async-riot-api

Release 0.1

Princic-1837592

Jan 16, 2022

FIRST OF ALL

1 Get started **3**

2 API reference **5**

2.1 Quickstart with this library 5
2.2 Creating your API token 5
2.3 Installation 6
2.4 Interacting with the API 6
2.5 LoLAPI class 7
2.6 Implemented types 16

Python Module Index **31**

Index **33**

async-riot-api is a Python library for League of Legends (and other Riot Games' games) players that offers an async and simple-to-use implementation of the Riot Games APIs based on Asyncio.

CHAPTER

ONE

GET STARTED

- *Quick Start*: To get started quickly.
- *API Access*: To get an API token.
- *API Interaction*: To interact with the API.

CHAPTER

TWO

API REFERENCE

- *LoLAPI class*: Details about the main class for interacting with the API.
- *Types*: List of the implemented types.

2.1 Quickstart with this library

async-riot-api is a very intuitive and simple to use library. Follow these steps to get started:

- Get your API token following *this guide*
- Install the library in your (virtual) environment
- Open your favourite text editor and paste the following code:

```
from async_riot_api import LoLAPI
import asyncio

async def main():
    api = LoLAPI('token', 'region', 'routing value', True)
    me = await api.get_summoner('my summoner name')
    print(me.to_string())

asyncio.get_event_loop().run_until_complete(main())
```

- Replace values that need to be replaced
- Start using the API!

2.2 Creating your API token

To use the Riot Games API first you need to create a token. In order to do this:

- Go to this page and follow the steps to register a product
- Start registering for a personal API key, since you probably don't have a working prototype needed for a production key
- The approvation step could take days (or weeks) to finish, so i suggest you go here and use a development API key until you get a permanent key
- Development API key need to be refreshed every 24h, so remember to refresh yours or your application will stop working

2.3 Installation

No particular version of Python is required to install and use this library, the only important thing is that your Python version supports asyncio and the required libraries. Anyway, it's recommended to use the latest version of both Python and pip.

- Install Python from the official site (or use your package manager)
- Install pip following the instructions
- Install async-riot-api using pip

```
$ pip3 install -U async-riot-api
```

- Or install directly from GitHub

```
$ pip3 install -U git+https://github.com/Princic-1837592/async-riot-api
```

- Verify your installation

```
from async_riot_api import LoLAPI
```

2.4 Interacting with the API

All the interaction with the API is made using the *LoLAPI* class.

- First you create an object of type *LoLAPI*.
- Then you can start calling its methods, corresponding to api methods (or extra features like champion search)

Methods are async, meaning that you need to be in an async context to be able to use this library. Async methods mean that you can use the features offered by asyncio to enhance your performances.

For example, imagine you want to get rank information about the ten participants of a match. Without using asyncio features you would do something like this:

```
ranks = [await api.get_solo_league(p.summonerId) for p in match.participants]
```

This would take a relatively long time to finish, about more than 1.5s.

Now, let's use the function `gather` from asyncio:

```
ranks = await asyncio.gather(
    *[api.get_solo_league(p.summonerId) for p in match.participants]
)
```

This time the code would execute in far less time than before, thanks to `asyncio.gather` running all the coroutines in a parallel way. This way the time required to finish this call is the maximum time required by a single request, and not the sum of all requests.

2.5 LoLAPI class

On this page you will find the list of methods available for the *LoLAPI* class. Dispite the name, *LoLAPI* is not only for League of Legends, but includes all the methods from the original Riot Games API.

The majority of methods in this class are a direct implementation of Riot Games API methods, while the others are extras made to add features and (hopefully) simplify your experience with this library. Those that return the result of an API method always return an instance of a subclass of *RiotApiResponse*.

class async_riot_api.**LoLAPI**

Main class to interact with the API. Offers async methods corresponding to API methods and more.

It is important to notice that this implementation is exception-free, not meaning that it's impossible to make mistake, but meaning that errors returned by the API are not raised as exceptions. Instead, they are returned as *RiotApiError*, containing information about the error. To distinguish between a successful response and an error, you can easily use the object as a boolean expression:

```
summoner = await api.get_account_by_puuid(puuid)
if not summoner:
    print(f'Error with status code {summoner.status_code}: {summoner.message}')
else:
    print(summoner.to_string(sep = '|   '))
```

Another important thing to know is that some methods are simple functions, not coroutines, meaning that they only work with offline data without making any request. They are usually static methods.

Parameters

- **api_key** (*str*) – your API token
- **region** (*str*) – region you want to use
- **routing_value** (*str*) – one among 'AMERICA', 'ASIA', 'ESPORTS', 'EUROPE' or 'SEA. Needed for some API calls, depends on region
- **debug** (*bool*) – if you want the LoLAPI object to print the url of every request made

async get_account_by_puuid(*puuid: str*) → *async_riot_api.types.AccountDto*

To get an account given its puuid.

Original method.

Parameters **puuid** (*str*) – puuid of the account

Returns account data

Return type *AccountDto*

async get_account_by_game_name(*game_name: str*, *tag_line: str*) → *async_riot_api.types.AccountDto*

To get an account given its name and tag line.

Original method.

Parameters

- **game_name** (*str*) – in-game name of the account
- **tag_line** (*str*) – no documentation found

Returns account data

Return type *AccountDto*

async get_active_shards(*game: str*, *puuid: str*) → *async_riot_api.types.ActiveShardDto*
No documentation found.

Original method.

Parameters

- **game** (*str*) – one of 'val' or 'lor'
- **puuid** (*str*) – puuid of the account

Returns shard data

Return type *ActiveShardDto*

async get_masteries(*summoner_id: str*) → List[*async_riot_api.types.ChampionMasteryDto*]
Get the list of masteries for a summoner.

Original method.

Parameters **summoner_id** (*str*) – summoner ID

Returns list of masteries for the given summoner

Return type List[*ChampionMasteryDto*]

async get_champion_mastery(*summoner_id: str*, *champion_id: int*) → *async_riot_api.types.ChampionMasteryDto*
Get a specific champion mastery for the given summoner.

Original method.

Parameters

- **summoner_id** (*str*) –
- **champion_id** (*int*) –

Returns champion mastery for given summoner and champion

Return type *ChampionMasteryDto*

async get_mastery_score(*summoner_id: str*) → int
Get the total mastery score of a summoner. Mastery score is given by the sum of individual champion mastery levels.

Original method.

Parameters **summoner_id** (*str*) –

Returns mastery score of the given summoner

Return type int

async get_champion_rotation() → *async_riot_api.types.ChampionInfo*
Get champion rotations, including free-to-play and low-level free-to-play rotations.

Original method.

Returns information about champion rotations

Return type *ChampionInfo*

async get_clash_players_by_summoner_id(*summoner_id: str*) → List[*async_riot_api.types.PlayerDto*]
Get a list of active Clash players for a given summoner ID. If a summoner registers for multiple tournaments at the same time (e.g., Saturday and Sunday) then both registrations would appear in this list.

Original method.

Parameters **summoner_id** (*str*) –

Returns list of players

Return type List[*PlayerDto*]

async get_clash_team_by_id(*team_id: str*) → *async_riot_api.types.ClashTeamDto*
Get a clash team by its ID.

Original method.

Parameters **team_id** (*str*) –

Returns information about the team

Return type *TeamDto*

async get_clash_tournaments() → List[*async_riot_api.types.TournamentDto*]
Get all active or upcoming tournaments.

Original method.

Returns list of tournaments

Return type List[*TournamentDto*]

async get_clash_tournament_by_team_id(*team_id: str*) → *async_riot_api.types.TournamentDto*
Get tournament by team ID.

Original method.

Parameters **team_id** (*str*) –

Returns information about tournament

Return type *TournamentDto*

async get_clash_tournament_by_id(*tournament_id: int*) → *async_riot_api.types.TournamentDto*
Get info about a clash tournament by its ID.

Original method.

Parameters **tournament_id** (*int*) –

Returns information about the tournament

Return type *TournamentDto*

async get_summoners_by_league_exp(*queue: str*, *tier: str*, *division: str*, *page: int = 1*) →
Set[*async_riot_api.types.LeagueEntryDTO*]
This is an experimental (and personally untested) endpoint added as a duplicate of *get_summoners_by_league()*, but it also supports the apex tiers (Challenger, Grandmaster, and Master).

Original method..

Parameters

- **queue** (*str*) – one among 'RANKED_SOLO_5x5', 'RANKED_TFT', 'RANKED_FLEX_SR' or 'RANKED_FLEX_TT'
- **tier** (*str*) – rank tier, between 'IRON' and 'CHALLENGER'
- **division** (*str*) – rank division, between 'I' and 'IV' (in roman numbers)
- **page** (*int*) – page to select, starting from 1. Limited based on the number of entries, it's suggested to iter until results are found

Returns set of summoners for the requested queue, tier and division

Return type Set[*LeagueEntryDTO*]

async get_challenger_leagues(*queue: str*) → *async_riot_api.types.LeagueListDTO*
Get the list of challengers.

Original method.

Parameters **queue** (*str*) – one among 'RANKED_SOLO_5x5', 'RANKED_FLEX_SR' or 'RANKED_FLEX_TT'

Returns set of challengers

Return type *LeagueListDTO*

async get_league(*summoner_id: str*) → Set[*async_riot_api.types.LeagueEntryDTO*]
Get the set of league entries for a given summoner.

Original method.

Parameters **summoner_id** (*str*) –

Returns information about their ranks in every queue

Return type Set[*LeagueEntryDTO*]

async get_summoners_by_league(*queue: str*, *tier: str*, *division: str*, *page: int = 1*) → Set[*async_riot_api.types.LeagueEntryDTO*]
Get the list of summoners that are currently in the given rank of the given queue. Only supports non-apex tiers. To get information about apex tiers, look at *get_summoners_by_league_exp()*.

Original method.

Parameters

- **queue** (*str*) – one among 'RANKED_SOLO_5x5', 'RANKED_FLEX_SR' or 'RANKED_FLEX_TT'
- **tier** (*str*) – rank tier, between 'IRON' and 'DIAMOND'
- **division** (*str*) – rank division, between 'I' and 'IV' (in roman numbers)
- **page** (*int*) – page to select, starting from 1. Limited based on the number of entries, it's suggested to iter until results are found

Returns set of summoners for the requested queue, tier and division

Return type Set[*LeagueEntryDTO*]

async get_grand_master_leagues(*queue: str*) → *async_riot_api.types.LeagueListDTO*
Same as *get_challenger_leagues()*, but for grand masters.

Original method.

async get_leagues(*league_id: str*) → *async_riot_api.types.LeagueListDTO*
Get the list of summoners in the given league by its ID.

Original method.

Parameters **league_id** (*str*) –

Returns list of summoners currently in the given league

Return type *LeagueListDTO*

async get_master_leagues(*queue: str*) → *async_riot_api.types.LeagueListDTO*
Same as *get_challenger_leagues()*, but for masters.

Original method.

async get_platform_data_v3() → *async_riot_api.types.ShardStatus*
DEPRECATED Get information about LoL server status.

Original method.

Returns the current server status

Return type *ShardStatus*

async get_platform_data() → *async_riot_api.types.PlatformDataDto*
Get information about LoL server status.

Original method.

Returns the current LoL server status

Return type *PlatformDataDto*

async get_lor_matches(*puuid: str*) → List[str]
Get the list of LoR matches played by the given summoner. Often used before *get_lor_match()*.

Original method.

Parameters **puuid** (*str*) –

Returns list of match IDs sorted by recent

Return type List[str]

async get_lor_match(*match_id: str*) → *async_riot_api.types.LorMatchDto*
Get information about the given LoR match. Often used after *get_lor_matches()*.

Original method.

Parameters **match_id** (*str*) –

Returns useful information about the given LoR match and its players

Return type *LorMatchDto*

async get_lor_leaderboards() → *async_riot_api.types.LorLeaderboardDto*
Get the list of players in Master tier.

Original method.

Returns players in LoR Master tier

Return type *LorLeaderboardDto*

async get_lor_status() → *async_riot_api.types.PlatformDataDto*
Get information about LoR servers status.

Original method.

Returns the current LoR server status

Return type *PlatformDataDto*

async get_matches(*puuid: str*, ***, *startTime: Optional[int] = None*, *endTime: Optional[int] = None*, *queue: Optional[int] = None*, *type: Optional[str] = None*, *start: int = 0*, *count: int = 20*) → List[str]
Get the list of LoL matches played by the given summoner. Often used before *get_match()*. Allows filtering by start/end time, queue and type of match.

Original method.

Parameters

- **puuid** (*str*) –
- **startTime** (*int*) – epoch timestamp in seconds. The matchlist started storing timestamps on June 16th, 2021. Any matches played before June 16th, 2021 won't be included in the results if the startTime filter is set
- **endTime** (*int*) – epoch timestamp in seconds
- **queue** (*int*) – queue filter for the list. Queue IDs can be found here. Queue and type parameters are mutually inclusive
- **type** (*str*) – one among 'ranked', 'normal', 'tourney' or 'tutorial'
- **start** (*int*) – start index, starting from 0. Default 0
- **count** (*int*) – number of match IDs to return. Must be in the range 0-100. Default 20

Returns list of match IDs sorted by recent

Return type List[str]

async get_match(*match_id: str*) → *async_riot_api.types.MatchDto*

Get information about the given LoL match. Often used after *get_matches()*.

Original method.

Parameters **match_id** (*str*) –

Returns useful information about the given LoR match and its players

Return type *MatchDto*

async get_timeline(*match_id: str*) → *async_riot_api.types.MatchTimelineDto*

Get additional information about a match, ordered by time, organized in "frames". This kind of response contains information about items bought, skills unlocked, summoners position and more. Unfortunately on the original doc this method is not documented at all. Not even the return type is documented except for its name, so anything about this method comes from experimentation.

Original method.

Parameters **match_id** (*str*) –

Returns more data about the match, ordered by time

Return type *MatchTimelineDto*

async get_active_games(*summoner_id: str*) → *async_riot_api.types.CurrentGameInfo*

Get information about the active game played by the given summoner.

IMPORTANT This method returns a *RiotApiError* if the summoner is not in a game.

Original method.

Parameters **summoner_id** (*str*) –

Returns information about the current match and its players, if exists

Return type *CurrentGameInfo*

async get_featured_games() → *async_riot_api.types.FeaturedGames*

Get a list of games that are currently being played. It is not clear to me what are the criteria for a game to be listed here, and I haven't found anything on the documentation. Anyway, this method could be useful for those who need to harvest a large amount of data from real matches.

Original method.

Returns games that are currently being played

Return type *FeaturedGames*

async get_summoner_by_account_id(*account_id: str*) → *async_riot_api.types.SummonerDTO*

Get information about a summoner by its account ID. You can get an account ID using *get_summoner_by_name()*.

Original method.

Parameters **account_id** (*str*) –

Returns basic information about the summoner

Return type *SummonerDTO*

async get_summoner_by_name(*summoner_name: str*) → *async_riot_api.types.SummonerDTO*

Probably the first method you will need to call at the beginning of any program involving Riot Games API. This method allows accessing basic information about a summoner given its name. With this information you will be able to use any other method requiring a summoner ID, account ID or puuid.

Original method.

Parameters **summoner_name** (*str*) – name of the summoner you are looking for

Returns basic information about the summoner

Return type *SummonerDTO*

async get_summoner_by_puuid(*puuid: str*) → *async_riot_api.types.SummonerDTO*

Get information about a summoner by its puuid. You can get a puuid using *get_summoner_by_name()*.

Original method.

Parameters **puuid** (*str*) –

Returns basic information about the summoner

Return type *SummonerDTO*

async get_summoner_by_summoner_id(*summoner_id: str*) → *async_riot_api.types.SummonerDTO*

Get information about a summoner by its summoner ID. You can get a summoner ID using *get_summoner_by_name()*.

Original method.

Parameters **summoner_id** (*str*) –

Returns basic information about the summoner

Return type *SummonerDTO*

async get_nth_match(*puuid: str*, *n: int = 0*) → Optional[*async_riot_api.types.MatchDto*]

Directly get information about a summoner's match given its index, starting from 0. This is just a shorcut for *get_matches()* and *get_match()*.

Parameters

- **puuid** (*str*) – puuid of the summoner
- **n** (*int*) – index of the match, starting from 0. Default 0

Returns information about the match, if exists. None otherwise

Return type Optional[*MatchDto*]

async get_last_match(*puuid: str*) → Optional[*async_riot_api.types.MatchDto*]
Directly get information about a summoner's last match. This is just a shortcut for *get_nth_match()*.

Parameters **puuid** (*str*) –

Returns same as *get_nth_match()*

Return type Optional[*MatchDto*]

async get_solo_league(*summoner_id: str*) → Optional[*async_riot_api.types.LeagueEntryDTO*]
Directly get information about a summoner's SOLO rank.

Parameters **summoner_id** (*str*) –

Returns given summoner's SOLO rank, if exists.

Return type *LeagueEntryDTO*

async get_flex_league(*summoner_id: str*) → Optional[*async_riot_api.types.LeagueEntryDTO*]
Same as *get_solo_league()*, but FLEX.

Parameters **summoner_id** (*str*) –

Returns given summoner's FLEX rank, if exists.

Return type *LeagueEntryDTO*

static get_profile_icon_url(*icon_id: int*) → str
Returns the url to the given icon.

IMPORTANT: no check will be made about data existence, meaning that passing a wrong icon_id will simply result in a broken url. No error will be raised.

Parameters **icon_id** (*int*) –

Returns url to the icon

Return type str

static get_champion_image_url_from_id(*champ_id: int*, *skin: int = 0*, *type: str = 'splash'*) → str
Returns the url to the image for the given champion, skin and type.

IMPORTANT: no check will be made about data existence, meaning that passing a wrong champ_id, skin or type will simply result in a broken url. No error will be raised.

Parameters

- **champ_id** (*int*) – champion ID, corresponding to ShortChampionDD.int_id
- **skin** (*int*) – number of the requested skin, starting from 0 for the default skin. Default 0
- **type** (*str*) – type of image. Can be 'splash' or 'loading'. Default 'splash'

Returns url to the image

Return type str

static compute_champion_from_similar_name(*search_name: str*) → *async_riot_api.types.ShortChampionDD*
Computes the most similar champion to the given name. The similarity computation is made using this library.

Parameters **search_name** (*str*) – name to search

Returns champion whose name is the most similar to the given one

Return type *ShortChampionDD*

static compute_language(*search_language: str*) → str
Computes the most similar language available from this list. The similarity computation is made using this library.

Parameters **search_language** (*str*) – language to search

Returns most similar language

Return type str

static get_version() → int
Get the latest version of the game.

Returns latest version of the game

Return type int

static get_queue(*queue_id: int*) → *async_riot_api.types.QueueDD*
Get information about the given queue.

Parameters **queue_id** (*int*) – queue ID

Returns information about the queue

Return type *QueueDD*

static get_champion_from_correct_name(*name: str*) → Optional[*async_riot_api.types.ShortChampionDD*]
Get the short champion given its correct name. Useful instead of *compute_champion_from_similar_name()* if you already know the correct name.

Parameters **name** (*str*) – correct name of the champion, case-sensitive

Returns short information about the champion

Return type Optional[*ShortChampionDD*]

static get_champion_from_id(*champ_id: int*) → Optional[*async_riot_api.types.ShortChampionDD*]
Get the short champion given its ID. You can get a champ ID from many API calls.

Parameters **champ_id** (*int*) – integer champion ID, same as `ShortChampionDD.int_id`

Returns short champion

Return type Optional[*ShortChampionDD*]

async static get_full_champion_from_correct_name(*name: str*, *language: str = 'en'*) → *async_riot_api.types.ChampionDD*
Get the complete information about a champion given its correct name, in any available language. If the passed language is not present in this list, *compute_language()* is called.

Parameters

- **name** (*str*) – correct name of a champion, same as `ShortChampionDD.id`
- **language** (*str*) – any available language. Default 'en'

Returns full information about a champion

Return type *ChampionDD*

static get_map_icon_url(*map_id: int*) → str
Returns the url to the image for the given map. Map ID can be found in *MatchDto*. Complete list of map IDs here.

`IMPORTANT`: no check will be made about data existence, meaning that passing a wrong map_id will simply result in a broken url. No error will be raised.

Parameters **map_id** (*int*) – map ID

Returns url to the given map image

Return type str

2.6 Implemented types

class async_riot_api.types.**RiotApiResponse**(*success: bool = True*, ***kwargs*)

Bases: object

Superclass of all API responses.

Parameters **success** (*bool*) – wether the response was successful. Useful to spot errors

to_string(*, *level: int = 0*, *sep=' '*, *nl: str = '\n'*)

Returns a prettified string representation of the object.

Parameters

- **level** (*int*) – starting level of indentation. Default: 0
- **sep** (*str*) – character sequence for indentation. Default: 4 spaces
- **nl** (*str*) – new line sequence. Default 'n'

class async_riot_api.types.**RiotApiError**(*message: str = 'Bad Request'*, *status_code: int = 400*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

General API response error.

Parameters

- **message** (*str*) – message contained in the response. Default 'Bad Request'
- **status_code** (*int*) – error code from the response. Default 400

class async_riot_api.types.**ShortChampionDD**(*blurb: str*, *id: str*, *image: dict*, *info: dict*, *key: str*, *name: str*, *partype: str*, *stats: dict*, *tags: List[str]*, *title: str*, *version: str*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Short information about a champion

Parameters

- **blurb** (*str*) – short description
- **id** (*str*) – name of the champion without non-alphabetic characters
- **image** (*ChampionImageDD*) – information about the images of a champion
- **info** (*ChampionInfoDD*) – schematic information about the champion
- **key** (*str*) – unique key for a champion. For some reason this is originally a string, despite representing an integer
- **name** (*str*) – complete name of the champion
- **partype** (*str*) – type of energy used by the champion. Usually 'Mana' but could be 'Energy' or others
- **stats** (*ChampionStatsDD*) – statistics of the champion
- **tags** (*List[str]*) – tags about the champion, like 'Fighter', 'Mage'

- **title** (*str*) – short title of the champion
- **version** (*str*) – valid version for this object

Other attributes:

int_id (int): integer representation of param `ShortChampionDD.key`. Not present in the original data type, useful for some methods and, more importantly, coherent with the represented value

class async_riot_api.types.**ChampionDD**(*id: str*, *key: str*, *name: str*, *title: str*, *image: dict*, *skins: List[dict]*, *lore: str*, *blurb: str*, *allytips: List[str]*, *enemytips: List[str]*, *tags: List[str]*, *partype: str*, *info: dict*, *stats: dict*, *spells: List[dict]*, *passive: dict*, *recommended: list*, *version: str*, ***kwargs*)

Bases: *async_riot_api.types.ShortChampionDD*

Complete information about a champion.

Look at *ShortChampionDD* for the complete list of parameters.

Parameters

- **skins** (List[*ChampionSkinDD*]) – list of skins
- **lore** (*str*) – lore of the champion
- **allytips** (*List[str]*) – list of tips for summoners playing the champion
- **enemytips** (*List[str]*) – list of tipo for summoners playing against the champion
- **spells** (List[*ChampionSpellDD*]) – list of information about this champion's spells
- **passive** (*ChampionPassiveDD*) – information about this champion's passive ability
- **recommended** (*List[unknown]*) – no idea of what this is, haven't found any champion with a non-empty list of `recommended`

class async_riot_api.types.**ChampionImageDD**(*full: str*, *sprite: str*, *group: str*, *x: int*, *y: int*, *w: int*, *h: int*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Details about the champion's image.

Parameters

- **full** (*str*) – file name of the image. The complete url can be obtained from *get_champion_image_url_from_id()*
- **sprite** (*str*) – don't really know what this is, some kind of image with more images inside. You can find more info here
- **group** (*str*) – sub-category in which you can find the sprite of this image, more info in the same link as above
- **x** (*int*) – x coordinate of the sprite in which you can find this image
- **y** (*int*) – y coordinate of the sprite in which you can find this image
- **w** (*int*) – width of the image in the sprite, starting from coordinates (x, y)
- **h** (*int*) – height of the image in the sprite, starting from coordinates (x, y)

class async_riot_api.types.**ChampionSkinDD**(*id: str*, *num: int*, *name: str*, *chromas: bool*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Details about the champion's skins.

Parameters

- **id** (*str*) – unique id of the skin. It is made by concatenating the champ ID and the skin number (with 3 digits). Example: champion "Veigar" (ID 45), skin "Final Boss" (num 8), result: "45008"
- **num** (*int*) – number of the skin
- **name** (*str*) – name of the skin, including the champion name (if present)
- **chromas** (*bool*) – if the skin has got chromas

class async_riot_api.types.**ChampionInfoDD**(*attack: int*, *defense: int*, *magic: int*, *difficulty: int*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Schematic information about the champion. You can find this information in the LoL client by going to a champion's page.

Parameters

- **attack** (*int*) – the higher this value, the higher the champion deals damage using auto attacks
- **defense** (*int*) – the higher this value, the higher the champion is tanky
- **magic** (*int*) – the higher this value, the higher the champion deals damage using spells
- **difficulty** (*int*) – the higher this value, the more difficult is to master this champion

class async_riot_api.types.**ChampionStatsDD**(*hp: int*, *hpperlevel: int*, *mp: int*, *mpperlevel: int*, *movespeed: int*, *armor: int*, *armorperlevel: float*, *spellblock: int*, *spellblockperlevel: float*, *attackrange: int*, *hpregen: int*, *hpregenperlevel: int*, *mpregen: int*, *mpregenperlevel: int*, *crit: int*, *critperlevel: int*, *attackdamage: int*, *attackdamageperlevel: int*, *attackspeedperlevel: float*, *attackspeed: float*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Detailed information about a champion's base stats and how they increase when leveling up. Here i list their meanings and their unit of measurement where:

- `u` stands for "unit"
- `l` stands for "level"
- `s` stands for "second"

Parameters

- **hp** (*int*) – base health points (`u`)
- **hpperlevel** (*int*) – extra HP per level (`u / l`)
- **mp** (*int*) – base mana points (`u`)
- **mpperlevel** (*int*) – extra mana points per level (`u / l`)
- **movespeed** (*int*) – base movement speed (`u / s`)
- **armor** (*int*) – base armor (`u`)
- **armorperlevel** (*float*) – extra armor per level (`u / l`)
- **spellblock** (*int*) – base magic resistance (`u`)
- **spellblockperlevel** (*float*) – extra magic resistance per level (`u / l`)

- **attackrange** (*int*) – base attack range (`u`)
- **hpregen** (*int*) – base HP regeneration (`u / 5s`)
- **hpregenperlevel** (*int*) – extra HP regeneration per level (`u / 5s / l`)
- **mpregen** (*int*) – base mana regeneration (`u / 5s`)
- **mpregenperlevel** (*int*) – extra mana regeneration per level (`u / 5s / l`)
- **crit** (*int*) – base critical chance (`u`)
- **critperlevel** (*int*) – extra critical chance per level (`u / l`)
- **attackdamage** (*int*) – base attack damage (`u`)
- **attackdamageperlevel** (*int*) – extra attack damage per level (`u / l`)
- **attackspeedperlevel** (*float*) – extra attack speed per level (`u / s / l`)
- **attackspeed** (*float*) – base attack speed (`u / s`)

class async_riot_api.types.**ChampionSpellDD**(*id: str*, *name: str*, *description: str*, *tooltip: str*, *maxrank: int*, *cooldown: List[int]*, *cooldownBurn: str*, *cost: List[int]*, *costBurn: str*, *datavalues: dict*, *effect: List[Optional[List[int]]]*, *effectBurn: List[Optional[str]]*, *vars: List[Any]*, *costType: str*, *maxammo: str*, *range: List[int]*, *rangeBurn: str*, *image: dict*, *leveltip: Optional[dict] = None*, *resource: Optional[str] = None*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Specific information about a champion spell (skill). Complete documentation about string placeholders and parsing is here.

Parameters

- **id** (*str*) – spell's ID, including champion name and spell name
- **name** (*str*) – spell's name
- **description** (*str*) – spell's description
- **tooltip** (*str*) – similar to `description`, but contains placeholders to build a string including data about damage per level, AP/AD scaling ecc. Can be parsed to make the string look like the one in game
- **maxrank** (*int*) – maximum rank for this ability
- **cooldown** (*List[int]*) – data about placeholders for cooldown
- **cooldownBurn** (*str*) – like `cooldown`, but as string
- **cost** (*List[int]*) – data about placeholders for cost
- **costBurn** (*str*) – like `cost`, but as string
- **datavalues** (*ChampionSpellDatavaluesDD*) – no documentation found. No champion found with a non-empty dict of `datavalues`
- **effect** (*List[Optional[List[int]]]*) – like `cost`, but when the spell costs health. The first element is always None for design reasons
- **effectBurn** (*List[Optional[str]]*) – like `effect`, but as string

- **vars** (*List[unknown]*) – no documentation found. No champion found with a non-empty list of `vars`
- **costType** (*str*) – type of resources spent for using the ability
- **maxammo** (*str*) – in case the spell has ammos, like traps. '-1' if no ammos. For some reason this integer is represented as a string
- **range** (*List[int]*) – data about placeholders for range
- **rangeBurn** (*str*) – like `range`, but as string
- **image** (*ChampionSpellImageDD*) – details about the spell's image
- **leveltip** (*ChampionSpellLeveltipDD*) – data about placeholders for levels
- **resource** (*Optional[str]*) – placeholder for cost

class async_riot_api.types.**ChampionSpellLeveltipDD**(*label: List[str]*, *effect: List[str]*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**ChampionSpellDatavaluesDD**(***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**ChampionSpellImageDD**(*full: str*, *sprite: str*, *group: str*, *x: int*, *y: int*, *w: int*, *h: int*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**ChampionPassiveDD**(*name: str*, *description: str*, *image: dict*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**ChampionPassiveImageDD**(*full: str*, *sprite: str*, *group: str*, *x: int*, *y: int*, *w: int*, *h: int*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**QueueDD**(*queueId: int*, *map: str*, *description: str*, *notes: str*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

Representation of a queue. Not actually returned by any API call, but still usefu sometimes.

Parameters

- **queueId** (*int*) – queue ID
- **map** (*str*) – map name
- **description** (*str*) – description of the queue
- **notes** (*str*) – notes about the queue, like 'deprecated since version x'

class async_riot_api.types.**AccountDto**(*puuid: str*, *gameName: str*, *tagLine: str*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

Basic information about a Riot Games account.

Parameters

- **puuid** (*str*) – puuid of the account. Useful for many API methods
- **gameName** (*str*) – in-game name of the account
- **tagLine** (*str*) – tag line of the account

class async_riot_api.types.**ActiveShardDto**(*puuid: str*, *game: str*, *activeShard: str*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

No idea about what this is. Probably the server in which the summoner plays.

Parameters

- **puuid** (*str*) – puuid
- **game** (*str*) – game. Can be 'val' or 'lor'
- **activeShard** (*str*) – probably the server in which the summoner plays

class async_riot_api.types.**ChampionMasteryDto**(*championPointsUntilNextLevel: int*, *chestGranted: bool*, *championId: int*, *lastPlayTime: int*, *championLevel: int*, *summonerId: str*, *championPoints: int*, *championPointsSinceLastLevel: int*, *tokensEarned: int*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Information about a summoner's mastery levels, tokens and chests.

Parameters

- **championPointsUntilNextLevel** (*int*) – points needed for the mastery to upgrade to the next level
- **chestGranted** (*bool*) – if the player already got the weekly chest on the champion
- **championId** (*int*) – champion ID
- **lastPlayTime** (*int*) – laste time the player played the champion
- **lastPlayTimeSeconds** (*int*) – laste time the player played the champion (granted to be in seconds)
- **championLevel** (*int*) – mastery level for the champion. Min 1, max 7
- **summonerId** (*str*) – summoner ID
- **championPoints** (*int*) – mastery points for the champion
- **championPointsSinceLastLevel** (*int*) – points earned since last mastery level
- **tokensEarned** (*int*) – tokens earned to upgrade the mastery level to level 6 (0-2) or 7 (0-3)

class async_riot_api.types.**ChampionInfo**(*maxNewPlayerLevel: int*, *freeChampionIdsForNewPlayers: List[int]*, *freeChampionIds: List[int]*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Information about the current champion rotation and the new players' champion rotation.

Parameters

- **maxNewPlayerLevel** (*int*) – max level for a player to have the beginner rotation available
- **freeChampionIdsForNewPlayers** (*List[int]*) – list of champion IDs free-to-play for beginners
- **freeChampionIds** (*List[int[*) – list of champion IDs free-to-play for non-beginner players

class async_riot_api.types.**PlayerDto**(*summonerId: str*, *teamId: str*, *position: str*, *role: str*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Data about a player in a clash.

Parameters

- **summonerId** (*str*) – summoner ID
- **teamId** (*str*) – team ID

- **position** (*str*) – position selected by the summoner
- **role** (*str*) – role selected by the summoner

class async_riot_api.types.**ClashTeamDto**(*id: str*, *tournamentId: int*, *name: str*, *iconId: int*, *tier: int*, *captain: str*, *abbreviation: str*, *players: List[dict]*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**TournamentDto**(*id: int*, *themeId: int*, *nameKey: str*, *nameKeySecondary: str*, *schedule: List[dict]*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Information about a tournament.

Parameters

- **id** (*int*) – tournament ID
- **themeId** (*int*) – no idea, not originally documented
- **nameKey** (*str*) – no idea, not originally documented
- **nameKeySecondary** (*str*) – no idea, not originally documented
- **schedule** (*TournamentPhaseDto*) – schedule for this tournament

class async_riot_api.types.**TournamentPhaseDto**(*id: int*, *registrationTime: int*, *startTime: int*, *cancelled: bool*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Schedule information for a tournament.

Parameters

- **id** (*int*) – ID
- **registrationTime** (*int*) – timestamp in ms
- **registrationTimeSeconds** (*int*) – timestamp in seconds
- **startTime** (*int*) – timestamp in ms
- **startTimeSeconds** (*int*) – timestamp in seconds
- **cancelled** (*bool*) – wether the tournament is cancelled

class async_riot_api.types.**LeagueListDTO**(*tier: str*, *leagueId: str*, *queue: str*, *name: str*, *entries: List[dict]*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

List of information about leagues for summoners in the same queue.

Parameters

- **tier** (*str*) – rank tier, like 'CHALLENGER' or 'MASTER'
- **leagueId** (*str*) – league ID
- **queue** (*str*) – queue type, like 'RANKED_SOLO_5x5'
- **name** (*str*) – list name
- **entries** (List[*LeagueItemDTO*]) – entries for this list

class async_riot_api.types.**LeagueItemDTO**(*summonerId: str*, *summonerName: str*, *leaguePoints: int*, *rank: str*, *wins: int*, *losses: int*, *veteran: bool*, *inactive: bool*, *freshBlood: bool*, *hotStreak: bool*, *miniSeries: Optional[dict] = None*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Simplified information about a summoner's rank in a queue, returned by methods for apex tiers. Some information are missing since they are included in the higher level object `LeagueListDTO` containing this object.

Parameters

- **summonerId** (*str*) – summoner ID
- **summonerName** (*str*) – summoner name
- **leaguePoints** (*int*) – aka LP
- **rank** (*str*) – rank of the summoner, between 'I' and 'IV' (in roman numbers)
- **wins** (*int*) – wins for this season
- **losses** (*int*) – losses for this season
- **veteran** (*bool*) – wether the summoner is a veteran in this rank
- **inactive** (*bool*) – wether the summoner is inactive
- **freshBlood** (*bool*) – wether the summoner is a new entry in this rank
- **hotStreak** (*bool*) – wether the summoner is on a hot streak (winning streak)
- **miniSeries** (Optional[*MiniSeriesDTO*]) – information about a summoner miniseries, if they are about to get promoted from a tier to the next

class async_riot_api.types.**LeagueEntryDTO**(*summonerId: str*, *summonerName: str*, *queueType: str*, *leaguePoints: int*, *wins: int*, *losses: int*, *hotStreak: bool*, *veteran: bool*, *freshBlood: bool*, *inactive: bool*, *miniSeries: Optional[dict] = None*, *leagueId: Optional[str] = None*, *tier: Optional[str] = None*, *rank: Optional[str] = None*, ***kwargs*)

Bases: *async_riot_api.types.LeagueItemDTO*

Complete information about summoner's league.

Look at *LeagueItemDTO* for the complete list of parameters.

Parameters

- **queueType** (*str*) – queue for this entry, like 'RANKED_SOLO_5x5'
- **leagueId** (*Optional[str]*) – league ID
- **tier** (*Optional[str]*) – tier for this entry, like 'SILVER' or 'DIAMOND'

Other attributes:

short (Optional[str]): short representation of rank and tier. For example 'DIAMOND III' becomes 'D3'. Exception is made for 'GRANDMASTER x' which becomes 'GMx' due to the ambuguity between 'GOLD' and 'GRANDMASTER'

class async_riot_api.types.**MiniSeriesDTO**(*losses: int*, *progress: str*, *target: int*, *wins: int*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Information about a summoner's miniseries, if they are about to get promoted to the next tier. Miniseries consist in 5 matches in which you have to win 3 times to get promoted. Before season 11, miniseries were also required when passing from a rank to another, with 3 matches instead of 5 and 2 victories instead of 3.

Parameters

- **losses** (*int*) – losses in this miniseries
- **progress** (*str*) – string representing wins and losses. for example 'WWLNN' means two wins, one loss and two matches remained
- **target** (*int*) – number of wins to reach
- **wins** (*int*) – wins in this miniseries

class async_riot_api.types.**ShardStatus**(*name: str*, *slug: str*, *locales: List[str]*, *hostname: str*, *region_tag: str*, *services: List[dict]*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**Service**(*name: str*, *slug: str*, *status: str*, *incidents: List[dict]*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**Incident**(*id: int*, *active: bool*, *created_at: str*, *updates: List[dict]*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**Message**(*id: str*, *author: str*, *heading: str*, *content: str*, *severity: str*, *created_at: str*, *updated_at: str*, *translations: List[dict]*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**Translation**(*locale: str*, *heading: str*, *content: str*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**PlatformDataDto**(*id: str*, *name: str*, *locales: List[str]*, *maintenances: List[dict]*, *incidents: List[dict]*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**StatusDto**(*id: int*, *maintenance_status: str*, *incident_severity: Optional[str]*, *titles: List[dict]*, *updates: List[dict]*, *created_at: str*, *archive_at: str*, *updated_at: Optional[str]*, *platforms: List[str]*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**ContentDto**(*locale: str*, *content: str*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**UpdateDto**(*id: int*, *author: str*, *publish: bool*, *publish_locations: List[str]*, *translations: List[dict]*, *created_at: str*, *updated_at: str*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**LorMatchDto**(*metadata: dict*, *info: dict*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Base object containing inforamtion about a LoR match.

Parameters

- **metadata** (*LorMetadataDto*) – to access the ordered list of participants
- **info** (*LorInfoDto*) – more detailed infor about players

class async_riot_api.types.**LorMetadataDto**(*data_version: str*, *match_id: str*, *participants: List[str]*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Metadata about the match.

Parameters

- **data_version** (*str*) – version of the game

- **match_id** (*str*) – match ID
- **participants** (*List[str]*) – ordered list of participants, represented by their puuid

class async_riot_api.types.**LorInfoDto**(*game_mode: str*, *game_type: str*, *game_start_time_utc: str*, *game_version: str*, *players: List[dict]*, *total_turn_count: int*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Detailed information about a LoR match. Contains information about mode, duration and players.

Parameters

- **game_mode** (*str*) – game mode
- **game_type** (*str*) – game type
- **game_start_time_utc** (*str*) – game start time utc
- **game_version** (*str*) – game version
- **players** (List[*LorPlayerDto*]) – list of detailed information about the players involved in the match, in the same order as *LorMetadataDto*
- **total_turn_count** (*int*) – total turn count

class async_riot_api.types.**LorPlayerDto**(*puuid: str*, *deck_id: str*, *deck_code: str*, *factions: List[str]*, *game_outcome: str*, *order_of_play: int*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Detailed information about a player in a LoR match.

Parameters

- **puuid** (*str*) – puuid
- **deck_id** (*str*) – deck ID
- **deck_code** (*str*) – deck code
- **factions** (*List[str]*) – factions persent in the deck
- **game_outcome** (*str*) – result of the game
- **order_of_play** (*int*) – order if play in the game

class async_riot_api.types.**LorLeaderboardDto**(*players: List[dict]*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

List of players in LoR Master tier. :param players: list of players :type players: List[*LorLeaderboardPlayerDto*]

class async_riot_api.types.**LorLeaderboardPlayerDto**(*name: str*, *rank: int*, *lp: int*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

Information about a player in LoR Master tier.

Parameters

- **name** (*str*) – summoner's name
- **rank** (*int*) – summoner's rank
- **lp** (*int*) – summoner's LP

class async_riot_api.types.**MatchDto**(*metadata: dict*, *info: dict*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**MetadataDto**(*dataVersion: str*, *matchId: str*, *participants: List[str]*, ***kwargs*)
Bases: async_riot_api.types.RiotApiResponse

class async_riot_api.types.**InfoDto**(*gameCreation: int*, *gameDuration: int*, *gameId: int*, *gameMode: str*, *gameName: str*, *gameStartTimestamp: int*, *gameType: str*, *gameVersion: str*, *mapId: int*, *participants: List[str]*, *platformId: str*, *queueId: int*, *teams: List[dict]*, *tournamentCode: Optional[str] = None*, *gameEndTimestamp: int = 0*, ***kwargs*)
Bases: async_riot_api.types.RiotApiResponse

class async_riot_api.types.**ParticipantDto**(*assists: int*, *baronKills: int*, *bountyLevel: int*, *champExperience: int*, *champLevel: int*, *championId: int*, *championName: str*, *championTransform: int*, *consumablesPurchased: int*, *damageDealtToBuildings: int*, *damageDealtToObjectives: int*, *damageDealtToTurrets: int*, *damageSelfMitigated: int*, *deaths: int*, *detectorWardsPlaced: int*, *doubleKills: int*, *dragonKills: int*, *firstBloodAssist: bool*, *firstBloodKill: bool*, *firstTowerAssist: bool*, *firstTowerKill: bool*, *gameEndedInEarlySurrender: bool*, *gameEndedInSurrender: bool*, *goldEarned: int*, *goldSpent: int*, *individualPosition: str*, *inhibitorKills: int*, *inhibitorsLost: int*, *item0: int*, *item1: int*, *item2: int*, *item3: int*, *item4: int*, *item5: int*, *item6: int*, *itemsPurchased: int*, *killingSprees: int*, *kills: int*, *lane: str*, *largestCriticalStrike: int*, *largestKillingSpree: int*, *largestMultiKill: int*, *longestTimeSpentLiving: int*, *magicDamageDealt: int*, *magicDamageDealtToChampions: int*, *magicDamageTaken: int*, *neutralMinionsKilled: int*, *nexusKills: int*, *nexusLost: int*, *objectivesStolen: int*, *objectivesStolenAssists: int*, *participantId: int*, *pentaKills: int*, *perks: dict*, *physicalDamageDealt: int*, *physicalDamageDealtToChampions: int*, *physicalDamageTaken: int*, *profileIcon: int*, *puuid: str*, *quadraKills: int*, *riotIdName: str*, *riotIdTagline: str*, *role: str*, *sightWardsBoughtInGame: int*, *spell1Casts: int*, *spell2Casts: int*, *spell3Casts: int*, *spell4Casts: int*, *summoner1Casts: int*, *summoner1Id: int*, *summoner2Casts: int*, *summoner2Id: int*, *summonerId: str*, *summonerLevel: int*, *summonerName: str*, *teamEarlySurrendered: bool*, *teamId: int*, *teamPosition: str*, *timeCCingOthers: int*, *timePlayed: int*, *totalDamageDealt: int*, *totalDamageDealtToChampions: int*, *totalDamageShieldedOnTeammates: int*, *totalDamageTaken: int*, *totalHeal: int*, *totalHealsOnTeammates: int*, *totalMinionsKilled: int*, *totalTimeCCDealt: int*, *totalTimeSpentDead: int*, *totalUnitsHealed: int*, *tripleKills: int*, *trueDamageDealt: int*, *trueDamageDealtToChampions: int*, *trueDamageTaken: int*, *turretKills: int*, *turretsLost: int*, *unrealKills: int*, *visionScore: int*, *visionWardsBoughtInGame: int*, *wardsKilled: int*, *wardsPlaced: int*, *win: bool*, *inhibitorTakedowns: int = 0*, *nexusTakedowns: int = 0*, *turretTakedowns: int = 0*, ***kwargs*)
Bases: async_riot_api.types.RiotApiResponse

class async_riot_api.types.**PerksDto**(*statPerks: dict*, *styles: List[dict]*, ***kwargs*)
Bases: async_riot_api.types.RiotApiResponse

class async_riot_api.types.**PerkStatsDto**(*defense: int*, *flex: int*, *offense: int*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**PerkStyleDto**(*description: str*, *selections: List[dict]*, *style: int*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**PerkStyleSelectionDto**(*perk: int*, *var1: int*, *var2: int*, *var3: int*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**TeamDto**(*bans: List[dict]*, *objectives: dict*, *teamId: int*, *win: bool*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**BanDto**(*championId: int*, *pickTurn: int*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**ObjectivesDto**(*baron: dict*, *champion: dict*, *dragon: dict*, *inhibitor: dict*, *riftHerald: dict*, *tower: dict*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**ObjectiveDto**(*first: bool*, *kills: int*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**MatchTimelineDto**(*metadata: dict*, *info: dict*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**MTLInfoDto**(*frameInterval: int*, *frames: List[dict]*, *gameId: int*, *participants: List[dict]*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**MTLFrameDto**(*events: List[dict]*, *participantFrames: dict*, *timestamp: int*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**MTLEventDto**(*timestamp: int*, *type: str*, *levelUpType: Optional[str] = None*, *participantId: Optional[int] = None*, *skillSlot: Optional[int] = None*, *realTimestamp: Optional[int] = None*, *itemId: Optional[int] = None*, *afterId: Optional[int] = None*, *beforeId: Optional[int] = None*, *goldGain: Optional[int] = None*, *creatorId: Optional[int] = None*, *wardType: Optional[int] = None*, *assistingParticipantIds: Optional[List[int]] = None*, *bounty: Optional[int] = None*, *killStreakLength: Optional[int] = None*, *killerId: Optional[int] = None*, *position: Optional[dict] = None*, *victimDamageDealt: Optional[List[dict]] = None*, *victimDamageReceived: Optional[List[dict]] = None*, *victimId: Optional[int] = None*, *killType: Optional[int] = None*, *level: Optional[int] = None*, *multiKillLength: Optional[int] = None*, *laneType: Optional[str] = None*, *teamId: Optional[int] = None*, *killerTeamId: Optional[int] = None*, *monsterSubType: Optional[str] = None*, *monsterType: Optional[str] = None*, *buildingType: Optional[str] = None*, *towerType: Optional[str] = None*, *name: Optional[str] = None*, *gameId: Optional[int] = None*, *winningTeam: Optional[int] = None*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**MTLDamageDto**(*basic: bool*, *magicDamage: int*, *name: str*, *participantId: int*, *physicalDamage: int*, *spellName: str*, *spellSlot: int*, *trueDamage: int*, *type: str*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**MTLParticipantFramesDto**(*f1: dict*, *f2: dict*, *f3: dict*, *f4: dict*, *f5: dict*, *f6: dict*, *f7: dict*, *f8: dict*, *f9: dict*, *f10: dict*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**MTLParticipantFrameDto**(*championStats: dict*, *currentGold: int*, *damageStats: dict*, *goldPerSecond: int*, *jungleMinionsKilled: int*, *level: int*, *minionsKilled: int*, *participantId: int*, *position: dict*, *timeEnemySpentControlled: int*, *totalGold: int*, *xp: int*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**MTLChampionStatsDto**(*abilityHaste: int*, *abilityPower: int*, *armor: int*, *armorPen: int*, *armorPenPercent: int*, *attackDamage: int*, *attackSpeed: int*, *bonusArmorPenPercent: int*, *bonusMagicPenPercent: int*, *ccReduction: int*, *cooldownReduction: int*, *health: int*, *healthMax: int*, *healthRegen: int*, *lifesteal: int*, *magicPen: int*, *magicPenPercent: int*, *magicResist: int*, *movementSpeed: int*, *omnivamp: int*, *physicalVamp: int*, *power: int*, *powerMax: int*, *powerRegen: int*, *spellVamp: int*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**MTLDamageStatsDto**(*magicDamageDone: int*, *magicDamageDoneToChampions: int*, *magicDamageTaken: int*, *physicalDamageDone: int*, *physicalDamageDoneToChampions: int*, *physicalDamageTaken: int*, *totalDamageDone: int*, *totalDamageDoneToChampions: int*, *totalDamageTaken: int*, *trueDamageDone: int*, *trueDamageDoneToChampions: int*, *trueDamageTaken: int*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**MTLPositionDto**(*x: int*, *y: int*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**MTLParticipantDto**(*participantId: int*, *puuid: str*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**CurrentGameInfo**(*gameId: int*, *gameType: str*, *gameStartTime: int*, *mapId: int*, *gameLength: int*, *platformId: str*, *gameMode: str*, *bannedChampions: List[dict]*, *gameQueueConfigId: int*, *observers: dict*, *participants: List[dict]*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**BannedChampion**(*championId: int*, *teamId: int*, *pickTurn: int*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**CurrentGameParticipant**(*championId: int*, *perks: dict*, *profileIconId: int*, *bot: bool*, *teamId: int*, *summonerName: str*, *summonerId: str*, *spell1Id: int*, *spell2Id: int*, *gameCustomizationObjects: List[dict]*, ***kwargs*)

Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**Perks**(*perkIds: List[int]*, *perkStyle: int*, *perkSubStyle: int*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**GameCustomizationObject**(*category: str*, *content: str*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**FeaturedGames**(*gameList: List[dict]*, *clientRefreshInterval: int*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**FeaturedGameInfo**(*gameMode: str*, *gameLength: int*, *mapId: int*, *gameType: str*, *bannedChampions: List[dict]*, *gameId: int*, *observers: dict*, *gameQueueConfigId: int*, *gameStartTime: int*, *participants: List[dict]*, *platformId: str*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**Observer**(*encryptionKey: str*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**Participant**(*teamId: int*, *spell1Id: int*, *spell2Id: int*, *championId: int*, *profileIconId: int*, *summonerName: str*, *bot: bool*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

class async_riot_api.types.**SummonerDTO**(*accountId: str*, *profileIconId: int*, *revisionDate: int*, *name: str*, *id: str*, *puuid: str*, *summonerLevel: int*, ***kwargs*)
Bases: *async_riot_api.types.RiotApiResponse*

PYTHON MODULE INDEX

a

async_riot_api.types, 16

INDEX

A

AccountDto (*class in async_riot_api.types*), 20
ActiveShardDto (*class in async_riot_api.types*), 20
async_riot_api.types
    module, 16

B

BanDto (*class in async_riot_api.types*), 27
BannedChampion (*class in async_riot_api.types*), 28

C

ChampionDD (*class in async_riot_api.types*), 17
ChampionImageDD (*class in async_riot_api.types*), 17
ChampionInfo (*class in async_riot_api.types*), 21
ChampionInfoDD (*class in async_riot_api.types*), 18
ChampionMasteryDto (*class in async_riot_api.types*), 21
ChampionPassiveDD (*class in async_riot_api.types*), 20
ChampionPassiveImageDD (*class in async_riot_api.types*), 20
ChampionSkinDD (*class in async_riot_api.types*), 17
ChampionSpellDatavaluesDD (*class in async_riot_api.types*), 20
ChampionSpellDD (*class in async_riot_api.types*), 19
ChampionSpellImageDD (*class in async_riot_api.types*), 20
ChampionSpellLeveltipDD (*class in async_riot_api.types*), 20
ChampionStatsDD (*class in async_riot_api.types*), 18
ClashTeamDto (*class in async_riot_api.types*), 22
compute_champion_from_similar_name() (*async_riot_api.LoLAPI static method*), 14
compute_language() (*async_riot_api.LoLAPI static method*), 14
ContentDto (*class in async_riot_api.types*), 24
CurrentGameInfo (*class in async_riot_api.types*), 28
CurrentGameParticipant (*class in async_riot_api.types*), 28

F

FeaturedGameInfo (*class in async_riot_api.types*), 29
FeaturedGames (*class in async_riot_api.types*), 29

G

GameCustomizationObject (*class in async_riot_api.types*), 29
get_account_by_game_name() (*async_riot_api.LoLAPI method*), 7
get_account_by_puuid() (*async_riot_api.LoLAPI method*), 7
get_active_games() (*async_riot_api.LoLAPI method*), 12
get_active_shards() (*async_riot_api.LoLAPI method*), 7
get_challenger_leagues() (*async_riot_api.LoLAPI method*), 10
get_champion_from_correct_name() (*async_riot_api.LoLAPI static method*), 15
get_champion_from_id() (*async_riot_api.LoLAPI static method*), 15
get_champion_image_url_from_id() (*async_riot_api.LoLAPI static method*), 14
get_champion_mastery() (*async_riot_api.LoLAPI method*), 8
get_champion_rotation() (*async_riot_api.LoLAPI method*), 8
get_clash_players_by_summoner_id() (*async_riot_api.LoLAPI method*), 8
get_clash_team_by_id() (*async_riot_api.LoLAPI method*), 9
get_clash_tournament_by_id() (*async_riot_api.LoLAPI method*), 9
get_clash_tournament_by_team_id() (*async_riot_api.LoLAPI method*), 9
get_clash_tournaments() (*async_riot_api.LoLAPI method*), 9
get_featured_games() (*async_riot_api.LoLAPI method*), 12
get_flex_league() (*async_riot_api.LoLAPI method*), 14
get_full_champion_from_correct_name()

(*async_riot_api.LoLAPI static method*), 15
get_grand_master_leagues() (*async_riot_api.LoLAPI method*), 10
get_last_match() (*async_riot_api.LoLAPI method*), 13
get_league() (*async_riot_api.LoLAPI method*), 10
get_leagues() (*async_riot_api.LoLAPI method*), 10
get_lor_leaderboards() (*async_riot_api.LoLAPI method*), 11
get_lor_match() (*async_riot_api.LoLAPI method*), 11
get_lor_matches() (*async_riot_api.LoLAPI method*), 11
get_lor_status() (*async_riot_api.LoLAPI method*), 11
get_map_icon_url() (*async_riot_api.LoLAPI static method*), 15
get_master_leagues() (*async_riot_api.LoLAPI method*), 10
get_masteries() (*async_riot_api.LoLAPI method*), 8
get_mastery_score() (*async_riot_api.LoLAPI method*), 8
get_match() (*async_riot_api.LoLAPI method*), 12
get_matches() (*async_riot_api.LoLAPI method*), 11
get_nth_match() (*async_riot_api.LoLAPI method*), 13
get_platform_data() (*async_riot_api.LoLAPI method*), 11
get_platform_data_v3() (*async_riot_api.LoLAPI method*), 11
get_profile_icon_url() (*async_riot_api.LoLAPI static method*), 14
get_queue() (*async_riot_api.LoLAPI static method*), 15
get_solo_league() (*async_riot_api.LoLAPI method*), 14
get_summoner_by_account_id() (*async_riot_api.LoLAPI method*), 13
get_summoner_by_name() (*async_riot_api.LoLAPI method*), 13
get_summoner_by_puuid() (*async_riot_api.LoLAPI method*), 13
get_summoner_by_summoner_id() (*async_riot_api.LoLAPI method*), 13
get_summoners_by_league() (*async_riot_api.LoLAPI method*), 10
get_summoners_by_league_exp() (*async_riot_api.LoLAPI method*), 9
get_timeline() (*async_riot_api.LoLAPI method*), 12
get_version() (*async_riot_api.LoLAPI static method*), 15

I

Incident (*class in async_riot_api.types*), 24
InfoDto (*class in async_riot_api.types*), 26

L

LeagueEntryDTO (*class in async_riot_api.types*), 23
LeagueItemDTO (*class in async_riot_api.types*), 22
LeagueListDTO (*class in async_riot_api.types*), 22
LoLAPI (*class in async_riot_api*), 7
LorInfoDto (*class in async_riot_api.types*), 25
LorLeaderboardDto (*class in async_riot_api.types*), 25
LorLeaderboardPlayerDto (*class in async_riot_api.types*), 25
LorMatchDto (*class in async_riot_api.types*), 24
LorMetadataDto (*class in async_riot_api.types*), 24
LorPlayerDto (*class in async_riot_api.types*), 25

M

MatchDto (*class in async_riot_api.types*), 25
MatchTimelineDto (*class in async_riot_api.types*), 27
Message (*class in async_riot_api.types*), 24
MetadataDto (*class in async_riot_api.types*), 25
MiniSeriesDTO (*class in async_riot_api.types*), 23
module
async_riot_api.types, 16
MTLChampionStatsDto (*class in async_riot_api.types*), 28
MTLDamageDto (*class in async_riot_api.types*), 27
MTLDamageStatsDto (*class in async_riot_api.types*), 28
MTLEventDto (*class in async_riot_api.types*), 27
MTLFrameDto (*class in async_riot_api.types*), 27
MTLInfoDto (*class in async_riot_api.types*), 27
MTLParticipantDto (*class in async_riot_api.types*), 28
MTLParticipantFrameDto (*class in async_riot_api.types*), 28
MTLParticipantFramesDto (*class in async_riot_api.types*), 27
MTLPositionDto (*class in async_riot_api.types*), 28

O

ObjectiveDto (*class in async_riot_api.types*), 27
ObjectivesDto (*class in async_riot_api.types*), 27
Observer (*class in async_riot_api.types*), 29

P

Participant (*class in async_riot_api.types*), 29
ParticipantDto (*class in async_riot_api.types*), 26
Perks (*class in async_riot_api.types*), 28
PerksDto (*class in async_riot_api.types*), 26
PerkStatsDto (*class in async_riot_api.types*), 26
PerkStyleDto (*class in async_riot_api.types*), 27
PerkStyleSelectionDto (*class in async_riot_api.types*), 27
PlatformDataDto (*class in async_riot_api.types*), 24
PlayerDto (*class in async_riot_api.types*), 21

Q

QueueDD (*class in async_riot_api.types*), 20

R

RiotApiError (*class in async_riot_api.types*), 16
RiotApiResponse (*class in async_riot_api.types*), 16

S

Service (*class in async_riot_api.types*), 24
ShardStatus (*class in async_riot_api.types*), 24
ShortChampionDD (*class in async_riot_api.types*), 16
StatusDto (*class in async_riot_api.types*), 24
SummonerDTO (*class in async_riot_api.types*), 29

T

TeamDto (*class in async_riot_api.types*), 27
to_string() (*async_riot_api.types.RiotApiResponse method*), 16
TournamentDto (*class in async_riot_api.types*), 22
TournamentPhaseDto (*class in async_riot_api.types*), 22
Translation (*class in async_riot_api.types*), 24

U

UpdateDto (*class in async_riot_api.types*), 24