Riot API Libraries Documentation

WxWatch

Feb 01, 2020

Contents:

CHAPTER 1

You Should Know. . .

There are a few crucial things to know about the Riot API.

- Your API key expires every 24 hours unless you applied for and received a key for a personal or production application. If you're getting a 403 response from the Riot API, you probably just need to refresh your key.
- Applications take ~ two weeks (10 business days, barring holidays) to process. Make sure it's verified (riot.txt). If it's been longer than two weeks and you didn't change your application during that time period, ask a Guru on Discord for help.
- You should use a library! All libraries are developed by the community and will make using the Riot API *much* easier. Nearly all of them take care of rate limiting, for example.
- If you have questions, you can ask on the Discord, but check here first. We try to collect common questions and put them in this documentation.
- To convert champion IDs to names, use Data Dragon.
- These docs are written by the community. Riot's official docs can be found here.

1.1 Riot API Libraries

1.1.1 C#

RiotSharp (202)

RiotSharp's ASP.NET Core integration. **NOTE: NuGet releases are severely outdated, although library itself is maintained!**

Links

License: MIT
Tags: v4, rate-limiting, asp-net-core

Last Updated: Feb 1, 2020

Camille (14)

Fully rate limited, automatic retrying, thread-safe. V4 Supported. Automatic nightly releases.

Links

- NuGet

License: LGPL-3.0
Tags: v3, v4, rate-limiting
Last Updated: Jan 30, 2020

1.1.2 Go

golio (7)

League of Legends API client written in Golang

Links

- src
- docs

License: MIT
Tags: v4, rate-limiting, caching
Last Updated: Feb 1, 2020

1.1.3 Java

R4J (27)

A Java library containing the API for every Riot game

Links

License: Apache-2.0
Tags: v4
Last Updated: Jan 29, 2020

---

orianna (107)

A highly configurable, usability-focused Riot API framework that takes care of all the details for you so you can focus on building your application

Links

- Maven
- Documentation
- JavaDoc

License: MIT
Tags: v4, rate-limiting, caching
Last Updated: Nov 24, 2019

---

riot-api-java (168)

A simple to use Riot Games API wrapper for Java. This library makes it easy to gather and use League of Legends data in your apps.

Links

- JitPack

License: Apache-2.0
Tags: v4, rate-limiting
Last Updated: Jul 9, 2019

1.1.4 JavaScript

LeagueJS (48)

A Javascript Wrapper for the League of Legends API

Links

- npm
- gitter

License: MIT
Tags: v4, caching, ddragon, rate-limiting
Last Updated: Jan 8, 2020

---

kayn (102)

superagent-inspired Node.js lib (w/ **some** TypeScript support) for accessing Riot's League of Legend's API (discord: cnguy#3614)

Links

- npm
- rate-limiter-credits

License: MIT
Tags: v4, rate-limiting, caching
Last Updated: Dec 28, 2019

---

twisted (3)

Fetching riot games api data

Links

- npm
- github
- examples

License: None
Tags: rate-limiting, v4, lol, tft, caching
Last Updated: Dec 23, 2019

Pyke (21)

Riot API, DDragon and CommunityDragon Library for NodeJS

Links

- npm

License: GPL-3.0
Tags: v4, ddragon, communitydragon
Last Updated: Dec 15, 2019

TeemoJS (13)

Fast & tiny, automatic retries & smart rate limiting, V4 & champion.gg support, all in 300 lines.

Links

- npm

License: LGPL-3.0
Tags: v3, v4, rate-limiting
Last Updated: Dec 13, 2019

MundoScript (1)

Node.js League/LoL/League Of Legends API wrapper, Focus is to be easy to use

Links

- npm
- Documentation

License: None
Tags: v4
Last Updated: May 15, 2019

1.1.5 Julia

LOLTools.jl (1)

Julia package to the Riot Games API for League of Legends.

Links

License: NOASSERTION
Tags: v4
Last Updated: Jan 30, 2020

1.1.6 PHP

riot-api (73)

Riot League of Legends & DataDragon API wrappers for PHP7

Links

- GitHub Wiki
- Packagist

License: GPL-3.0
Tags: v3, v4, rate-limiting, cli
Last Updated: Dec 3, 2019

1.1.7 Python

Riot-Watcher (272)

Python wrapper for the Riot Games API for League of Legends

Links

- Documentation
- PyPi

License: MIT
Tags: v4, rate-limiting
Last Updated: Jan 8, 2020

---

pantheon (7)

Riot API library for Python and asyncio

Links

License: MIT
Tags: v4, rate-limiting
Last Updated: Jan 7, 2020

---

cassiopeia (244)

A highly configurable, usability-focused Riot API framework that takes care of all the details for you so you can focus on building your application

Links

- PyPi
- Documentation

License: MIT
Tags: v4, rate-limiting, caching
Last Updated: Dec 5, 2019

1.1.8 Rust

Riven (5)

Tried and tested Riot API design, in Rust

Links

- Docs.rs
- Crates.io

License: MIT
Tags: v3, v4, rate-limiting, tft
Last Updated: Jan 30, 2020

1.1.9 Swift

LeagueAPI (18)

Framework providing all League of Legends data, with cache, rate-limit handling with auto retry system. Compatible with Carthage and Cocoapod.

Links

- Github
- Documentation

License: MIT
Tags: v4, rate-limiting
Last Updated: Nov 7, 2019

DragonService (1)

Swift package to fetch data from DataDragon

Links

License: MIT
Tags: v4
Last Updated: Nov 20, 2018

1.1.10 TypeScript

twisted (3)

Fetching riot games api data

Links

- npm
- github
- examples

License: None
Tags: rate-limiting, v4, lol, tft, caching
Last Updated: Dec 23, 2019

1.2 Data Dragon

Data Dragon, or ddragon for short, is a set of static data files that provides images and info about champions, runes, and items. This includes info to translate champion IDs to names.

You can download the entire set of data and images by downloading this tarball (make sure to change the * version * to the latest patch): https://ddragon.leagueoflegends.com/cdn/dragontail-9.3.1.tgz. The file is about 0.5 GB.

You can also access individual files by going to specific URLs, for example: http://ddragon.leagueoflegends.com/cdn/9.3.1/data/en_US/champion.json.

To see what files are in ddragon, PykeGithub on Discord has created a website that lets you explore ddragon. Do not download files from ddragonexplorer.com. Instead, replace `ddragonexplorer.com` with `ddragon.leagueoflegends.com` and download the file directly from Riot's servers.

1.2.1 Caching

Because the data in ddragon only changes when new patches come out, you can cache the data by saving it to your computer. Your app can then load the data from disk rather than requesting it over the www. This will speed up your app and reduce the load on Riot's servers, which ensures the servers don't go down due to abnormally high usage.

In general, it's a good idea to cache data that you will use often and that doesn't change often.

1.2.2 Community Dragon

Community Dragon, or cdragon for short, is a massive collection of community-generated files to augment the data in ddragon.

If you can't find what you're looking for in ddragon, look in cdragon. This is a useful link to start at. If you can't find what you're looking for because there's so many files and folders to go through, ask on Discord for help.

One commonly sought-after set of images are the rune stat images, which can be found here.

1.2.3 Common Issues

- If You can't find something in ddragon, Look in cdragon or ask on Discord.
- The image filenames for champions, items, etc. can be found by looking at the `image.full` fields in the json data.
- Monkeyking. Yes, Wukong's internal name is `monkeyking`. You can identify the correct internal name for every champion by looking at the `key` attribute in the champion.json file.
- The data in ddragon is inaccurate, especially champion spell data and item stats. This is an unfortunate situation that is surprisingly difficult to solve. If you want to know why, you can ask on Discord. There is no perfect or even close to perfect source for champion spell data, despite significant effort. The League Wikia is your best resource.
- If a new patch came out but ddragon hasn't updated, wait a little while. Sometimes it takes two days after a patch for the new version of ddragon to be released. Other times ddragon is two days ahead of the patch.
- Data from Data Dragon is encoded using UTF-8. To ensure you receive a properly encoded response include the Accept-Charset header specifying UTF-8 as the desired response encoding.
- Other issues: You may be able to find some information by searching the github issues, otherwise ask on Discord.

1.2.4 Most Recent Version

You can get the latest version of ddragon from the version.json file.

Because patches get released on different regions at different times, one region may be on the new version of ddragon while another region may be be on an old version. You can check which patch a region is on by looking at the realms files for each region. The one for North America can be found at https://ddragon.leagueoflegends.com/realms/na.json. The exact time the patch gets released on a region may not correspond to exactly when the corresponding realm file is updated to the new patch.

1.2.5 Patch Data

ddragon does not provide data for when patches were released. Instead, you can find this data on cdragon's github. See the readme on github for information about how to use the file.

1.2.6 Languages

The languages supported by ddragon can be found here.

CODE	LANGUAGE
cs_CZ	Czech (Czech Republic)
el_GR	Greek (Greece)
pl_PL	Polish (Poland)
ro_RO	Romanian (Romania)
hu_HU	Hungarian (Hungary)
en_GB	English (United Kingdom)
de_DE	German (Germany)
es_ES	Spanish (Spain)
it_IT	Italian (Italy)
fr_FR	French (France)
ja_JP	Japanese (Japan)
ko_KR	Korean (Korea)
es_MX	Spanish (Mexico)
es_AR	Spanish (Argentina)
pt_BR	Portuguese (Brazil)
en_US	English (United States)
en_AU	English (Australia)
ru_RU	Russian (Russia)
tr_TR	Turkish (Turkey)
ms_MY	Malay (Malaysia)
en_PH	English (Republic of the Philippines)
en_SG	English (Singapore)
th_TH	Thai (Thailand)
vn_VN	Vietnamese (Viet Nam)
id_ID	Indonesian (Indonesia)
zh_MY	Chinese (Malaysia)
zh_CN	Chinese (China)
zh_TW	Chinese (Taiwan)

1.3 Your Application

You can apply for a personal or production app by clicking "Register Project" on the main dev portal page.

Do not apply for a project if you just want to test the API. If you are developing a project or just testing things out, use your development key that Riot gives you automatically when you signed up. This key expires every 24 hours, so you need to refresh it every day that you work on your app. This is deliberate functionality and your project will not be given a non-development key if all you are doing is testing the API. Even the Rioters use development keys that expire every 24 hours.

Applications take ~ two weeks (10 business days, barring holidays) to process. Make sure it's verified (riot.txt). If it's been longer than two weeks and you didn't change your application during that time period, ask a Guru on Discord for help.

You can find your app ID by going to https://developer.riotgames.com/apps, selecting your project from the list on the left, and copying the number in the URL (https://developer.riotgames.com/app/YOURAPPIDISHERE/info).

1.3.1 Production App

Register your project to apply for a Production API Key with access to the Standard APIs and/or Tournaments API. A Production API key requires a working prototype. Production API keys are not for testing purposes. You may submit an application in the planning stage of your project, but your application for a production key won't be approved until your project is ready for public consumption. Work in progress should be tested using your demo key.

1.3.2 Personal App

Register your project to apply for a Personal API Key without the verification process. This is only for the Standard API's, not Tournaments. Personal Keys require a detailed description ofthe project.

1.4 PUUIDs and Other IDs

The Riot API uses three IDs for players: summoner IDs, account IDs, and PUUIDs. Different APIs use different IDs, and you should use whichever ID is required by the API you are using.

Summoner and account IDs are only unique per region, and PUUIDs are unique globally.

Because the PUUID is globally unique, when a player transfers regions their PUUID will not change. This allows you to track summoners that have transferred regions. If a player transfers regions, their summoner and account IDs will change.

All IDs are encrypted using encryption keys unique to each project. An ID obtained with your dev key will not work with your production key (and vice versa). When you refresh a key, the encrypted ID will not change.

1.5 LCU - The League Client

The LCU, which stands for "League Client Update" is the League Client. You use parts of the LCU for your projects.

This page contains up-to-date information for which endpoints you are allowed to use.

It's important that you create an application on the dev portal telling Riot how you are using the LCU in your project. The more information they have about how people are using the LCU, the better feedback they can give to other teams at Riot.

The LCU must be running on your computer, and you must be logged in, in order to use the LCU endpoints.

1.5.1 Rift Explorer

Pupix on Discord created an excellent app called Rift Explorer to explore the LCU endpoints. You can download it here.

Vivi on Discord create a website that allows you to view the swagger for the LCU endpoints online.

1.5.2 Tips

Starting the LCU with the flag `--mode unattended` will allow you to use some of the LCU endpoints without logging in. This is convenient for running the LCU on a headless (no-monitor) server.

You can create a normal game mode lobby by passing the queue ID (`{"queueId":430}`) to the `/lol-lobby/v2/` POST endpoint.

To create a custom game lobby, use the below POST request. Change `PRACTICETOOL` to `CLASSIC` if you don't want it to be a practice game.

```
POST
/lol-lobby/v2/lobby
{
  "customGameLobby": {
    "configuration": {
      "gameMode": "PRACTICETOOL", "gameMutator": "", "gameServerRegion": "", "mapId": 11, "mutators": {"id": 1}, "spectatorPolicy": "AllAllowed", "teamSize": 5
    },
    "lobbyName": "Name",
    "lobbyPassword": null
  },
  "isCustom": true
}
```

These flags are useful: `--app-port=1337 --headless --allow-multiple-clients`.

If you are running on Linux with Wine, you need to set the Wine prefix to Windows XP and have wine-staging for compatibility with the anti cheat.

1.6 Mobile Apps (& CORS Error)

Client-side calls to the Riot API are blocked because there is no way to make them without exposing your API key to users. You will need to set up a backend server that can make API calls while keeping your API key secure. If you want to quickly set up a proxy server, you can create some functions with AWS Lambda or use Kernel.

If your project is only intended for personal usage, you can modify your browser settings to disable CORS (the mechanism which blocks you from making client-side calls). However, you will not be allowed to publish your site if you do this. More info about disabling CORS can be found here.

You can create a mobile app that uses the API, but you'll need to create a webpage to explain the app, and a server to proxy your API calls. The website should contain enough information so that Rioters can evaluate your project (to ensure it isn't breaking any rules) without actually downloading it. You'll also need to set up a server to make your API calls from. You cannot securely store your API key in a distributed app, so your app will need to communicate with a server, which will in turn communicate with the API.

1.7 Replay API

The Replay API is a new game client API that allows developers to adjust the in-game camera during replays. We have also released League Director, which uses these APIs and will give a good jumping off point for any development.

Because the Replay API is fairly new, the best place to start are the doc pages from Riot about the Replay API and League Director.

1.8 Collecting Data

1.8.1 Match Data

The best way to collect a large set of matches is not straightforward. Typically you need code that does this:

1. Collect a few summoners by hand. Just your own summoner is likely enough, or you could use all challenger players.
2. Get the account IDs of those summoners.
3. Get their match history on the queue type you want (paginate as far as you want to go, or filter by patch).
4. Success! You have a ton of match IDs, from there, you can fetch their timelines or parse the matches however you want.

Alternatively, Canisback on the Discord currently hosts a list of matches IDs that you can use to pull summoners from the *matches/{matchId}* endpoints. These lists are provided for free to the community for use, and may go down or stop being updated at any time.

1.8.2 Summoner Data

You can collect summoner data similar to how you collect match data (see above).

You can also use the League endpoints to get lists of ranked summoners. The positional league endpoints provide a paginated list of all summoners in a Tier + Division + Position (e.g. all ranked Diamond II Top laners). Alternatively, Canisback on the Discord currently hosts a list of league IDs that you can use to pull summoners from the *leagues/{leagueId}* endpoints. These lists are provided for free to the community for use, and may go down or stop being updated at any time.

1.9 Info About Specific Data

1.9.1 Summoners

- See the *PUUIDs and Other IDs* section for information about summoner IDs, account IDs, and PUUIDs.

1.9.2 Tournaments

—

1.9.3 Match

- Matches are kept for 2 years, timelines for 1 year.
- The creation date timestamps in milliseconds (not seconds).
- The pick order of participants is 1-5-2-6-3-7-4-8-5-9.
- The `totalGame` fields is incorrect in most cases. It's best practice to just ignore it.
- Season ID's should only be used if you're looking for data that's over a year old, as it can take months for the ID's to be updated for the current season. Because of this, it's best to use timestamps instead. You can find the approximate start time for each season here.
- You can find seed data for matches at these urls: https://s3-us-west-1.amazonaws.com/riot-developer-portal/seed-data/matches1.json ... https://s3-us-west-1.amazonaws.com/riot-developer-portal/seed-data/matches10.json

Custom Data

- Currently, custom game data (excluding the tournament system), is not retrievable from the API due to privacy policies. Riot is slowly but surely working towards an external RSO (Riot Sign-On) solution that will allow individual players to "Okay" websites to use their custom game data.

1.9.4 Verification

RSO is Riot Sign-On, the login logic you see whenever you access anything that requires your Riot account. RSO is not yet ready for the public to use, but Riot has implemented a work-around for websites to verify users.

Websites should generate a random string for the user to input into their client. See this gif for an illustration. The verification endpoint is notorious for having issues. The user may need to restart their client a few times to get it working, or try again later.

1.9.5 Current Game

- Spectator time is not accurate and inconsistent. See this issue for more details.

1.9.6 Replay Files

- Reverse engineering spectator (.rolf) files is not possible and against the Terms of Service. The encryption of the .rofl files changes each patch. This is to prevent cheating.

1.10 Identifying Champion Positions

1.10.1 What is the problem?

You want to know where each champion played on Summoner's Rift.

The Riot API provides role and lane data, but the values are often inaccurate and make it difficult to identify which position a champion played in. There are many different ways of fixing this data, and how you want to fix the data is likely dependent on your use case.

The definitions of "role", "lane", and "position" are all similar, but for the sake of this explanation we use the Riot API's definition of them. The values Riot provides for Role, Lane, and Position are below:

Role: DUO, DUO_CARRY, DUO_SUPPORT, NONE, and SOLO

Lane: TOP_LANE, MID_LANE, BOT_LANE, and JUNGLE

Position: TOP, MIDDLE, JUNGLE, BOTTOM, UTILITY, APEX, and NONE

1.10.2 Fixing the Role and Lane data

We almost always want to use the Position data in our projects, but Riot only returns Role and Lane in the post-match data.

Below are three ways to calculate "correct" Position data.

A simple mapping

The level of accuracy you need is often dependent on your application. For example, if you're making a website and want to display opponents side by side, it's probably not the end of the world if 1 in 10 games (10%) are incorrect. The easiest (and most inaccurate) way to correct roles is to just use the Role and Lane data and convert to Position based on the below mapping. This yields about 87.5% accuracy.

```
{
    (MID_LANE, SOLO): MIDDLE,
    (TOP_LANE, SOLO): TOP,
    (JUNGLE, NONE): JUNGLE,
    (BOT_LANE, DUO_CARRY): BOTTOM,
    (BOT_LANE, DUO_SUPPORT): UTILITY
}
```

Use playrates

Alternatively, we can use play rate data (e.g. Lux plays mid 30% of the time and support 70% of the time) to determine where each champion on the team played. This method does not use the Lane or Role data at all, and instead relies purely on playrates. This has the advantage that you can calculate the position of each champion before the game has ended, so it works for apps that use the current-game endpoint. Playrate data can be difficult to calculate by yourself, but the champion.gg API provides this data for the current patch. If you want playrate data for previous patches, you'll have to calculate it yourself by pulling a bunch of matches from the Riot API and calculating the statistics. An implementation of this algorithm can be found here: https://github.com/meraki-analytics/role-identification. This yields accuracies in the ~ 95% range.

Use timeline data

Finally, we can use most-match data in the timeline objects to identify the position on the map of the champions throughout the game. We can also look at items, runes, summoner spells, etc. to help narrow down what position the champion likely played in. This is easiest implemented using a machine learning approach. Training data can be found here: https://github.com/Canisback/roleML/blob/master/data/verification_results.csv. An implementation of an SVM (a machine learning model) to identify lanes and roles can be found here: https://github.com/meraki-analytics/role-identification/blob/timeline/timeline-roleidentification/finished_timeline_roleID.ipynb that you can use as an example for your own model. A decision tree is another good machine learning algorithm for this task.

Other useful data

You may also want to consider the history of the summoner playing the champion. For example, you might look at their recent games and see that they played mid 80% of the time. There's then a good chance that they are playing mid in other games as well. This won't be a perfect predictor, but this information may improve your method.