100000.

**console\_send**(*line: str*) → None

Used to send a command to console.

**Parameters** **line** (*str*) – Console command.

**create\_match**(*match\_settings: dathost.settings.MatchSettings*) →

Tuple[*dathost.models.match.MatchModel*, *dathost.match.blocking.BlockingMatch*]

Creates a match.

**Parameters** **match\_settings** (*MatchSettings*) – Holds details on the match.

**Returns**

- *MatchModel* – Holds match details.
- *BlockingMatch* – Used to interact with a match.

**delete**() → None

Used to delete a sever.

**duplicate**(*sync: bool = False*) → Tuple[*dathost.models.server.ServerModel*,  
*dathost.server.blocking.ServerBlocking*]

Used to duplicate a server.

**Parameters** **sync** (*bool*) – Used to force update server cache, by default False

**Returns**

- *ServerModel* – Holds server data.
- *ServerBlocking* – Used to interact with server.

**file**(*pathway: str*) → *dathost.server.blocking.file.BlockingFile*

Used to interact with a file on the server.

**Parameters** *pathway* (*str*) – Pathway of file on server.

**Returns**

**Return type** *BlockingFile*

**for ... in files**(*hide\_default: bool = False, path: Optional[str] = None, file\_sizes: bool = False, deleted\_files: bool = False*) → Generator[Tuple[*dathost.models.file.FileModel*, *dathost.server.blocking.file.BlockingFile*], None, None]

Used to list files.

**Parameters**

- **hide\_default** (*bool, optional*) – by default False
- **path** (*str, optional*) – Path to use as root, by default None
- **file\_sizes** (*bool, optional*) – by default False
- **deleted\_files** (*bool, optional*) – Include deleted files in list, by default False

**Yields**

- *FileModel* – Holds details on a file.
- *BlockingFile* – Used to interact with a file.

**ftp\_reset**() → None

Resets the FRP password.

**get**() → *dathost.models.server.ServerModel*

Used to get details on server.

**Returns** Holds data on server.

**Return type** *ServerModel*

**metrics**() → *dathost.models.metrics.MetricsModel*

Used to get server metrics.

**Returns** Holds details on server metrics.

**Return type** *MetricsModel*

**reset**() → None

Used to reset the server.

**start**(*allow\_host\_reassignment: bool = True*) → None

Used to stop the server.

**Parameters** *allow\_host\_reassignment* (*bool, optional*) – By default True

**stop**() → None

Used to stop the server.

**sync**() → None

Used to sync files from server to cache.

**update**(*settings: dathost.settings.ServerSettings*) → None

Update servers paramters.

**Parameters** *settings* (*ServerSettings*) – Used to configure server.

## Match

**class** dathost.match.blocking.**BlockingMatch**

**get()** → *dathost.models.match.MatchModel*

Gets details on a match

**Returns** Holds match details.

**Return type** *MatchModel*

## Backup

**class** dathost.server.blocking.backup.**BlockingBackup**

**restore()** → None

Used to restore a backup.

## File

**class** dathost.server.blocking.file.**BlockingFile**

**delete()** → None

Deletes file.

**download()** → bytes

Used to download a file into memory.

**Returns**

**Return type** bytes

## Notes

Its recommended to use `download_iterate` for large files.

**download\_iterate()** → None

**Raises** *AwaitingOnly* – This function is meant only for awaiting code.

**move(destination: str)** → None

Used for moving a file.

**Parameters** **destination** (*str*) –

## Notes

When called the `file_path` changes to the given destination.

**save**(*local\_pathway: str*) → None

Saves file to local pathway.

**Parameters** `local_pathway (str)` – Pathway to save file to.

**unzip**(*destination: str*) → None

Used to unzip a file.

**Parameters** `destination (str)` –

**upload**(*data: bytes*) → None

Used for uploading raw bytes.

**Parameters** `data (bytes)` – Data to upload.

**upload\_file**(*local\_pathway: str*) → None

Used to upload a local file.

**Parameters** `local_pathway (str)` – Local file to upload.

## 2.4 Settings

### 2.4.1 ServerSettings

```
class dathost.settings.ServerSettings(name: Optional[str] = None, location: Optional[str] = None,  
                                     custom_domain: Optional[str] = None, autostop: Optional[bool]  
                                     = None, autostop_minutes: Optional[int] = None, mysql:  
                                     Optional[bool] = None, scheduled_commands: Optional[List[str]]  
                                     = None, user_data: Optional[str] = None, reboot_on_crash:  
                                     Optional[bool] = None, max_disk_usage_gb: Optional[int] =  
                                     None, manual_sort_order: Optional[int] = None, core_dump:  
                                     Optional[bool] = None, prefer_dedicated: Optional[bool] = None)
```

```
csgo(slots: Optional[int] = None, tickrate: Optional[int] = None, game_token: Optional[str] = None,  
      rcon_password: Optional[str] = None, game_mode: Optional[str] = None, autoload_configs:  
      Optional[List[str]] = None, disable_bots: bool = False, workshop_start_map_id: Optional[int] = None,  
      csay_plugin: bool = False, gotv: bool = False, sourcemod: bool = False, insecure: bool = False,  
      map_group: Optional[str] = None, start_map: Optional[str] = None, password: Optional[str] = None,  
      pure: bool = True, admins: Optional[List[Any]] = None, plugins: Optional[List[Any]] = None,  
      steam_key: Optional[str] = None, workshop_id: Optional[int] = None, maps_source: Optional[str] =  
      None) → dathost.settings.ServerSettings
```

Used for configuring a CS: GO server.

**Parameters**

- **slots** (*int*) –
- **game\_token** (*str*) –
- **tickrate** (*int*) –
- **game\_mode** (*str, optional*) – by default None
- **autoload\_configs** (*List[str], optional*) – by default None

- **disable\_bots** (*bool, optional*) – by default False
- **csay\_plugin** (*bool, optional*) – by default False
- **gotv** (*bool, optional*) – by default False
- **sourcemod** (*bool, optional*) – by default False
- **insecure** (*bool, optional*) – by default False
- **map\_group** (*str, optional*) – by default None
- **start\_map** (*str, optional*) – by default None
- **password** (*str, optional*) – by default None
- **pure** (*bool, optional*) – by default True
- **rcon\_password** (*str, optional*) – by default None
- **admins** (*List[Any], optional*) – by default None
- **plugins** (*List[Any], optional*) – by default None
- **steam\_key** (*str, optional*) – by default None
- **workshop\_id** (*int, optional*) – by default None
- **workshop\_start\_map\_id** (*int, optional*) – by default None
- **maps\_source** (*int, optional*) – by default None

#### Raises

- **MultipleGames** – Raised when you attempt to create one server with multiple games.
- **InvalidSlotSize** – Raised when slot size is below 5 or above 64.
- **InvalidTickrate** – Raised when tickrate is invalid.

#### Returns

Return type *ServerSettings*

**teamspeak**(*slots: int*) → *dathost.settings.ServerSettings*

Used for configuring a teamspeak server.

Parameters **slots** (*int*) –

#### Raises

- **MultipleGames** – Raised when you attempt to create one server with multiple games.
- **InvalidSlotSize** – Raised when slot size is below 5 or above 500.

#### Returns

Return type *ServerSettings*

**tf2**(*slots: Optional[int] = None, rcon\_password: Optional[str] = None, gotv: bool = False, sourcemod: bool = False, insecure: bool = False, password: Optional[str] = None, admins: Optional[list] = None*) → *dathost.settings.ServerSettings*

Used for configuring a TF2 server.

#### Parameters

- **rcon\_password** (*str*) –
- **slots** (*int*) –

- **gotv** (*bool, optional*) – by default False
- **sourcemod** (*bool, optional*) – by default False
- **insecure** (*bool, optional*) – by default False
- **password** (*str, optional*) – by default None
- **admins** (*list, optional*) – by default None

**Raises**

- **MultipleGames** – Raised when you attempt to create one server with multiple games.
- **InvalidSlotSize** – Raised when slot size is below 5 or above 32.

**Returns**

**Return type** *ServerSettings*

**valheim**(*password: Optional[str] = None, world\_name: Optional[str] = None, plus: Optional[bool] = None, admins: Optional[List[Any]] = None*) → *dathost.settings.ServerSettings*

Used to configure valheim server.

**Parameters**

- **password** (*str, optional*) – by default None
- **world\_name** (*str, optional*) – by default None
- **plus** (*bool, optional*) – by default None
- **admins** (*List[Any], optional*) – List of SteamIDs in any format, by default None

**Returns**

**Return type** *ServerSettings*

**Raises** **MultipleGames** –

## 2.4.2 MatchSettings

**class** *dathost.settings.MatchSettings*(*connection\_time: int = 300, knife\_round: bool = False, wait\_for\_spectators: bool = True, enable\_pause: bool = False, enable\_ready: bool = False, enable\_tech\_pause: bool = False, ready\_min\_players: int = 1, wait\_for\_coaches: bool = True, warmup\_time: int = 15*)

**playwin**(*webhook: Optional[str] = None*) → *dathost.settings.MatchSettings*

Enables playwin AC.

**Parameters** **webhook** (*str, optional*) – Webhook to push playwin results, by default None

**Returns**

**Return type** *MatchSettings*

**spectators**(*players: list*) → *dathost.settings.MatchSettings*

Spectators

**Parameters** **players** (*list*) – List of spectator steam IDs, steamID 64, 32 & u are supported.

**Returns**

**Return type** *MatchSettings*

**team\_1**(*players: list, coach: Optional[Union[str, int]] = None*) → *dathost.settings.MatchSettings*  
Team 1 players

**Parameters**

- **players** (*list*) – List of spectator steam IDs, steamID 64, 32 & u are supported.
- **coach** (*Union[str, int]*) – Steam id of coach, by default None

**Returns**

**Return type** *MatchSettings*

**team\_2**(*players: list, coach: Optional[Union[str, int]] = None*) → *dathost.settings.MatchSettings*  
Team 2 players

**Parameters**

- **players** (*list*) – List of spectator steam IDs, steamID 64, 32 & u are supported.
- **coach** (*Union[str, int]*) – Steam id of coach, by default None

**Returns**

**Return type** *MatchSettings*

**webhook**(*match\_end: str, round\_end: str, authorization: Optional[str] = None*) →  
*dathost.settings.MatchSettings*

Used to set webhooks.

**Parameters**

- **match\_end** (*str*) – URL of match end webhook.
- **round\_end** (*str*) – URL of round end webhook.
- **authorization** (*str, optional*) – by default None

**Returns**

**Return type** *MatchSettings*

## 2.5 Models

### 2.5.1 Account

**class** *dathost.models.account.AccountModel*

Holds information around a account.

**account\_id**

**Type** *str*

**email**

**Type** *str*

**gravatar\_url**

**Type** *str*

**credits**

**Type** *str*

**seconds\_left**  
    Type int

**time\_left**  
    Type int

**roles**  
    Type str

**trial**  
    Type bool

**accepted\_terms\_of\_service\_version**  
    Type int

**subscription\_pay\_with\_credits**  
    Type bool

**affiliate**  
    Type bool

**first\_month\_discount\_percentage**  
    Type float

**confirmed\_at**  
    Type datetime

## 2.5.2 Backup

**class** dathost.models.backup.**BackupModel**  
    Holds detail on backups.

**backup\_name**  
    Type str

**timestamp**  
    Type datetime.datetime

## 2.5.3 File

**class** dathost.models.file.**FileModel**  
    Used to hold details on file.

**path**  
    Type str

**size**  
    Type str, optional



## 2.5.4 Match

**class** dathost.models.match.MatchModel

Holds match details.

**match\_id**

Type str

**server\_id**

Type str

**connect\_time**

Type int

**round\_end\_webhook**

Type str

**match\_end\_webhook**

Type str

**finished**

Type bool

**cancel\_reason**

Type str

**rounds\_played**

Type int

**spectators**

Type list

**team\_1**

Type *TeamModel*

**team\_2**

Type *TeamModel*

**knife\_round**

Type bool

**playwin**

Type bool

**playwin\_webhook**

Type str

**playwin\_result**

Type dict

**warmup\_time**

Type int

**wait\_for\_spectators**

```
        Type bool
enable_pause
        Type bool
enable_ready
        Type bool
enable_tech_pause
        Type bool
team_1_coach
        Type str
team_2_coach
        Type str
wait_for_coaches
        Type bool
for ... in players() → Generator[dathost.models.match.MatchPlayerModel, None, None]
    Used to list players.
        Yields PlayerModel – Holds details on player.
class dathost.models.match.MatchPlayerModel
    Holds match player details.
    steamid
        Type str
    kills
        Type int
    deaths
        Type int
    assists
        Type int
    kdr
        Type float
class dathost.models.match.TeamModel
    Holds details on team.
    score
        Type int
    players
        Type list
```

## 2.5.5 Metric

**class** dathost.models.metrics.**MetricsModel**

**for ... in all\_time\_players()** → Generator[*dathost.models.metrics.PlayerModel*, None, None]  
Used to list all time players.

**Yields** *PlayerModel* – Holds details on online players.

**for ... in maps()** → Generator[*dathost.models.metrics.MapsModel*, None, None]  
Used to list all maps what have been played.

**Yields** *MapsModel* – Holds details on maps.

**for ... in players\_online()** → Generator[*dathost.models.metrics.PlayerModel*, None, None]  
Used to list all players online.

**Yields** *PlayerModel* – Holds details on online players.

**for ... in players\_online\_graph()** → Generator[*dathost.models.metrics.PlayersOnlineGraphModel*, None, None]

Used to list all players online graph.

**Yields** *PlayersOnlineGraphModel* – Holds details on online player times.

**class** dathost.models.metrics.**MapsModel**

Holds map details

**map**

**Type** str

**seconds**

**Type** int

**class** dathost.models.metrics.**PlayerModel**

Holds player details

**name**

**Type** str

**duration**

**Type** int

**score**

**Type** int

**class** dathost.models.metrics.**PlayersOnlineGraphModel**

Holds player graph details

**timestamp**

**Type** str

**value**

**Type** str

## 2.5.6 Server

**class** dathost.models.server.ServerModel

Holds details on server

**server\_id**

**Type** str

**name**

**Type** str

**user\_data**

**Type** str

**match\_id**

**Type** str

**game**

**Type** str

**location**

**Type** str

**players\_online**

**Type** int

**status**

**Type** list

**booting**

**Type** bool

**server\_error**

**Type** str

**ip**

**Type** str

**raw\_ip**

**Type** str

**on**

**Type** bool

**ports**

**Type** *PortsModel*

**confirmed**

**Type** bool

**cost\_per\_hour**

**Type** int

**max\_cost\_per\_hour**

Type int  
month\_credits  
    Type float  
month\_reset\_at  
    Type datetime  
max\_cost\_per\_month  
    Type float  
subscription\_cycle\_months  
    Type int  
subscription\_renewal\_failed\_attempts  
    Type int  
mysql  
    Type bool  
autostop  
    Type bool  
autostop\_minutes  
    Type int  
mysql\_username  
    Type str  
mysql\_password  
    Type str  
ftp\_password  
    Type str  
disk\_usage\_bytes  
    Type int  
default\_file\_locations  
    Type list  
custom\_domain  
    Type str  
added\_voice\_server  
    Type str  
duplicate\_source\_server  
    Type str  
teamspeak  
    Type *TeamspeakModel*  
teamfortress

```

        Type TeamFortressModel
valheim
        Type ValheimModel
csgo
        Type CsgoModel
max_disk_usage_gb
        Type int
reboot_on_crash
        Type bool
core_dump
        Type bool
prefer_dedicated
        Type bool
for ... in scheduled_commands() → Generator[dathost.models.server.ScheduledCommandsModel,
                                             None, None]
    Lists scheduled commands.

    Yields ScheduledCommandsModel – Holds data on scheduled commands.
class dathost.models.server.PortsModel
    Holds details on ports
    game
        Type int
    gotv
        Type int
class dathost.models.server.ScheduledCommandsModel
    Holds details on scheduled commands
    name
        Type str
    action
        Type str
    command
        Type str
    run_at
        Type str
    repeat
        Type bool
class dathost.models.server.TeamspeakModel
    Holds details on teamspeak server
    slots
```

```
        Type int
    admin_token
        Type str
class dathost.models.server.TeamFortressModel
    Holds details on tf2.
    slots
        Type int
    rcon
        Type str
    password
        Type str
    admins
        Type str
    gotv
        Type bool
    sourcemod
        Type bool
    insecure
        Type bool
class dathost.models.server.ValheimModel
    Holds details on valheim server
    admins
        Type List[Int]
    plus
        Type bool
    slots
        Type int
    password
        Type str
    world_name
        Type str
```

## 2.6 Exceptions

**class** dathost.exceptions.DathostException

Base exception for dathost.

**class** dathost.exceptions.InvalidSlotSize

Raised when slot size is invalid.

**class** dathost.exceptions.MultipleGames

Raised when you attempt to create one server with multiple games.

**class** dathost.exceptions.InvalidTickrate

Raised when tickrate is invalid.

**class** dathost.exceptions.InvalidConsoleLine

Raised when console line is above 1 or above 100000.

**class** dathost.exceptions.AwaitingOnly

Raised when a coroutine called is awaiting supported only.

**class** dathost.exceptions.InvalidSteamID

Raised when give ID isn't understood.

**class** dathost.exceptions.NotFound

Resource not found.

**class** dathost.exceptions.BadRequest

Path is a directory or Cannot move file into itself.

**class** dathost.exceptions.ExceededStorage

Your disk quota of 30GB per server (excluding base installation) has been exceeded

**class** dathost.exceptions.ServerStart

Failed to start server.



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## A

accepted\_terms\_of\_service\_version  
(*dathost.models.account.AccountModel* attribute), 20

account() (*dathost.Awaiting* method), 7

account() (*dathost.Blocking* method), 12

account\_id (*dathost.models.account.AccountModel* attribute), 19

AccountModel (class in *dathost.models.account*), 19

action (*dathost.models.server.ScheduledCommandsModel* attribute), 26

added\_voice\_server (*dathost.models.server.ServerModel* attribute), 25

admin\_token (*dathost.models.server.TeamspeakModel* attribute), 27

admins (*dathost.models.server.TeamFortressModel* attribute), 27

admins (*dathost.models.server.ValheimModel* attribute), 27

affiliate (*dathost.models.account.AccountModel* attribute), 20

all\_time\_players() (*dathost.models.metrics.MetricsModel* method), 23

assists (*dathost.models.match.MatchPlayerModel* attribute), 22

autostop (*dathost.models.server.ServerModel* attribute), 25

autostop\_minutes (*dathost.models.server.ServerModel* attribute), 25

Awaiting (class in *dathost*), 7

AwaitingBackup (class in *dathost.server.awaiting.backup*), 10

AwaitingFile (class in *dathost.server.awaiting.file*), 11

AwaitingMatch (class in *dathost.match.awaiting*), 10

AwaitingOnly (class in *dathost.exceptions*), 28

## B

backup() (*dathost.server.awaiting.ServerAwaiting* method), 8

backup() (*dathost.server.blocking.ServerBlocking* method), 13

backup\_name (*dathost.models.backup.BackupModel* attribute), 20

BackupModel (class in *dathost.models.backup*), 20

backups() (*dathost.server.awaiting.ServerAwaiting* method), 8

backups() (*dathost.server.blocking.ServerBlocking* method), 13

BadRequest (class in *dathost.exceptions*), 28

Blocking (class in *dathost*), 12

BlockingBackup (class in *dathost.server.blocking.backup*), 15

BlockingFile (class in *dathost.server.blocking.file*), 15

BlockingMatch (class in *dathost.match.blocking*), 15

booting (*dathost.models.server.ServerModel* attribute), 24

## C

cancel\_reason (*dathost.models.match.MatchModel* attribute), 21

close() (*dathost.Awaiting* method), 7

close() (*dathost.Blocking* method), 12

command (*dathost.models.server.ScheduledCommandsModel* attribute), 26

confirmed (*dathost.models.server.ServerModel* attribute), 24

confirmed\_at (*dathost.models.account.AccountModel* attribute), 20

connect\_time (*dathost.models.match.MatchModel* attribute), 21

console\_retrieve() (*dathost.server.awaiting.ServerAwaiting* method), 8

console\_retrieve() (*dathost.server.blocking.ServerBlocking* method), 13

console\_send() (*dathost.server.awaiting.ServerAwaiting* method), 9

console\_send() (*dathost.server.blocking.ServerBlocking* method), 13

core\_dump (*dathost.models.server.ServerModel* attribute), 26

cost\_per\_hour (*dathost.models.server.ServerModel* attribute), 24

create\_match() (*dathost.server.awaiting.ServerAwaiting*

method), 9  
 create\_match() (dathost.server.blocking.ServerBlocking method), 13  
 create\_server() (dathost.Awaiting method), 7  
 create\_server() (dathost.Blocking method), 12  
 credits (dathost.models.account.AccountModel attribute), 19  
 csgo (dathost.models.server.ServerModel attribute), 26  
 csgo() (dathost.settings.ServerSettings method), 16  
 custom\_domain (dathost.models.server.ServerModel attribute), 25

## D

DathostException (class in dathost.exceptions), 28  
 deaths (dathost.models.match.MatchPlayerModel attribute), 22  
 default\_file\_locations (dathost.models.server.ServerModel attribute), 25  
 delete() (dathost.server.awaiting.file.AwaitingFile method), 11  
 delete() (dathost.server.awaiting.ServerAwaiting method), 9  
 delete() (dathost.server.blocking.file.BlockingFile method), 15  
 delete() (dathost.server.blocking.ServerBlocking method), 13  
 disk\_usage\_bytes (dathost.models.server.ServerModel attribute), 25  
 domains() (dathost.Awaiting method), 8  
 domains() (dathost.Blocking method), 12  
 download() (dathost.server.awaiting.file.AwaitingFile method), 11  
 download() (dathost.server.blocking.file.BlockingFile method), 15  
 download\_iterate() (dathost.server.awaiting.file.AwaitingFile method), 11  
 download\_iterate() (dathost.server.blocking.file.BlockingFile method), 15  
 duplicate() (dathost.server.awaiting.ServerAwaiting method), 9  
 duplicate() (dathost.server.blocking.ServerBlocking method), 13  
 duplicate\_source\_server (dathost.models.server.ServerModel attribute), 25  
 duration (dathost.models.metrics.PlayerModel attribute), 23

## E

email (dathost.models.account.AccountModel attribute), 19  
 enable\_pause (dathost.models.match.MatchModel attribute), 22

enable\_ready (dathost.models.match.MatchModel attribute), 22  
 enable\_tech\_pause (dathost.models.match.MatchModel attribute), 22  
 ExceededStorage (class in dathost.exceptions), 28

## F

file() (dathost.server.awaiting.ServerAwaiting method), 9  
 file() (dathost.server.blocking.ServerBlocking method), 13  
 FileModel (class in dathost.models.file), 20  
 files() (dathost.server.awaiting.ServerAwaiting method), 9  
 files() (dathost.server.blocking.ServerBlocking method), 14  
 finished (dathost.models.match.MatchModel attribute), 21  
 first\_month\_discount\_percentage (dathost.models.account.AccountModel attribute), 20  
 ftp\_password (dathost.models.server.ServerModel attribute), 25  
 ftp\_reset() (dathost.server.awaiting.ServerAwaiting method), 9  
 ftp\_reset() (dathost.server.blocking.ServerBlocking method), 14

## G

game (dathost.models.server.PortsModel attribute), 26  
 game (dathost.models.server.ServerModel attribute), 24  
 get() (dathost.match.awaiting.AwaitingMatch method), 10  
 get() (dathost.match.blocking.BlockingMatch method), 15  
 get() (dathost.server.awaiting.ServerAwaiting method), 10  
 get() (dathost.server.blocking.ServerBlocking method), 14  
 gotv (dathost.models.server.PortsModel attribute), 26  
 gotv (dathost.models.server.TeamFortressModel attribute), 27  
 gravatar\_url (dathost.models.account.AccountModel attribute), 19

## I

insecure (dathost.models.server.TeamFortressModel attribute), 27  
 InvalidConsoleLine (class in dathost.exceptions), 28  
 InvalidSlotSize (class in dathost.exceptions), 28  
 InvalidSteamID (class in dathost.exceptions), 28  
 InvalidTickrate (class in dathost.exceptions), 28  
 ip (dathost.models.server.ServerModel attribute), 24

## K

`kdr` (*dathost.models.match.MatchPlayerModel* attribute), 22

`kills` (*dathost.models.match.MatchPlayerModel* attribute), 22

`knife_round` (*dathost.models.match.MatchModel* attribute), 21

## L

`location` (*dathost.models.server.ServerModel* attribute), 24

## M

`map` (*dathost.models.metrics.MapsModel* attribute), 23

`maps()` (*dathost.models.metrics.MetricsModel* method), 23

`MapsModel` (class in *dathost.models.metrics*), 23

`match()` (*dathost.Awaiting* method), 8

`match()` (*dathost.Blocking* method), 12

`match_end_webhook` (*dathost.models.match.MatchModel* attribute), 21

`match_id` (*dathost.models.match.MatchModel* attribute), 21

`match_id` (*dathost.models.server.ServerModel* attribute), 24

`MatchModel` (class in *dathost.models.match*), 21

`MatchPlayerModel` (class in *dathost.models.match*), 22

`MatchSettings` (class in *dathost.settings*), 18

`max_cost_per_hour` (*dathost.models.server.ServerModel* attribute), 24

`max_cost_per_month` (*dathost.models.server.ServerModel* attribute), 25

`max_disk_usage_gb` (*dathost.models.server.ServerModel* attribute), 26

`metrics()` (*dathost.server.awaiting.ServerAwaiting* method), 10

`metrics()` (*dathost.server.blocking.ServerBlocking* method), 14

`MetricsModel` (class in *dathost.models.metrics*), 23

`month_credits` (*dathost.models.server.ServerModel* attribute), 25

`month_reset_at` (*dathost.models.server.ServerModel* attribute), 25

`move()` (*dathost.server.awaiting.file.AwaitingFile* method), 11

`move()` (*dathost.server.blocking.file.BlockingFile* method), 15

`MultipleGames` (class in *dathost.exceptions*), 28

`mysql` (*dathost.models.server.ServerModel* attribute), 25

`mysql_password` (*dathost.models.server.ServerModel* attribute), 25

`mysql_username` (*dathost.models.server.ServerModel* attribute), 25

## N

`name` (*dathost.models.metrics.PlayerModel* attribute), 23

`name` (*dathost.models.server.ScheduledCommandsModel* attribute), 26

`name` (*dathost.models.server.ServerModel* attribute), 24

`NotFound` (class in *dathost.exceptions*), 28

## O

`on` (*dathost.models.server.ServerModel* attribute), 24

## P

`password` (*dathost.models.server.TeamFortressModel* attribute), 27

`password` (*dathost.models.server.ValheimModel* attribute), 27

`path` (*dathost.models.file.FileModel* attribute), 20

`PlayerModel` (class in *dathost.models.metrics*), 23

`players` (*dathost.models.match.TeamModel* attribute), 22

`players()` (*dathost.models.match.MatchModel* method), 22

`players_online` (*dathost.models.server.ServerModel* attribute), 24

`players_online()` (*dathost.models.metrics.MetricsModel* method), 23

`players_online_graph()` (*dathost.models.metrics.MetricsModel* method), 23

`PlayersOnlineGraphModel` (class in *dathost.models.metrics*), 23

`playwin` (*dathost.models.match.MatchModel* attribute), 21

`playwin()` (*dathost.settings.MatchSettings* method), 18

`playwin_result` (*dathost.models.match.MatchModel* attribute), 21

`playwin_webhook` (*dathost.models.match.MatchModel* attribute), 21

`plus` (*dathost.models.server.ValheimModel* attribute), 27

`ports` (*dathost.models.server.ServerModel* attribute), 24

`PortsModel` (class in *dathost.models.server*), 26

`prefer_dedicated` (*dathost.models.server.ServerModel* attribute), 26

## R

`raw_ip` (*dathost.models.server.ServerModel* attribute), 24

`rcon` (*dathost.models.server.TeamFortressModel* attribute), 27

`reboot_on_crash` (*dathost.models.server.ServerModel* attribute), 26

`repeat` (*dathost.models.server.ScheduledCommandsModel* attribute), 26

`reset()` (*dathost.server.awaiting.ServerAwaiting* method), 10

reset() (*dathost.server.blocking.ServerBlocking* method), 14  
 restore() (*dathost.server.awaiting.backup.AwaitingBackup* method), 10  
 restore() (*dathost.server.blocking.backup.BlockingBackup* method), 15  
 roles (*dathost.models.account.AccountModel* attribute), 20  
 round\_end\_webhook (*dathost.models.match.MatchModel* attribute), 21  
 rounds\_played (*dathost.models.match.MatchModel* attribute), 21  
 run\_at (*dathost.models.server.ScheduledCommandsModel* attribute), 26  
  
**S**  
 save() (*dathost.server.awaiting.file.AwaitingFile* method), 11  
 save() (*dathost.server.blocking.file.BlockingFile* method), 16  
 scheduled\_commands() (*dathost.models.server.ServerModel* method), 26  
 ScheduledCommandsModel (class in *dathost.models.server*), 26  
 score (*dathost.models.match.TeamModel* attribute), 22  
 score (*dathost.models.metrics.PlayerModel* attribute), 23  
 seconds (*dathost.models.metrics.MapsModel* attribute), 23  
 seconds\_left (*dathost.models.account.AccountModel* attribute), 19  
 server() (*dathost.Awaiting* method), 8  
 server() (*dathost.Blocking* method), 12  
 server\_error (*dathost.models.server.ServerModel* attribute), 24  
 server\_id (*dathost.models.match.MatchModel* attribute), 21  
 server\_id (*dathost.models.server.ServerModel* attribute), 24  
 ServerAwaiting (class in *dathost.server.awaiting*), 8  
 ServerBlocking (class in *dathost.server.blocking*), 13  
 ServerModel (class in *dathost.models.server*), 24  
 servers() (*dathost.Awaiting* method), 8  
 servers() (*dathost.Blocking* method), 12  
 ServerSettings (class in *dathost.settings*), 16  
 ServerStart (class in *dathost.exceptions*), 28  
 size (*dathost.models.file.FileModel* attribute), 20  
 slots (*dathost.models.server.TeamFortressModel* attribute), 27  
 slots (*dathost.models.server.TeamspeakModel* attribute), 26  
 slots (*dathost.models.server.ValheimModel* attribute), 27  
  
 sourcemod (*dathost.models.server.TeamFortressModel* attribute), 27  
 spectators (*dathost.models.match.MatchModel* attribute), 21  
 spectators() (*dathost.settings.MatchSettings* method), 18  
 start() (*dathost.server.awaiting.ServerAwaiting* method), 10  
 start() (*dathost.server.blocking.ServerBlocking* method), 14  
 status (*dathost.models.server.ServerModel* attribute), 24  
 steamid (*dathost.models.match.MatchPlayerModel* attribute), 22  
 stop() (*dathost.server.awaiting.ServerAwaiting* method), 10  
 stop() (*dathost.server.blocking.ServerBlocking* method), 14  
 subscription\_cycle\_months (*dathost.models.server.ServerModel* attribute), 25  
 subscription\_pay\_with\_credits (*dathost.models.account.AccountModel* attribute), 20  
 subscription\_renewal\_failed\_attempts (*dathost.models.server.ServerModel* attribute), 25  
 sync() (*dathost.server.awaiting.ServerAwaiting* method), 10  
 sync() (*dathost.server.blocking.ServerBlocking* method), 14  
  
**T**  
 team\_1 (*dathost.models.match.MatchModel* attribute), 21  
 team\_1() (*dathost.settings.MatchSettings* method), 18  
 team\_1\_coach (*dathost.models.match.MatchModel* attribute), 22  
 team\_2 (*dathost.models.match.MatchModel* attribute), 21  
 team\_2() (*dathost.settings.MatchSettings* method), 19  
 team\_2\_coach (*dathost.models.match.MatchModel* attribute), 22  
 teamfortress (*dathost.models.server.ServerModel* attribute), 25  
 TeamFortressModel (class in *dathost.models.server*), 27  
 TeamModel (class in *dathost.models.match*), 22  
 teamspeak (*dathost.models.server.ServerModel* attribute), 25  
 teamspeak() (*dathost.settings.ServerSettings* method), 17  
 TeamspeakModel (class in *dathost.models.server*), 26  
 tf2() (*dathost.settings.ServerSettings* method), 17

`time_left` (*dathost.models.account.AccountModel* attribute), 20  
`timestamp` (*dathost.models.backup.BackupModel* attribute), 20  
`timestamp` (*dathost.models.metrics.PlayersOnlineGraphModel* attribute), 23  
`trial` (*dathost.models.account.AccountModel* attribute), 20

## U

`unzip()` (*dathost.server.awaiting.file.AwaitingFile* method), 11  
`unzip()` (*dathost.server.blocking.file.BlockingFile* method), 16  
`update()` (*dathost.server.awaiting.ServerAwaiting* method), 10  
`update()` (*dathost.server.blocking.ServerBlocking* method), 14  
`upload()` (*dathost.server.awaiting.file.AwaitingFile* method), 11  
`upload()` (*dathost.server.blocking.file.BlockingFile* method), 16  
`upload_file()` (*dathost.server.awaiting.file.AwaitingFile* method), 11  
`upload_file()` (*dathost.server.blocking.file.BlockingFile* method), 16  
`user_data` (*dathost.models.server.ServerModel* attribute), 24

## V

`valheim` (*dathost.models.server.ServerModel* attribute), 26  
`valheim()` (*dathost.settings.ServerSettings* method), 18  
`ValheimModel` (class in *dathost.models.server*), 27  
`value` (*dathost.models.metrics.PlayersOnlineGraphModel* attribute), 23

## W

`wait_for_coaches` (*dathost.models.match.MatchModel* attribute), 22  
`wait_for_spectators` (*dathost.models.match.MatchModel* attribute), 21  
`warmup_time` (*dathost.models.match.MatchModel* attribute), 21  
`webhook()` (*dathost.settings.MatchSettings* method), 19  
`world_name` (*dathost.models.server.ValheimModel* attribute), 27