olympia Documentation

Release 3.0

Mozilla Addons Team

Aug 05, 2020

CONTENTS

Add-ons Server is the codebase for https://addons.mozilla.org/; the source lives at https://github.com/mozilla/addons-server.

In the past, this project was *olympia*; documentation that refers to olympia refers to this project.

CHAPTER

ONE

CONTENTS

1.1 Security Bug Reports

This code and its associated production web page are included in the Mozilla's web and services bug bounty program. If you find a security vulnerability, please submit it via the process outlined in the program and FAQ pages. Further technical details about this application are available from the Bug Bounty Onramp page.

Please submit all security-related bugs through Bugzilla using the web security bug form. Never submit security-related bugs through a Github Issue or by email.

1.2 External API

This shows you how to use the addons.mozilla.org API at `/api/v4/` which is hosted at the following URLs:

Environment	URL
Production	`https://addons.mozilla.org/api/v4/`
Staging	`https://addons.allizom.org/api/v4/`
Development	`https://addons-dev.allizom.org/api/v4/`

Production Connect to this API for all normal operation.

Staging or Development Connect to these APIs if you need to work with a scratch database or you're testing features that aren't available in production yet. Your production account is not linked to any of these APIs.

Dive into the *overview section* and the *authentication section* for an example of how to get started using the API.

1.2.1 Overview

Note: These APIs are not frozen and can change at any time without warning. See *the API versions available* for details if you need stability.

This describes the details of the requests and responses you can expect from the addons.mozilla.org API.

Requests

All requests should be made with the header:

```
Content-type: application/json
```

Responses

Status Codes

There are some common API responses that you can expect to receive at times.

GET /api/v4/...

Status Codes

- 200 OK – Success.
- 201 Created – Creation successful.
- 202 Accepted – The request has been accepted for processing. This usually means one or more asyncronous tasks is being executed in the background so results aren't immediately visible.
- 204 No Content – Success (no content is returned).
- 400 Bad Request – There was a problem with the parameters sent with this request.
- 401 Unauthorized – Authentication is required or failed.
- 403 Forbidden – You are not permitted to perform this action.
- 404 Not Found – The requested resource could not be found.
- 500 Internal Server Error – An unknown error occurred.
- 503 Service Unavailable – The site is in maintenance mode at this current time and the operation can not be performed.

Bad Requests

When returning a `HTTP 400 Bad Request` response, the API will try to return some information about the error(s) in the body of the response, as a JSON object. The keys of that object indicate the field(s) that caused an error, and for each, a list of messages will be provided (often only one message will be present, but sometimes more). If the error is not attached to a specific field the key `non_field_errors` will be used instead.

Example:

```
{
    "username": ["This field is required."],
    "non_field_errors": ["Error not linked to a specific field."]
}
```

Unauthorized and Permission Denied

When returning `HTTP 401 Unauthorized` and `HTTP 403 Permission Denied` responses, the API will try to return some information about the error in the body of the response, as a JSON object. A `detail` property will contain a message explaining the error. In addition, in some cases, an optional `code` property will be present and will contain a constant corresponding to specific problems to help clients address the situation programmatically. The constants are as follows:

Value	Description
ERROR_INVALID_HEADER	The `Authorization` header is invalid.
ERROR_SIGNATURE_EXPIRED	The signature of the token indicates it has expired.
ERROR_DECODING_SIGNATURE	The token was impossible to decode and probably invalid.

Maintainance Mode

When returning `HTTP 503 Service Unavailable` responses the API may be in read-only mode. This means that for a short period of time we do not allow any write requests, this includes `POST`, `PATCH`, `PUT` and `DELETE` requests.

In case we are in read-only mode, the following behavior can be observed:

- `GET` requests behave normally
- `POST`, `PUT`, `PATCH`, and `DELETE` requests return 503 with a json response that contains a localized error message

The response when returning `HTTP 503 Service Unavailable` in case of read-only mode looks like this:

```
{
    "error": "Some features are temporarily disabled while we perform websi..."
}
```

In case we are not in read-only mode everything should be back working as normal. To check if the site is in read-only mode before submitting a response, the *site status api* can be called.

Pagination

By default, all endpoints returning a list of results are paginated. The default number of items per page is 25 and clients can use the *page_size* query parameter to change it to any value between 1 and 50. Exceptions to those rules are possible but will be noted in the corresponding documentation for affected endpoints.

The following properties will be available in paginated responses:

- *next*: the URL for the next page in the pagination.
- *previous*: the URL for the previous page in the pagination.
- *page_size*: The number of items per page in the pagination.
- *page_count*: The number of pages available in the pagination. It may be lower than *count / page_size* for elasticsearch based paginations that go beyond our *max_result_window* configuration.
- *count*: the total number of records.
- *results*: the array containing the results for this page.

Translated Fields

Fields that can be translated by users (typically name, description) have a special behaviour. They are returned as an object, by default, with languages as keys and translations as values, and by default all languages are returned:

```
{
    "name": {
        "en-US": "Games",
        "fr": "Jeux",
        "kn": ""
    }
}
```

However, for performance, if you pass the `lang` parameter to a `GET` request, then only the most relevant translation (the specified language or the fallback, depending on whether a translation is available in the requested language) will be returned.

Default API behavior

In API version 3 or 4 the response, if the `lang` parameter is passed, is a single string.

```
{
    "name": "Games"
}
```

This behaviour also applies to `POST`, `PATCH` and `PUT` requests: you can either submit an object containing several translations, or just a string. If only a string is supplied, it will only be used to translate the field in the current language.

v5 API behavior

In the experimental *v5 API* the response, if the `lang` parameter is passed, is an object containing only that translation.

```
{
    "name": {
        "en-US": "Games"
    }
}
```

For `POST`, `PATCH` and `PUT` requests you submit an object containing translations for any languages needing to be updated/saved. Any language not in the object is not updated, but is not removed.

For example, if there were existing translations of:

```
"name": {"en-US": "Games", "fr": "Jeux","kn": ""}
```

and the following request was made:

```
{
    "name": {
        "en-US": "Fun"
    }
}
```

Then the resulting translations would be:

```
"name": {"en-US": "Fun", "fr": "Jeux","kn": ""}
```

To delete a translation, pass `null` as the value for that language. (Note: this behavior is currently buggy/broken - see https://github.com/mozilla/addons-server/issues/8816 for more details)

Outgoing Links

If the `wrap_outgoing_links` query parameter is present, any external links returned for properties such as `support_url` or `homepage` will be wrapped through `outgoing.prod.mozaws.net`. Fields supporting some HTML, such as add-on `description`, always do this regardless of whether or not the query parameter is present.

Cross Origin

All APIs are available with Cross-Origin Resource Sharing unless otherwise specified.

Site Status

This special endpoint returns if the site is in read only mode, and if there is a site notice currently in effect. See *maintainance mode* for more details of when the site is read only and how requests are affected.

GET /api/v4/site/

Response JSON Object

- **read_only** (*boolean*) – Whether the site in read-only mode.
- **notice** (*string|null*) – A site-wide notice about any current known difficulties or restrictions. If this API is being consumed by a tool/frontend it should be displayed to the user.

API Versions

Default v4 API

All documentation here, unless otherwise specified, refers to the default *v4* APIs, which are considered stable. The request and responses are *NOT* frozen though, and can change at any time, depending on the requirements of addons-frontend (the primary consumer).

Frozen v3 API

Any consumer of the APIs that requires more stablity may consider using the *v3* API instead, which is frozen. No new API endpoints (so no new features) will be added to *v3* and we aim to make no breaking changes. Despite the aim, we can't guarantee 100% stability. The *v3* API will be maintained for as long as Firefox ESR60 is supported by Mozilla, i.e. at least October 23rd 2019.

The documentation for *v3* can be accessed at: *External API (V3)*

Experimental v5 API

The experimental *v5* API contains some additional changes/features which are either unstable, in-progress, or currently unsupported by addons-frontend. It will eventually become the new default API when the current default, *v4*, is frozen and the stable *v3* deprecated. Any reference to v5 specific behavior/properties/endpoints will be explicit in these docs.

v4 API changelog

- 2018-05-18: renamed /reviews/ endpoint to /ratings/ https://github.com/mozilla/addons-server/issues/6849
- 2018-05-25: renamed `rating.rating` property to `rating.score` https://github.com/mozilla/addons-server/pull/8332
- 2018-06-05: dropped `rating.title` property https://github.com/mozilla/addons-server/issues/8144
- 2018-07-12: added `type` property to autocomplete API. This change was also backported to the *v3* API. https://github.com/mozilla/addons-server/issues/8803
- 2018-07-19: localised field values are always returned as objects, even if only a single language is requested. Setting a localised value with a string is removed too - it must always be an object of one or more translations. https://github.com/mozilla/addons-server/issues/8794
- 2018-07-18: added `previews` property to discovery API `addons` object. This change was also backported to the *v3* API. https://github.com/mozilla/addons-server/issues/8863
- 2018-07-20: dropped `downloads` property from the collection add-ons results. https://github.com/mozilla/addons-server/issues/8944
- 2018-08-16: added `is_developer_reply` property to ratings. This changed was also backported to the *v3* API. https://github.com/mozilla/addons-server/issues/8993
- 2018-09-13: added `name` and `icon_url` properties to the `addon` object in ratings. This changed was also backported to the *v3* API. https://github.com/mozilla/addons-server/issues/9357
- 2018-09-27: backed out "localised field values are always returned as objects" change from 2018-07-19 from *v4* API. This is intended to be temporary change while addons-frontend upgrades. On addons-dev and addons stage enviroments the previous behavior is available as *api/v4dev*. The *v4dev* api is not available on AMO production server. https://github.com/mozilla/addons-server/issues/9467
- 2018-10-04: added `is_strict_compatibility_enabled` to discovery API `addons.current_version` object. This change was also backported to the *v3* API. https://github.com/mozilla/addons-server/issues/9520
- 2018-10-04: added `is_deleted` to the ratings API. This change was also backported to the *v3* API. https://github.com/mozilla/addons-server/issues/9371
- 2018-10-04: added `exclude_ratings` parameter to ratings API. This change was also backported to the *v3* API. https://github.com/mozilla/addons-server/issues/9424
- 2018-10-11: removed `locale_disambiguation` from the Language Tools API.
- 2018-10-11: added `created` to the addons API.
- 2018-10-18: added `_score` to the addons search API.
- 2018-10-25: changed `author` parameter on addons search API to accept user ids as well as usernames. This change was also backported to the *v3* API. https://github.com/mozilla/addons-server/issues/8901
- 2018-10-25: added `fxa_edit_email_url` parameter on accounts API to return the full URL for editing the user's email on FxA. https://github.com/mozilla/addons-server/issues/8674

- 2018-10-31: added `id` to discovery API `addons.current_version` object. This change was also backported to the *v3* API. https://github.com/mozilla/addons-server/issues/9855
- 2018-11-15: added `is_custom` to the license object in version detail output in the addons API.
- 2018-11-22: added `flags` to the rating object in the ratings API when `show_flags_for` parameter supplied.
- 2018-11-22: added `score` parameter to the ratings API list endpoint.
- 2019-01-10: added `release_notes` and `license` (except `license.text`) to search API results `current_version` objects.
- 2019-01-11: added new /reviewers/browse/ endpoint. https://github.com/mozilla/addons-server/issues/10322
- 2019-01-16: removed /api/{v3,v4,v5}/github api entirely. They have been marked as experimental. https://github.com/mozilla/addons-server/issues/10411
- 2019-02-21: added new /api/v4/reviewers/addon/(addon_id)/versions/ endpoint. https://github.com/mozilla/addons-server/issues/10432
- 2019-03-14: added new /reviewers/compare/ endpoint. https://github.com/mozilla/addons-server/issues/10323
- 2019-04-11: removed `id`, `username` and `url` from the `user` object in the activity review notes APIs. https://github.com/mozilla/addons-server/issues/11002
- 2019-04-18: added new optional parameters to abuse report endpoint
- 2019-05-09: added `is_recommended` to addons API. https://github.com/mozilla/addons-server/issues/11278
- 2019-05-16: added /reviewers/canned-responses/ endpoint. https://github.com/mozilla/addons-server/issues/11276
- 2019-05-23: added `is_recommended` to addons autocomplete API also. https://github.com/mozilla/addons-server/issues/11439
- 2019-05-23: changed the addons search API default sort when no query string is passed - now `sort=recommended,downloads`. Also made `recommended` sort available generally to the addons search API. https://github.com/mozilla/addons-server/issues/11432
- 2019-06-27: removed `sort` parameter from addon autocomplete API. https://github.com/mozilla/addons-server/issues/11664
- 2019-07-18: completely changed the 2019-05-16 added draft-comment related APIs. See #11380, #11379, #11378 and #11374
- 2019-07-25: added /hero/ endpoint to expose recommended addons and other content to frontend to allow customizable promos https://github.com/mozilla/addons-server/issues/11842.
- 2019-08-01: added alias `edition=MozillaOnline` for `edition=china` in /discovery/ endpoint.
- 2019-08-08: add support for externally hosted addons to /hero/ endpoints. https://github.com/mozilla/addons-server/issues/11882
- 2019-08-08: removed `heading_text` property from discovery api. https://github.com/mozilla/addons-server/issues/11817
- 2019-08-08: add secondary shelf to /hero/ endpoint. https://github.com/mozilla/addons-server/issues/11779
- 2019-08-15: dropped support for LWT specific statuses.
- 2019-08-15: added promo modules to secondary hero shelves. https://github.com/mozilla/addons-server/issues/11780
- 2019-08-15: removed /addons/compat-override/ from v4 and above. Still exists in /v3/ but will always return an empty response. https://github.com/mozilla/addons-server/issues/12063

- 2019-08-22: added `canned_response` property to draft comment api. https://github.com/mozilla/addons-server/issues/11807
- 2019-09-19: added /site/ endpoint to expose read-only mode and any site notice. Also added the same response to the /accounts/account/ non-public response as a convenience for logged in users. https://github.com/mozilla/addons-server/issues/11493)
- 2019-10-17: moved /authenticate endpoint from api/v4/accounts/authenticate to version-less api/auth/authenticate-callback https://github.com/mozilla/addons-server/issues/10487
- 2019-11-14: removed `is_source_public` property from addons API https://github.com/mozilla/addons-server/issues/12514
- 2019-12-05: removed /addons/featured endpoint from v4+ and featured support from other addon api endpoints. https://github.com/mozilla/addons-server/issues/12937
- 2020-01-23: added /scanner/results (internal API endpoint).
- 2020-02-06: added /reviewers/addon/(int:addon_id)/allow_resubmission/ and /reviewers/addon/(int:addon_id)/deny_resubmission/. https://github.com/mozilla/addons-server/issues/13409
- 2020-02-20: added `addon_install_source_url` to abuse report endpoint
- 2020-03-19: added /blocklist/block endpoint to expose add-on blocks https://github.com/mozilla/addons-server/issues/13706.
- 2020-03-26: added `addon_name` to blocklist/block api https://github.com/mozilla/addons-server/issues/13757

v5 API changelog

These are *v5* specific changes - *v4* changes apply also.

- 2018-09-27: created the *v4dev* API. The *v4dev* api is not available on AMO production server. See *translations* for details on the change to responses containing localisations. https://github.com/mozilla/addons-server/issues/9467
- 2019-05-09: renamed the experimental *v4dev* api to *v5* and made the *v5* API generally available (on AMO production also)

1.2.2 Authentication (External)

To access the API as an external consumer, you need to include a JSON Web Token (JWT) in the `Authorization` header for every request. This header acts as a one-time token that authenticates your user account. No JWT claims are made about the actual API request you are making.

If you are building an app that lives on the AMO domain, read the *documentation for internal authentication* instead.

Access Credentials

To create JWTs, first obtain a **key** and **secret** from the API Credentials Management Page.

Note: Keep your API keys secret and *never* commit them to a public code repository or share them with anyone, including Mozilla contributors.

If someone obtains your secret they can make API requests on behalf of your user account.

Create a JWT for each request

Prior to making every API request, you need to generate a fresh JWT. The JWT will have a short expiration time and is only valid for a single request so you can't cache or reuse it. You only need to include a few standard fields; here's what the raw JSON object needs to look like before it's signed:

```
{
    "iss": "your-api-key",
    "jti": "0.47362944623455405",
    "iat": 1447273096,
    "exp": 1447273156
}
```

iss This is a standard JWT claim identifying the *issuer*. Set this to the **API key** you generated on the credentials management page. For example: `user:543210:23`.

jti This is a standard JWT claim declaring a *JWT ID*. This value needs to have a high probability of being unique across all recent requests made by your issuer ID. This value is a type of cryptographic nonce designed to prevent replay attacks.

iat This is a standard JWT claim indicating the *issued at time*. It should be a Unix epoch timestamp and **must be in UTC time**.

exp This is a standard JWT claim indicating the *expiration time*. It should be a Unix epoch timestamp in UTC time and must be **no longer than five minutes** past the issued at time.

> Changed in version 2016-10-06: We increased the expiration time from 60 seconds to five minutes to workaround support for large and slow uploads.

Note: If you're having trouble authenticating, make sure your system clock is correct and consider synchronizing it with something like tlsdate.

Take this JSON object and sign it with the **API secret** you generated on the credentials management page. You must sign the JWT using the `HMAC-SHA256` algorithm (which is typically the default). The final JWT will be a blob of base64 encoded text, something like:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↪eyJpc3MiOiJ5b3VyLWFwaS1rZXkiLCJpYXQiOjE0NDcyNzMwOTYsImp0aSI6IjAuNDczNjI5NDQ2MjM0NTU0MDUiLCJleHAiOjE
↪TQ4B8GEm7UWZPcHuNGgjzD8EU9oUBVbL70Le1IeuYx0
```

Note: See jwt.io debugger for more information about the token.

Enter *secret* in the "VERIFY SIGNATURE" section to correctly verify the signature.

Here is an example of creating a JWT in NodeJS using the node-jsonwebtoken library:

```
var jwt = require('jsonwebtoken');

var issuedAt = Math.floor(Date.now() / 1000);
var payload = {
  iss: 'your-api-key',
  jti: Math.random().toString(),
  iat: issuedAt,
  exp: issuedAt + 60,
};
```

(continues on next page)

(continued from previous page)

```
var secret = 'your-api-secret';  // store this securely.
var token = jwt.sign(payload, secret, {
  algorithm: 'HS256',  // HMAC-SHA256 signing algorithm
});
```

Create an Authorization header

When making each request, put your generated JSON Web Token (JWT) into an HTTP Authorization header prefixed with `JWT`, like this:

```
Authorization: JWT eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↪eyJpc3MiOiJ5b3VyLWFwaS1rZXkiLCJpYXQiOjE0NDcyNzMwOTYsImp0aSI6IjAuNDczNjI5NDQ2MjM0NTU0MDUiLCJleHAiOjI
↪TQ4B8GEm7UWZPcHuNGgjzD8EU9oUBVbL70Le1IeuYx0
```

Example request

Using the *profile* as an example endpoint, here's what a JWT authenticated HTTP request would look like in curl:

```
curl "https://addons.mozilla.org/api/v4/accounts/profile/" \
     -H "Authorization: JWT eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↪eyJpc3MiOiJ5b3VyLWFwaS1rZXkiLCJpYXQiOjE0NDcyNzMwOTYsImp0aSI6IjAuNDczNjI5NDQ2MjM0NTU0MDUiLCJleHAiOjI
↪TQ4B8GEm7UWZPcHuNGgjzD8EU9oUBVbL70Le1IeuYx0"
```

Find a JWT library

There are robust open source libraries for creating JWTs in all major programming languages.

1.2.3 Authentication (internal)

This documents how to use authentication in your API requests when you are working on a web application that lives on AMO domain or subdomain. If you are looking for how to authenticate with the API from an external client, using your API keys, read the *documentation for external authentication* instead.

When using this authentication mechanism, the server is responsible for creating an API Token when the user logs in, and sends it back in the response. The clients must then include that token as an `Authorization` header on requests that need authentication. The clients never generate JWTs themselves.

Fetching the token

A fresh token, valid for 30 days, is automatically generated and added to the responses of the following endpoint:

- `/api/auth/authenticate-callback/`

The token is available in two forms:

- For the endpoint mentioned above, as a property called `token`.
- For all endpoints, as a cookie called `frontend_auth_token`. This cookie expires after 30 days and is set as `HttpOnly`.

Creating an Authorization header

When making an authenticated API request, put your generated API Token into an HTTP Authorization header prefixed with `Bearer`, like this:

```
Authorization: Bearer␣
↪eyJhdXRoX2hhc2giOiJiY2E0MTZkN2RiMGU3NjFmYTA2NDE4MjAzZWU1NTMwOTM4OGZhNzcxIiwidXNlcl9pZCI6MTIzNDV9:1c
↪gNo3EWU8IfL8
```

1.2.4 Abuse Reports

Note: These APIs are not frozen and can change at any time without warning. See *the API versions available* for alternatives if you need stability.

The following API endpoint covers abuse reporting

Submitting an add-on abuse report

The following API endpoint allows an abuse report to be submitted for an Add-on, either listed on https://addons.mozilla.org or not. Authentication is not required, but is recommended so reports can be responded to if necessary.

Warning: Except for the `message`, all strings have a maximum length of 255 characters and should be truncated by the client where necessary.

Warning: For `addon_install_method` and `addon_install_source` specifically, if an unsupported value is sent, it will be silently changed to `other` instead of raising a 400 error.

`POST /api/v4/abuse/report/addon/`

Request JSON Object

- **`addon`** (*`string`*) – The id, slug, or guid of the add-on to report for abuse (required).
- **`message`** (*`string`*) – The body/content of the abuse report (required).
- **`report_entry_point`** (*`string|null`*) – The report entry point. The accepted values are documented in the *table below*.
- **`addon_install_method`** (*`string|null`*) – The add-on install method. The accepted values are documented in the *table below*.
- **`addon_install_origin`** (*`string|null`*) – The add-on install origin.
- **`addon_install_source`** (*`string|null`*) – The add-on install source. The accepted values are documented in the *table below*.
- **`addon_install_source_url`** (*`string|null`*) – The add-on install source URL.
- **`addon_name`** (*`string|null`*) – The add-on name in the locale used by the client.
- **`addon_signature`** (*`string|null`*) – The add-on signature state. The accepted values are documented in the *table below*.

- **addon_summary** (*string|null*) – The add-on summary in the locale used by the client.
- **addon_version** (*string|null*) – The add-on version string.
- **app** (*string|null*) – The *application* used by the client. Can be either `firefox` or `android`.
- **appversion** (*string|null*) – The application version used by the client.
- **lang** (*string|null*) – The language code of the locale used by the client for the application.
- **client_id** (*string|null*) – The client's hashed telemetry ID.
- **install_date** (*string|null*) – The add-on install date.
- **operating_system** (*string|null*) – The client's operating system.
- **operating_system_version** (*string|null*) – The client's operating system version.
- **reason** (*string|null*) – The reason for the report. The accepted values are documented in the *table below*.

Response JSON Object

- **reporter** (*object|null*) – The user who submitted the report, if authenticated.
- **reporter.id** (*int*) – The id of the user who submitted the report.
- **reporter.name** (*string*) – The name of the user who submitted the report.
- **reporter.username** (*string*) – The username of the user who submitted the report.
- **reporter.url** (*string*) – The link to the profile page for of the user who submitted the report.
- **addon** (*object*) – The add-on reported for abuse.
- **addon.guid** (*string*) – The add-on extension identifier.
- **addon.id** (*int|null*) – The add-on id on AMO. If the guid submitted didn't match a known add-on on AMO, then null.
- **addon.slug** (*string|null*) – The add-on slug. If the guid submitted didn't match a known add-on on AMO, then null.
- **message** (*string*) – The body/content of the abuse report.
- **report_entry_point** (*string|null*) – The report entry point.
- **addon_install_method** (*string|null*) – The add-on install method.
- **addon_install_origin** (*string|null*) – The add-on install origin.
- **addon_install_source** (*string|null*) – The add-on install source.
- **addon_install_source_url** (*string|null*) – The add-on install source URL.
- **addon_name** (*string|null*) – The add-on name in the locale used by the client.
- **addon_signature** (*string|null*) – The add-on signature state.
- **addon_summary** (*string|null*) – The add-on summary in the locale used by the client.
- **addon_version** (*string|null*) – The add-on version string.

- **app** (*string|null*) – The application used by the client.
- **appversion** (*string|null*) – The application version used by the client.
- **lang** (*string|null*) – The language code of the locale used by the client for the application.
- **client_id** (*string|null*) – The client's hashed telemetry ID.
- **install_date** (*string|null*) – The add-on install date.
- **operating_system** (*string|null*) – The client's operating system.
- **operating_system_version** (*string|null*) – The client's operating system version.
- **reason** (*string|null*) – The reason for the report.

Accepted values for the `report_entry_point` parameter:

Value	Description
uninstall	Report button shown at uninstall time
menu	Report menu in Add-ons Manager
toolbar_context_menu	Report context menu on add-on toolbar
amo	Report button on an AMO page (using `navigator.mozAddonManager.reportAbuse`)

Accepted values for the `addon_install_method` parameter:

Note: This should match what is documented for `addonsManager.install.extra_keys.method` in Firefox telemetry event definition except that the values are normalized by being converted to lowercase with the `:` and `-` characters converted to `_`. In addition, extra values are supported for backwards-compatibility purposes, since Firefox before version 70 merged source and method into the same value. If an unsupported value is sent for this parameter, it will be silently changed to special `other` instead of raising a 400 error.

Value	Description
amwebapi	Add-on Manager Web API
link	Direct Link
installtrigger	InstallTrigger API
install_from_file	Local File
management_webext_api	WebExt Management API
drag_and_drop	Drag & Drop
sideload	Sideload
file_url	File URL
url	URL
other	Other
enterprise_policy	Enterprise Policy (obsolete, for backwards-compatibility)
distribution	Included in build (obsolete, for backwards-compatibility)
system_addon	System Add-on (obsolete, for backwards-compatibility)
temporary_addon	Temporary Add-on (obsolete, for backwards-compatibility)
sync	Sync (obsolete, for backwards-compatibility)

Accepted values for the `addon_install_source` parameter:

Note: This should match what is documented for `addonsManager.install.extra_keys.method` in Firefox telemetry event definition except that the values are normalized by being converted to lowercase with the `:` and `-` characters converted to `_`. We support the additional `other` value as a catch-all. If an unsupported value is sent for this parameter, it will be silently changed to `other` instead of raising a 400 error.

Value	Description
about_addons	Add-ons Manager
about_debugging	Add-ons Debugging
about_preferences	Preferences
amo	AMO
app_profile	App Profile
disco	Disco Pane
distribution	Included in build
extension	Extension
enterprise_policy	Enterprise Policy
file_url	File URL
gmp_plugin	GMP Plugin
internal	Internal
plugin	Plugin
rtamo	Return To AMO
sync	Sync
system_addon	System Add-on
temporary_addon	Temporary Add-on
unknown	Unknown
other	Other

Accepted values for the `addon_signature` parameter:

Value	Description
curated_and_partner	Curated and partner
curated	Curated
partner	Partner
non_curated	Non-curated
unsigned	Unsigned
broken	Broken
unknown	Unknown
missing	Missing
preliminary	Preliminary
signed	Signed
system	System
privileged	Privileged

Accepted values for the `reason` parameter:

Value	Description
damage	Damages computer and/or data
spam	Creates spam or advertising
settings	Changes search / homepage / new tab page without informing user
broken	Doesn't work, breaks websites, or slows Firefox down
policy	Hateful, violent, or illegal content
deceptive	Doesn't match description
unwanted	Wasn't wanted / impossible to get rid of
other	Something else

Submitting a user abuse report

The following API endpoint allows an abuse report to be submitted for a user account on https://addons.mozilla.org. Authentication is not required, but is recommended so reports can be responded to if necessary.

POST /api/v4/abuse/report/user/

Request JSON Object

- **user** (*string*) – The id or username of the user to report for abuse (required).
- **message** (*string*) – The body/content of the abuse report (required).

Response JSON Object

- **reporter** (*object|null*) – The user who submitted the report, if authenticated.
- **reporter.id** (*int*) – The id of the user who submitted the report.
- **reporter.name** (*string*) – The name of the user who submitted the report.
- **reporter.url** (*string*) – The link to the profile page for of the user who submitted the report.
- **reporter.username** (*string*) – The username of the user who submitted the report.
- **user** (*object*) – The user reported for abuse.
- **user.id** (*int*) – The id of the user reported.
- **user.name** (*string*) – The name of the user reported.
- **user.url** (*string*) – The link to the profile page for of the user reported.
- **user.username** (*string*) – The username of the user reported.
- **message** (*string*) – The body/content of the abuse report.

1.2.5 Accounts

Note: These APIs are not frozen and can change at any time without warning. See *the API versions available* for alternatives if you need stability. The only authentication method available at the moment is *the internal one*.

The following API endpoints cover a users account.

Account

This endpoint returns information about a user's account, by the account id. Only *developer* accounts are publicly viewable - other user's accounts will return a 404 not found response code. Most of the information is optional and provided by the user so may be missing or inaccurate.

A developer is defined as a user who is listed as a developer or owner of one or more approved add-ons.

GET /api/v4/accounts/account/(int:user_id|string:username)/

Response JSON Object

- **average_addon_rating** (*float*) – The average rating for addons the developer has listed on the website.
- **biography** (*string|null*) – More details about the user.
- **created** (*string*) – The date when this user first logged in and created this account.
- **has_anonymous_display_name** (*boolean*) – The user hasn't chosen a name.
- **has_anonymous_username** (*boolean*) – The user hasn't chosen a username.
- **homepage** (*string|null*) – The user's website.
- **id** (*int*) – The numeric user id.
- **is_addon_developer** (*boolean*) – The user has developed and listed add-ons on this website.
- **is_artist** (*boolean*) – The user has developed and listed themes on this website.
- **location** (*string|null*) – The location of the user.
- **name** (*string*) – The name chosen by the user, or the "Firefox user {id}" if not set.
- **num_addons_listed** (*int*) – The number of addons the developer has listed on the website.
- **occupation** (*string|null*) – The occupation of the user.
- **picture_type** (*string|null*) – the image type (only 'image/png' is supported) if a user photo has been uploaded, or null otherwise.
- **picture_url** (*string|null*) – URL to a photo of the user, or null if no photo has been uploaded.
- **username** (*string*) – deprecated property still included for backwards compatability. Previous chosen by the user, used in the account url. If not previously set will be a randomly generated string.

If you authenticate and access your own account by specifing your own `user_id` the following additional fields are returned. You can always access your account, regardless of whether you are a developer or not. If you have *Users:Edit* permission you will see these extra fields for all user accounts.

GET /api/v4/accounts/account/(int:user_id|string:username)/

Response JSON Object

- **deleted** (*boolean*) – Is the account deleted.
- **display_name** (*string|null*) – The name chosen by the user.

- **reviewer_name** (*string|null*) – The reviewer name chosen by the user, if set. Only available for users with any kind of reviewer permission.
- **email** (*string*) – Email address used by the user to login and create this account.
- **fxa_edit_email_url** (*string*) – The configured URL for editing the user's email on FxA.
- **last_login** (*string*) – The date of the last successful log in to the website.
- **last_login_ip** (*string*) – The IP address of the last successfull log in to the website.
- **is_verified** (*boolean*) – The user has been verified via FirefoxAccounts.
- **permissions** (*array*) – A list of the additional *permissions* this user has.
- **read_dev_agreement** (*boolean*) – The user has read, and agreed to, the developer agreement that is required to submit addons.
- **site_status** (*object*) – The *site status* - exposed here as a convenience to avoid an extra api call for logged in users.

Status Codes

- 200 OK – account found.
- 400 Bad Request – an error occurred, check the `error` value in the JSON.
- 404 Not Found – no account with that user id.

Important:

- `Biography` can contain HTML, or other unsanitized content, and it is the responsibiliy of the client to clean and escape it appropriately before display.

Permissions can be any arbritary string in the format *app:action*. Either *app* or *action* can be the wildcard *, so *:* means the user has permission to do all actions (i.e. full admin).

The following are some commonly tested permissions; see https://github.com/mozilla/addons-server/blob/master/src/olympia/constants/permissions.py for the full list.

Value	Description
AdminTools:View	Can access the website admin interface index page. Inner pages may require other/additional permissions.
Addons:Edit	Allows viewing and editing of any add-ons details in developer tools.
Addons:Review	Can access the add-on reviewer tools to approve/reject add-on submissions.
Personas:Review	Can access the theme reviewer tools to approve/reject theme submissions.

Profile

Note: This API requires *authentication*.

This endpoint is a shortcut to your own account. It returns an *account object*

GET /api/v4/accounts/profile/

Edit

Note: This API requires *authentication* and *Users:Edit* permission to edit accounts other than your own.

This endpoint allows some of the details for an account to be updated. Any fields in the *account* (or *self*) but not listed below are not editable and will be ignored in the patch request.

PATCH /api/v4/accounts/account/(int:user_id|string:username)/

Request JSON Object

- **biography** (*string|null*) – More details about the user. No links are allowed.
- **display_name** (*string|null*) – The name chosen by the user. Minimum length is 2, maximum length is 50 characters, and must contain at least 1 displayable character.
- **homepage** (*string|null*) – The user's website.
- **location** (*string|null*) – The location of the user.
- **occupation** (*string|null*) – The occupation of the user.

Uploading a picture

To upload a picture for the profile the request must be sent as content-type *multipart/form-data* instead of JSON. Images must be either PNG or JPG; the maximum file size is 4MB. Other *editable values* can be set at the same time.

PATCH /api/v4/accounts/account/(int:user_id|string:username)/

Request:

```
curl "https://addons.mozilla.org/api/v4/accounts/account/12345/"
    -g -XPATCH --form "picture_upload=@photo.png"
    -H "Authorization: Bearer <token>"
```

Parameters

- **user-id** – The numeric user id.

Form Parameters

- **picture_upload** – The user's picture to upload.

Request Headers

- Content-Type – multipart/form-data

Deleting the picture

To delete the account profile picture call the special endpoint.

DELETE /api/v4/accounts/account/(int:user_id|string:username)/picture

Delete

Note: This API requires *authentication* and *Users:Edit* permission to delete accounts other than your own.

This endpoint allows the account to be deleted. The data will be permanently removed, including profile details (picture, user name, display name, location, home page, biography, occupation), notification preferences, reviews, and collections. If the user authored any add-ons they will also be deleted, unless ownership is shared with other authors. In that case, the user will be removed as an author and the remaining authors will maintain ownership of the add-on.

DELETE /api/v4/accounts/account/(int:user_id|string:username)/

Notifications List

Note: This API requires *authentication* and *Users:Edit* permission to list notifications on accounts other than your own.

This endpoint allows you to list the account notifications set for the specified user. The result is an unpaginated list of the fields below. There are currently 10 notification types.

GET /api/v4/accounts/account/(int:user_id|string:username)/notifications/

Response JSON Object

- **name** (*string*) – The notification short name.
- **enabled** (*boolean*) – If the notification is enabled (defaults to True).
- **mandatory** (*boolean*) – If the notification can be set by the user.

Notifications Update

Note: This API requires *authentication* and *Users:Edit* permission to set notification preferences on accounts other than your own.

This endpoint allows account notifications to be set or updated. The request should be a dict of *name*:TruelFalse pairs. Any number of notifications can be changed; only non-mandatory notifications can be changed - attempting to set a mandatory notification will return an error.

POST /api/v4/accounts/account/(int:user_id|string:username)/notifications/

Request JSON Object

- **<name>** (*boolean*) – Is the notification enabled?

Notification Unsubscribe

This special endpoint is used to handle notification update requests coming from email unsubscribe links. Only a single notification can be changed, and it will always be updated to *enabled* = *False*. Only non-mandatory notifications can be changed - attempting to set a mandatory notification will return an error.

POST /api/v4/accounts/unsubscribe/

Request JSON Object

- **hash** (*string*) – The generated hash of the token
- **notification** (*string*) – The short name of the notification that should be disabled
- **token** (*string*) – The base64 encoded email address of the account

Response JSON Object

- **name** (*string*) – The notification short name.
- **enabled** (*boolean*) – If the notification is enabled (should always be False).
- **mandatory** (*boolean*) – If the notification can be set by the user (should always be False, or an error would have been sent instead).

Super-creation

Note: This API requires *authentication*.

This allows you to generate a new user account and sign in as that user.

Important:

- Your API user must be in the `Accounts:SuperCreate` group to access this endpoint. Use `manage.py createsuperuser --add-to-supercreate-group` to create a superuser with proper access.
- This endpoint is not available in all *API environments*.

POST /api/v4/accounts/super-create/

Request:

Parameters

- **email** – assign the user a specific email address. A fake email will be assigned by default.
- **username** – assign the user a specific username. A random username will be assigned by default.
- **fxa_id** – assign the user a Firefox Accounts ID, like one returned in the `uuid` parameter of a profile request. This is empty by default, meaning the user's account will need to be migrated to a Firefox Account.
- **group** – assign the user to a permission group. Valid choices:
  - **reviewer**: can access add-on reviewer pages, formerly known as Editor Tools
  - **admin**: can access any protected page

```
curl "https://addons.mozilla.org/api/v4/accounts/super-create/" \
    -X POST -H "Authorization: JWT <jwt-token>"
```

Response:

```
{
    "username": "super-created-7ee304ce",
    "display_name": "Super Created 7ee304ce",
    "user_id": 10985,
    "email": "super-created-7ee304ce@addons.mozilla.org",
    "fxa_id": null,
    "groups": [],
    "session_cookie": {
        "encoded": "sessionid=.eJyrVopPLC3JiC8tTi2KT...",
        "name": "sessionid",
        "value": ".eJyrVopPLC3JiC8tTi2KT..."
    }
}
```

Status Codes

- 201 Created – Account created.
- 422 Unprocessable Entity – Incorrect request parameters.

The session cookie will enable you to sign in for a limited time as this new user. You can pass it to any login-protected view like this:

```
curl --cookie sessionid=... -s -D - \
    "https://addons.mozilla.org/en-US/developers/addon/submit/1" \
    -o /dev/null
```

Session

Log out of the current session. This is for use with the *internal authentication* that authenticates browser sessions.

DELETE /api/v4/accounts/session/

Request:

```
curl "https://addons.mozilla.org/api/v4/accounts/session/"
    -H "Authorization: Bearer <jwt-token>" -X DELETE
```

Response:

```
{
    "ok": true
}
```

Status Codes

- 200 OK – session logged out.
- 401 Unauthorized – authentication failed.

1.2.6 Activity

Note: These APIs are not frozen and can change at any time without warning. See *the API versions available* for alternatives if you need stability. The only authentication method available at the moment is *the internal one*.

Review Notes List

This endpoint allows you to list the approval/rejection review history for a version of an add-on.

GET /api/v4/addons/addon/(int:addon_id|string:addon_slug|string:addon_guid)/versions/(int:id)/revie

Note: All add-ons require authentication and either reviewer permissions or a user account listed as a developer of the add-on.

Response JSON Object

- **count** (*int*) – The number of versions for this add-on.
- **next** (*string*) – The URL of the next page of results.
- **previous** (*string*) – The URL of the previous page of results.
- **results** (*array*) – An array of *per version review notes*.

Review Notes Detail

This endpoint allows you to fetch a single review note for a specific version of an add-on.

Note: To allow reviewers to stay anonymous if they wish, the `user` object only contains the name of the reviewer or author. That name may, for some actions, be an alias and not the usual name of the user.

GET /api/v4/addons/addon/(int:addon_id|string:addon_slug|string:addon_guid)/versions/(int:id)/revie

int: *id*/

Response JSON Object

- **id** (*int*) – The id for a review note.
- **action** (*string*) – The *type of review note*.
- **action_label** (*string*) – The text label of the action.
- **user.name** (*string*) – The name of the reviewer or author.
- **comments** (*string*) – The text content of the review note.
- **date** (*string*) – The date the review note was created.

Possible values for the `action` field:

Value	Description
approved	Version, or file in the version, was approved
rejected	Version, or file in the version, was rejected
review-requested	Developer requested review
more-information-requested	Reviewer requested more information from developer
super-review-requested	Add-on was referred to an admin for attention
comment	Reviewer added comment for other reviewers
review-note	Generic review comment

Incoming Mail End-point

This endpoint allows a mail server or similar to submit a json object containing single email into AMO which will be processed. The only type of email currently supported is a reply to an activity email (e.g an add-on review, or a reply to an add-on review). Any other content or invalid emails will be discarded.

`POST /api/v4/activity/mail`

Note: This API endpoint uses a custom authentication method. The value *SecretKey* in the submitted json must match one defined in *settings.INBOUND_EMAIL_SECRET_KEY*. The IP address of the request must match one defined in *settings.ALLOWED_CLIENTS_EMAIL_API*, if defined.

Request JSON Object

- **`SecretKey`** (*`string`*) – A value that matches *settings.INBOUND_EMAIL_SECRET_KEY*.
- **`Message.TextBody`** (*`string`*) – The plain text body of the email.
- **`To`** (*`array`*) – Array of To email addresses. All will be parsed, and the first matching the correct format used.
- **`To[].EmailAddress`** (*`string`*) – An email address in the format *reviewreply+randomuuidstring@addons.mozilla.org*.

1.2.7 Add-ons

Note: These APIs are not frozen and can change at any time without warning. See *the API versions available* for alternatives if you need stability. The only authentication method available at the moment is *the internal one*.

Search

This endpoint allows you to search through public add-ons.

GET /api/v4/addons/search/

Query Parameters

- **q** (*string*) – The search query. The maximum length allowed is 100 characters.
- **app** (*string*) – Filter by *add-on application* availability.
- **appversion** (*string*) – Filter by application version compatibility. Pass the full version as a string, e.g. `46.0`. Only valid when the `app` parameter is also present.
- **author** (*string*) – Filter by exact (listed) author username or user id. Multiple author usernames or ids can be specified, separated by comma(s), in which case add-ons with at least one matching author are returned.
- **category** (*string*) – Filter by *category slug*. `app` and `type` parameters need to be set, otherwise this parameter is ignored.
- **color** (*string*) – (Experimental) Filter by color in RGB hex format, trying to find themes that approximately match the specified color. Only works for static themes.
- **exclude_addons** (*string*) – Exclude add-ons by `slug` or `id`. Multiple add-ons can be specified, separated by comma(s).
- **guid** (*string*) – Filter by exact add-on guid. Multiple guids can be specified, separated by comma(s), in which case any add-ons matching any of the guids will be returned. As guids are unique there should be at most one add-on result per guid specified. For usage with Firefox, instead of separating multiple guids by comma(s), a single guid can be passed in base64url format, prefixed by the `rta:` string.
- **lang** (*string*) – Activate translations in the specific language for that query. (See *translated fields*)
- **page** (*int*) – 1-based page number. Defaults to 1.
- **page_size** (*int*) – Maximum number of results to return for the requested page. Defaults to 25.
- **platform** (*string*) – Filter by *add-on platform* availability.
- **recommended** (*boolean*) – Filter to only add-ons recommended by Mozilla. Only `recommended=true` is supported.
- **tag** (*string*) – Filter by exact tag name. Multiple tag names can be specified, separated by comma(s), in which case add-ons containing *all* specified tags are returned.
- **type** (*string*) – Filter by *add-on type*. Multiple types can be specified, separated by comma(s), in which case add-ons that are any of the matching types are returned.
- **sort** (*string*) – The sort parameter. The available parameters are documented in the *table below*.

Response JSON Object

- **count** (*int*) – The number of results for this query.
- **next** (*string*) – The URL of the next page of results.
- **previous** (*string*) – The URL of the previous page of results.

- **results** (*array*) – An array of *add-ons*. As described below, the following fields are omitted for performance reasons: `release_notes` and `license` fields on `current_version` as well as `picture_url` from `authors`. The special `_score` property is added to each add-on object, it contains a float value representing the relevancy of each add-on for the given query.

Available sorting parameters:

Parameter	Description
created	Creation date, descending.
downloads	Number of weekly downloads, descending.
hotness	Hotness (average number of users progression), descending.
random	Random ordering. Only available when no search query is passed and when filtering to only return recommended add-ons.
rating	Bayesian rating, descending.
recommended	Recommended add-ons above non-recommend add-ons. Only available combined with another sort - ignored on its own. Also ignored if combined with relevance as it already takes into account recommended status.
relevance	Search query relevance, descending. Ignored without a query.
updated	Last updated date, descending.
users	Average number of daily users, descending.

The new default behavior is to sort by relevance if a search query (`q`) is present; otherwise place recommended add-ons first, then non recommended add-ons, then sorted by average daily users, descending. (`sort=recommended,users`). This is the default on AMO dev server.

The default on AMO production currently is to sort by relevance if a search query (`q`) is present; otherwise sort by number of weekly downloads, descending.

You can combine multiple parameters by separating them with a comma. For instance, to sort search results by downloads and then by creation date, use `sort=downloads,created`. The only exception is the `random` sort parameter, which is only available alone.

Autocomplete

Similar to *add-ons search endpoint* above, this endpoint allows you to search through public add-ons. Because it's meant as a backend for autocomplete though, there are a couple key differences:

- No pagination is supported. There are no `next`, `prev` or `count` fields, and passing `page_size` or `page` has no effect, a maximum of 10 results will be returned at all times.
- Only a subset of fields are returned.
- `sort` is not supported. Sort order is always `relevance` if `q` is provided, or the *search default* otherwise.

GET /api/v4/addons/autocomplete/

Query Parameters

- **q** (*string*) – The search query.

- **app** (*string*) – Filter by *add-on application* availability.
- **appversion** (*string*) – Filter by application version compatibility. Pass the full version as a string, e.g. `46.0`. Only valid when the `app` parameter is also present.
- **author** (*string*) – Filter by exact (listed) author username. Multiple author names can be specified, separated by comma(s), in which case add-ons with at least one matching author are returned.
- **category** (*string*) – Filter by *category slug*. `app` and `type` parameters need to be set, otherwise this parameter is ignored.
- **lang** (*string*) – Activate translations in the specific language for that query. (See *translated fields*)
- **platform** (*string*) – Filter by *add-on platform* availability.
- **tag** (*string*) – Filter by exact tag name. Multiple tag names can be specified, separated by comma(s).
- **type** (*string*) – Filter by *add-on type*.

Response JSON Object

- **results** (*array*) – An array of *add-ons*. Only the `id`, `icon_url`, `is_recommended`, `name`, `type` and `url` fields are supported though.

Detail

This endpoint allows you to fetch a specific add-on by id, slug or guid.

Note: Non-public add-ons and add-ons with only unlisted versions require both authentication and reviewer permissions or an account listed as a developer of the add-on.

A 401 or 403 error response will be returned when clients don't meet those requirements. Those responses will contain the following properties:

- `detail`: string containing a message about the error.
- `is_disabled_by_developer`: boolean set to `true` when the add-on has been voluntarily disabled by its developer.
- `is_disabled_by_mozilla`: boolean set to `true` when the add-on has been disabled by Mozilla.

GET /api/v4/addons/addon/(int:id|string:slug|string:guid)/

Query Parameters

- **app** (*string*) – Used in conjunction with `appversion` below to alter `current_version` behaviour. Need to be a valid *add-on application*.
- **appversion** (*string*) – Make `current_version` return the latest public version of the add-on compatible with the given application version, if possible, otherwise fall back on the generic implementation. Pass the full version as a string, e.g. `46.0`. Only valid when the `app` parameter is also present. Currently only compatible with language packs through the add-on detail API, ignored for other types of add-ons and APIs.
- **lang** (*string*) – Activate translations in the specific language for that query. (See *Translated Fields*)

- **wrap_outgoing_links** (*string*) – If this parameter is present, wrap outgoing links through `outgoing.prod.mozaws.net` (See *Outgoing Links*)

Response JSON Object

- **id** (*int*) – The add-on id on AMO.
- **authors** (*array*) – Array holding information about the authors for the add-on.
- **authors[].id** (*int*) – The id for an author.
- **authors[].name** (*string*) – The name for an author.
- **authors[].url** (*string*) – The link to the profile page for an author.
- **authors[].username** (*string*) – The username for an author.
- **authors[].picture_url** (*string*) – URL to a photo of the user, or */static/img/anon_user.png* if not set. For performance reasons this field is omitted from the search endpoint.
- **average_daily_users** (*int*) – The average number of users for the add-on (updated daily).
- **categories** (*object*) – Object holding the categories the add-on belongs to.
- **categories[app_name]** (*array*) – Array holding the *category slugs* the add-on belongs to for a given *add-on application*. (Combine with the add-on `type` to determine the name of the category).
- **contributions_url** (*string*) – URL to the (external) webpage where the addon's authors collect monetary contributions, if set. Can be an empty value.
- **created** (*string*) – The date the add-on was created.
- **current_version** (*object*) – Object holding the current *version* of the add-on. For performance reasons the `license` field omits the `text` property from both the search and detail endpoints.
- **default_locale** (*string*) – The add-on default locale for translations.
- **description** (*string|object|null*) – The add-on description (See *translated fields*).
- **developer comments** (*string|object|null*) – Additional information about the add-on provided by the developer. (See *translated fields*).
- **edit_url** (*string*) – The URL to the developer edit page for the add-on.
- **guid** (*string*) – The add-on extension identifier.
- **has_eula** (*boolean*) – The add-on has an End-User License Agreement that the user needs to agree with before installing (See *add-on EULA and privacy policy*).
- **has_privacy_policy** (*boolean*) – The add-on has a Privacy Policy (See *add-on EULA and privacy policy*).
- **homepage** (*string|object|null*) – The add-on homepage (See *translated fields*).
- **icon_url** (*string*) – The URL to icon for the add-on (including a cachebusting query string).
- **icons** (*object*) – An object holding the URLs to an add-ons icon including a cachebusting query string as values and their size as properties. Currently exposes 32, 64, 128 pixels wide icons.

- **is_disabled** (*boolean*) – Whether the add-on is disabled or not.
- **is_experimental** (*boolean*) – Whether the add-on has been marked by the developer as experimental or not.
- **is_recommended** (*boolean*) – The add-on is recommended by Mozilla.
- **name** (*string|object|null*) – The add-on name (See *translated fields*).
- **last_updated** (*string*) – The date of the last time the add-on was updated by its developer(s).
- **latest_unlisted_version** (*object|null*) – Object holding the latest unlisted *version* of the add-on. This field is only present if the user has unlisted reviewer permissions, or is listed as a developer of the add-on.
- **previews** (*array*) – Array holding information about the previews for the add-on.
- **previews[].id** (*int*) – The id for a preview.
- **previews[].caption** (*string|object|null*) – The caption describing a preview (See *translated fields*).
- **previews[].image_size[]** (*int*) – width, height dimensions of of the preview image.
- **previews[].image_url** (*string*) – The URL (including a cachebusting query string) to the preview image.
- **previews[].thumbnail_size[]** (*int*) – width, height dimensions of of the preview image thumbnail.
- **previews[].thumbnail_url** (*string*) – The URL (including a cachebusting query string) to the preview image thumbnail.
- **public_stats** (*boolean*) – Boolean indicating whether the add-on stats are public or not.
- **ratings** (*object*) – Object holding ratings summary information about the add-on.
- **ratings.count** (*int*) – The total number of user ratings for the add-on.
- **ratings.text_count** (*int*) – The number of user ratings with review text for the add-on.
- **ratings_url** (*string*) – The URL to the user ratings list page for the add-on.
- **ratings.average** (*float*) – The average user rating for the add-on.
- **ratings.bayesian_average** (*float*) – The bayesian average user rating for the add-on.
- **requires_payment** (*boolean*) – Does the add-on require payment, non-free services or software, or additional hardware.
- **review_url** (*string*) – The URL to the reviewer review page for the add-on.
- **slug** (*string*) – The add-on slug.
- **status** (*string*) – The *add-on status*.
- **summary** (*string|object|null*) – The add-on summary (See *translated fields*).
- **support_email** (*string|object|null*) – The add-on support email (See *translated fields*).

- **support_url** (*string|object|null*) – The add-on support URL (See *translated fields*).
- **tags** (*array*) – List containing the text of the tags set on the add-on.
- **type** (*string*) – The *add-on type*.
- **url** (*string*) – The (absolute) add-on detail URL.
- **weekly_downloads** (*int*) – The number of downloads for the add-on in the last week. Not present for lightweight themes.

Possible values for the `status` field / parameter:

Value	Description
public	Fully Reviewed
deleted	Deleted
disabled	Disabled by Mozilla
nominated	Awaiting Full Review
incomplete	Incomplete
unreviewed	Awaiting Preliminary Review

Possible values for the keys in the `compatibility` field, as well as the `app` parameter in the search API:

Value	Description
android	Firefox for Android
firefox	Firefox

Note: For possible version values per application, see valid application versions.

Possible values for the `current_version.files[].platform` field:

Value	Description
all	All
mac	Mac
linux	Linux
android	Android
windows	Windows

Possible values for the `type` field / parameter:

Note: For backwards-compatibility reasons, the value for type of `theme` refers to a deprecated XUL Complete Theme. `persona` are another type of depreated theme. New webextension packaged non-dynamic themes are `statictheme`.

Value	Description
theme	Depreated. Theme (Complete Theme, XUL-based)
search	Search Engine
persona	Deprecated. Theme (Lightweight Theme, persona)
language	Language Pack (Application)
extension	Extension
dictionary	Dictionary
statictheme	Theme (Static Theme)

Add-on and Version Submission

See *Uploading a version*.

Versions List

This endpoint allows you to list all versions belonging to a specific add-on.

GET /api/v4/addons/addon/(int:addon_id|string:addon_slug|string:addon_guid)/versions/

Note: Non-public add-ons and add-ons with only unlisted versions require both:

- authentication
- reviewer permissions or an account listed as a developer of the add-on

---

Query Parameters

- **filter** (*string*) – The *filter* to apply.
- **lang** (*string*) – Activate translations in the specific language for that query. (See *translated fields*)
- **page** (*int*) – 1-based page number. Defaults to 1.
- **page_size** (*int*) – Maximum number of results to return for the requested page. Defaults to 25.

Response JSON Object

- **count** (*int*) – The number of versions for this add-on.
- **next** (*string*) – The URL of the next page of results.
- **previous** (*string*) – The URL of the previous page of results.
- **results** (*array*) – An array of *versions*.

By default, the version list API will only return public versions (excluding versions that have incomplete, disabled, deleted, rejected or flagged for further review files) - you can change that with the `filter` query parameter, which may require authentication and specific permissions depending on the value:

Value	Description
all_without_unlisted	Show all listed versions attached to this add-on. Requires either reviewer permissions or a user account listed as a developer of the add-on.
all_with_unlisted	Show all versions (including unlisted) attached to this add-on. Requires either reviewer permissions or a user account listed as a developer of the add-on.
all_with_deleted	Show all versions attached to this add-on, including deleted ones. Requires admin permissions.

Version Detail

This endpoint allows you to fetch a single version belonging to a specific add-on.

GET /api/v4/addons/addon/(int:addon_id|string:addon_slug|string:addon_guid)/versions/(int:id)/

Query Parameters

- **lang** (*string*) – Activate translations in the specific language for that query. (See *translated fields*)

Response JSON Object

- **id** (*int*) – The version id.
- **channel** (*string*) – The version channel, which determines its visibility on the site. Can be either `unlisted` or `listed`.
- **compatibility** (*object*) – Object detailing which *applications* the version is compatible with. The exact min/max version numbers in the object correspond to valid application versions. Example:

```
{
  "compatibility": {
    "android": {
      "min": "38.0a1",
      "max": "43.0"
    },
    "firefox": {
      "min": "38.0a1",
      "max": "43.0"
    }
  }
}
```

- **compatibility[app_name].max** (*object*) – Maximum version of the corresponding app the version is compatible with. Should only be enforced by clients if `is_strict_compatibility_enabled` is `true`.
- **compatibility[app_name].min** (*object*) – Minimum version of the corresponding app the version is compatible with.
- **edit_url** (*string*) – The URL to the developer edit page for the version.
- **files** (*array*) – Array holding information about the files for the version.
- **files[].id** (*int*) – The id for a file.
- **files[].created** (*string*) – The creation date for a file.

- **files[].hash** (*string*) – The hash for a file.
- **files[].is_mozilla_signed_extension** (*boolean*) – Whether the file was signed with a Mozilla internal certificate or not.
- **files[].is_restart_required** (*boolean*) – Whether the file requires a browser restart to work once installed or not.
- **files[].is_webextension** (*boolean*) – Whether the file is a WebExtension or not.
- **files[].optional_permissions[]** (*array*) – Array of the optional webextension permissions for this File, as strings. Empty for non-webextensions.
- **files[].permissions[]** (*array*) – Array of the webextension permissions for this File, as strings. Empty for non-webextensions.
- **files[].platform** (*string*) – The *platform* for a file.
- **files[].size** (*int*) – The size for a file, in bytes.
- **files[].status** (*int*) – The *status* for a file.
- **files[].url** (*string*) – The (absolute) URL to download a file. Clients using this API can append an optional `src` query parameter to the url which would indicate the source of the request (See *download sources*).
- **license** (*object*) – Object holding information about the license for the version.
- **license.is_custom** (*boolean*) – Whether the license text has been provided by the developer, or not. (When `false` the license is one of the common, predefined, licenses).
- **license.name** (*string|object|null*) – The name of the license (See *translated fields*).
- **license.text** (*string|object|null*) – The text of the license (See *translated fields*). For performance reasons this field is omitted from add-on detail endpoint.
- **license.url** (*string|null*) – The URL of the full text of license.
- **release_notes** (*string|object|null*) – The release notes for this version (See *translated fields*).
- **reviewed** (*string*) – The date the version was reviewed at.
- **is_strict_compatibility_enabled** (*boolean*) – Whether or not this version has strictCompatibility. set.
- **version** (*string*) – The version number string for the version.

Add-on EULA and Privacy Policy

This endpoint allows you to fetch an add-on EULA and privacy policy.

GET /api/v4/addons/addon/(int:id|string:slug|string:guid)/eula_policy/

Note: Non-public add-ons and add-ons with only unlisted versions require both:

- authentication
- reviewer permissions or an account listed as a developer of the add-on

---

Response JSON Object

- **eula** (*string|object|null*) – The text of the EULA, if present (See *translated fields*).
- **privacy_policy** (*string|object|null*) – The text of the Privacy Policy, if present (See *translated fields*).

Language Tools

This endpoint allows you to list all public language tools add-ons available on AMO.

GET /api/v4/addons/language-tools/

Note: Because this endpoint is intended to be used to feed a page that displays all available language tools in a single page, it is not paginated as normal, and instead will return all results without obeying regular pagination parameters. The ordering is left undefined, it's up to the clients to re-order results as needed before displaying the add-ons to the end-users.

In addition, the results can be cached for up to 24 hours, based on the full URL used in the request.

Query Parameters

- **app** (*string*) – Mandatory. Filter by *add-on application* availability.
- **appversion** (*string*) – Filter by application version compatibility. Pass the full version as a string, e.g. `46.0`. Only valid when both the `app` and `type` parameters are also present, and only makes sense for Language Packs, since Dictionaries are always compatible with every application version.
- **author** (*string*) – Filter by exact (listed) author username. Multiple author names can be specified, separated by comma(s), in which case add-ons with at least one matching author are returned.
- **lang** (*string*) – Activate translations in the specific language for that query. (See *translated fields*)
- **type** (*string*) – Mandatory when `appversion` is present. Filter by *add-on type*. The default is to return both Language Packs or Dictionaries.

Response JSON Object

- **results** (*array*) – An array of language tools.
- **results[].id** (*int*) – The add-on id on AMO.
- **results[].current_compatible_version** (*object*) – Object holding the latest publicly available *version* of the add-on compatible with the `appversion` parameter used. Only present when `appversion` is passed and valid. For performance reasons, only the following version properties are returned on the object: `id`, `files`, `reviewed`, and `version`.
- **results[].default_locale** (*string*) – The add-on default locale for translations.
- **results[].name** (*string|object|null*) – The add-on name (See *translated fields*).
- **results[].guid** (*string*) – The add-on extension identifier.

- **results[].slug** (*string*) – The add-on slug.
- **results[].target_locale** (*string*) – For dictionaries and language packs, the locale the add-on is meant for. Only present when using the Language Tools endpoint.
- **results[].type** (*string*) – The *add-on type*.
- **results[].url** (*string*) – The (absolute) add-on detail URL.

Replacement Add-ons

This endpoint returns a list of suggested replacements for legacy add-ons that are unsupported in Firefox 57. Added to support the TAAR recommendation service.

GET /api/v4/addons/replacement-addon/

Query Parameters

- **page** (*int*) – 1-based page number. Defaults to 1.
- **page_size** (*int*) – Maximum number of results to return for the requested page. Defaults to 25.

Response JSON Object

- **count** (*int*) – The total number of replacements.
- **next** (*string*) – The URL of the next page of results.
- **previous** (*string*) – The URL of the previous page of results.
- **results** (*array*) – An array of replacements matches.
- **results[].guid** (*string*) – The extension identifier of the legacy add-on.
- **results[].replacement[]** (*string*) – An array of guids for the replacements add-ons. If there is a direct replacement this will be a list of one add-on guid. The list can be empty if all the replacement add-ons are invalid (e.g. not publicly available on AMO). The list will also be empty if the replacement is to a url that is not an addon or collection.

Recommendations

This endpoint provides recommendations of other addons to install, fetched from the recommendation service. Four recommendations are fetched, but only valid, publicly available addons are shown (so max 4 will be returned, and possibly less).

GET /api/v4/addons/recommendations/

Query Parameters

- **guid** (*string*) – Fetch recommendations for this add-on guid.
- **lang** (*string*) – Activate translations in the specific language for that query. (See *translated fields*)
- **recommended** (*boolean*) – Fetch recommendations from the recommendation service, or return a curated fallback list instead.

Response JSON Object

- **outcome** (*string*) – Outcome of the response returned. Will be either: `recommended` - responses from recommendation service; `recommended_fallback` - service timed

out or returned empty or invalid results so we returned fallback; `curated` - `recommended=False` was requested so fallback returned.

- **`fallback_reason`** (*`string|null`*) – if `outcome` was `recommended_fallback` then the reason why. Will be either: `timeout`, `no_results`, or `invalid_results`.
- **`count`** (*`int`*) – The number of results for this query.
- **`next`** (*`string`*) – The URL of the next page of results.
- **`previous`** (*`string`*) – The URL of the previous page of results.
- **`results`** (*`array`*) – An array of *add-ons*. The following fields are omitted for performance reasons: `release_notes` and `license` fields on `current_version` and `current_beta_version`, as well as `picture_url` from `authors`.

1.2.8 Blocklist

Note: These APIs are not frozen and can change at any time without warning. See *the API versions available* for alternatives if you need stability.

Blocks

This endpoint returns an add-on Block from the blocklist, specified by guid or id.

`GET /api/v4/blocklist/block/(int:block_id|string:guid)`

Query Parameters

- **`lang`** (*`string`*) – Activate translations in the specific language for that query. (See *Translated Fields*)
- **`wrap_outgoing_links`** (*`string`*) – If this parameter is present, wrap outgoing links through `outgoing.prod.mozaws.net` (See *Outgoing Links*)

Response JSON Object

- **`id`** (*`int`*) – The id for the block.
- **`created`** (*`string`*) – The date the block was created.
- **`modified`** (*`string`*) – The date the block was last updated.
- **`addon_name`** (*`string|object|null`*) – The add-on name, if we have details of an add-on matching that guid (See *translated fields*).
- **`guid`** (*`string`*) – The guid of the add-on being blocked.
- **`min_version`** (*`string`*) – The minimum version of the add-on that will be blocked. “0” is the lowest version, meaning all versions up to max_version will be blocked. (“0” - “*” would be all versions).
- **`max_version`** (*`string`*) – The maximum version of the add-on that will be blocked. “*” is the highest version, meaning all versions from min_version will be blocked. (“0” - “*” would be all versions).
- **`reason`** (*`string|null`*) – Why the add-on needed to be blocked.
- **`url`** (*`string|null`*) – A url to the report/request that detailed why the add-on should potentially be blocked. Typically a bug report on bugzilla.mozilla.org.

1.2.9 Categories

Note: These APIs are not frozen and can change at any time without warning. See *the API versions available* for alternatives if you need stability.

Category List

Categories are defined by a name, a slug, a type and an application. Slugs are only guaranteed to be unique for a given `app` and `type` combination, and can therefore be re-used for different categories.

This endpoint is not paginated.

`GET /api/v4/addons/categories/`

Response JSON Object

- **`id`** (*`int`*) – The category id.
- **`name`** (*`string`*) – The category name. Returns the already translated string.
- **`slug`** (*`string`*) – The category slug. See *csv table* for more possible values.
- **`application`** (*`string`*) – Application, see *add-on application* for more details.
- **`misc`** (*`boolean`*) – Whether or not the category is miscellaneous.
- **`type`** (*`string`*) – Category type, see *add-on type* for more details.
- **`weight`** (*`int`*) – Category weight used in sort ordering.
- **`description`** (*`string|null`*) – The category description. Returns the already translated string.

Current categories

Name	Slug	Type	Application
Alerts & Updates	alerts-updates	extension	firefox
Appearance	appearance	extension	firefox
Bookmarks	bookmarks	extension	firefox
Download Management	download-management	extension	firefox
Feeds, News & Blogging	feeds-news-blogging	extension	firefox
Games & Entertainment	games-entertainment	extension	firefox
Language Support	language-support	extension	firefox
Photos, Music & Videos	photos-music-videos	extension	firefox
Privacy & Security	privacy-security	extension	firefox
Search Tools	search-tools	extension	firefox
Shopping	shopping	extension	firefox
Social & Communication	social-communication	extension	firefox
Tabs	tabs	extension	firefox
Web Development	web-development	extension	firefox
Other	other	extension	firefox
Animals	animals	theme	firefox
Compact	compact	theme	firefox
Large	large	theme	firefox

continues on next page

Table 1 – continued from previous page

Name	Slug	Type	Application
Miscellaneous	miscellaneous	theme	firefox
Modern	modern	theme	firefox
Nature	nature	theme	firefox
OS Integration	os-integration	theme	firefox
Retro	retro	theme	firefox
Sports	sports	theme	firefox
General	general	dictionary	firefox
Bookmarks	bookmarks	search	firefox
Business	business	search	firefox
Dictionaries & Encyclopedias	dictionaries-encyclopedias	search	firefox
General	general	search	firefox
Kids	kids	search	firefox
Multiple Search	multiple-search	search	firefox
Music	music	search	firefox
News & Blogs	news-blogs	search	firefox
Photos & Images	photos-images	search	firefox
Shopping & E-Commerce	shopping-e-commerce	search	firefox
Social & People	social-people	search	firefox
Sports	sports	search	firefox
Travel	travel	search	firefox
Video	video	search	firefox
General	general	language	firefox
Abstract	abstract	persona	firefox
Causes	causes	persona	firefox
Fashion	fashion	persona	firefox
Film and TV	film-and-tv	persona	firefox
Firefox	firefox	persona	firefox
Foxkeh	foxkeh	persona	firefox
Holiday	holiday	persona	firefox
Music	music	persona	firefox
Nature	nature	persona	firefox
Other	other	persona	firefox
Scenery	scenery	persona	firefox
Seasonal	seasonal	persona	firefox
Solid	solid	persona	firefox
Sports	sports	persona	firefox
Websites	websites	persona	firefox
Appearance and Customization	appearance	extension	thunderbird
Calendar and Date/Time	calendar	extension	thunderbird
Chat and IM	chat	extension	thunderbird
Contacts	contacts	extension	thunderbird
Folders and Filters	folders-and-filters	extension	thunderbird
Import/Export	importexport	extension	thunderbird
Language Support	language-support	extension	thunderbird
Message Composition	composition	extension	thunderbird
Message and News Reading	message-and-news-reading	extension	thunderbird
Miscellaneous	miscellaneous	extension	thunderbird
Privacy and Security	privacy-and-security	extension	thunderbird
Tags	tags	extension	thunderbird

continues on next page

Table 1 – continued from previous page

Name	Slug	Type	Application
Compact	compact	theme	thunderbird
Miscellaneous	miscellaneous	theme	thunderbird
Modern	modern	theme	thunderbird
Nature	nature	theme	thunderbird
General	general	dictionary	thunderbird
General	general	language	thunderbird
Bookmarks	bookmarks	extension	seamonkey
Downloading and File Management	downloading-and-file-management	extension	seamonkey
Interface Customizations	interface-customizations	extension	seamonkey
Language Support and Translation	language-support-and-translation	extension	seamonkey
Miscellaneous	miscellaneous	extension	seamonkey
Photos and Media	photos-and-media	extension	seamonkey
Privacy and Security	privacy-and-security	extension	seamonkey
RSS, News and Blogging	rss-news-and-blogging	extension	seamonkey
Search Tools	search-tools	extension	seamonkey
Site-specific	site-specific	extension	seamonkey
Web and Developer Tools	web-and-developer-tools	extension	seamonkey
Miscellaneous	miscellaneous	theme	seamonkey
General	general	dictionary	seamonkey
General	general	language	seamonkey
Device Features & Location	device-features-location	extension	android
Experimental	experimental	extension	android
Feeds, News, & Blogging	feeds-news-blogging	extension	android
Performance	performance	extension	android
Photos & Media	photos-media	extension	android
Security & Privacy	security-privacy	extension	android
Shopping	shopping	extension	android
Social Networking	social-networking	extension	android
Sports & Games	sports-games	extension	android
User Interface	user-interface	extension	android

1.2.10 Collections

Note: These APIs are not frozen and can change at any time without warning. See *the API versions available* for alternatives if you need stability. The only authentication method available at the moment is *the internal one*.

The following API endpoints cover user created collections.

List

Note: This API requires *authentication*.

This endpoint allows you to list all collections authored by the specified user. The results are sorted by the most recently updated collection first.

GET /api/v4/accounts/account/(int:user_id|string:username)/collections/

Response JSON Object

- **count** (*int*) – The number of results for this query.
- **next** (*string*) – The URL of the next page of results.
- **previous** (*string*) – The URL of the previous page of results.
- **results** (*array*) – An array of *collections*.

Detail

This endpoint allows you to fetch a single collection by its `slug`. It returns any `public` collection by the specified user. You can access a non-`public` collection only if it was authored by you, the authenticated user. If you have `Admin:Curation` permission you can see any collection belonging to the `mozilla` user.

GET /api/v4/accounts/account/(int:user_id|string:username)/collections/(string: *collection_slug*) **/**

Response JSON Object

- **id** (*int*) – The id for the collection.
- **addon_count** (*int*) – The number of add-ons in this collection.
- **author.id** (*int*) – The id of the author (creator) of the collection.
- **author.name** (*string*) – The name of the author.
- **author.url** (*string*) – The link to the profile page for of the author.
- **author.username** (*string*) – The username of the author.
- **default_locale** (*string*) – The default locale of the description and name fields. (See *translated fields*).
- **description** (*string|object|null*) – The description the author added to the collection. (See *translated fields*).
- **modified** (*string*) – The date the collection was last updated.
- **name** (*string|object*) – The name of the collection. (See *translated fields*).
- **public** (*boolean*) – Whether the collection is *listed* - publicly viewable.
- **slug** (*string*) – The name used in the URL.
- **url** (*string*) – The (absolute) collection detail URL.
- **uuid** (*string*) – A unique identifier for this collection; primarily used to count addon installations that come via this collection.

If the `with_addons` parameter is passed then *addons in the collection* are returned along with the detail. Add-ons returned are limited to the first 25 in the collection, in the default sort (popularity, descending). Filtering is as per *collection addon list endpoint* - i.e. defaults to only including public add-ons. Additional add-ons can be returned from the *Collection Add-on list endpoint*.

GET /api/v4/accounts/account/(int:user_id|string:username)/collections/(string:*collection_slug*)/?with_addons

Query Parameters

- **filter** (*string*) – The *filter* to apply.

Response JSON Object

- **id** (*int*) – The id for the collection.
- **addon_count** (*int*) – The number of add-ons in this collection.
- **addons** (*array*) – An array of *addons with notes*.

... rest as *collection detail response*

Create

Note: This API requires *authentication*.

This endpoint allows a collection to be created under your account. Any fields in the *collection* but not listed below are not settable and will be ignored in the request.

POST /api/v4/accounts/account/(int:user_id|string:username)/collections/

Request JSON Object

- **default_locale** (*string|null*) – The default locale of the description and name fields. Defaults to *en-US*. (See *translated fields*).
- **description** (*string|object|null*) – The description the author added to the collection. (See *translated fields*).
- **name** (*string|object*) – The name of the collection. (required) (See *translated fields*).
- **public** (*boolean*) – Whether the collection is *listed* - publicly viewable. Defaults to *True*.
- **slug** (*string*) – The name used in the URL (required).

Edit

Note: This API requires *authentication*. If you have `Admin:Curation` permission you can edit any collection belonging to the `mozilla` user.

This endpoint allows some of the details for a collection to be updated. Any fields in the *collection* but not listed below are not editable and will be ignored in the patch request.

PATCH /api/v4/accounts/account/(int:user_id|string:username)/collections/(string: *collection_slug*) **/**

Request JSON Object

- **default_locale** (*string*) – The default locale of the description and name fields. (See *translated fields*).
- **description** (*string|object|null*) – The description the author added to the collection. (See *translated fields*).
- **name** (*string|object*) – The name of the collection. (See *translated fields*).
- **public** (*boolean*) – Whether the collection is *listed* - publicly viewable.
- **slug** (*string*) – The name used in the URL.

Delete

Note: This API requires *authentication*.

This endpoint allows the collection to be deleted.

DELETE /api/v4/accounts/account/(int:user_id|string:username)/collections/(string: *collection_slug*) **/**

Collection Add-ons List

This endpoint lists the add-ons in a collection, together with collector's notes.

GET /api/v4/accounts/account/(int:user_id|string:username)/collections/(string: *collection_slug*) **/ addons/**

Query Parameters

- **filter** (*string*) – The *filter* to apply.

- **sort** (*string*) – The sort parameter. The available parameters are documented in the *table below*.

Response JSON Object

- **count** (*int*) – The number of results for this query.
- **next** (*string*) – The URL of the next page of results.
- **previous** (*string*) – The URL of the previous page of results.
- **results** (*array*) – An array of *items* in this collection.

Available sorting parameters:

Parameter	Description
added	Date the add-on was added to the collection, ascending.
popularity	Number of total weekly downloads of the add-on, ascending.
name	Add-on name, ascending.

All sort parameters can be reversed, e.g. '-added' for descending dates. The default sorting is by popularity, descending ('-popularity'). There can only be one sort parameter, multiple orderings are not supported.

By default, the collection addon list API will only return public add-ons (excluding add-ons that have no approved listed versions, are disabled or deleted) - you can change that with the `filter` query parameter:

Value	Description
all	Show all add-ons in the collection, including those that have non-public statuses. This still excludes deleted add-ons.
all_with_deleted	Show all add-ons in the collection, including deleted add-ons too.

Collection Add-ons Detail

This endpoint gets details of a single add-on in a collection, together with collector's notes.

GET /api/v4/accounts/account/(int:user_id|string:username)/collections/(string:*collection_slug***)/addons/(int:addon_id|stri**

Response JSON Object

- **addon** (*object*) – The *add-on* for this item.
- **notes** (*string|object|null*) – The collectors notes for this item. (See *translated fields*).

Collection Add-ons Create

Note: This API requires *authentication*.

This endpoint allows a single add-on to be added to a collection, optionally with collector's notes.

POST /api/v4/accounts/account/(int:user_id|string:username)/collections/(string: *collection_slug*) **/ addons/**

Request JSON Object

- **addon** (*string*) – The add-on id or slug to be added (required).
- **notes** (*string|object|null*) – The collectors notes for this item. (See *translated fields*).

Collection Add-ons Edit

Note: This API requires *authentication*. If you have `Admin:Curation` permission you can edit the add-ons of any collection belonging to the `mozilla` user. If you have `Collections:Contribute` permission you can edit the add-ons of mozilla's `Featured Themes` collection.

This endpoint allows the collector's notes for single add-on to be updated.

PATCH /api/v4/accounts/account/(int:user_id|string:username)/collections/(string: *collection_slug*) **/ addons/ (int:addon_id|st**

Request JSON Object

- **notes** (*string|object|null*) – The collectors notes for this item. (See *translated fields*).

Collection Add-ons Delete

Note: This API requires *authentication*. If you have `Admin:Curation` permission you can remove add-ons from any collection belonging to the `mozilla` user. If you have `Collections:Contribute` permission you can remove add-ons from mozilla's `Featured Themes` collection.

This endpoint allows a single add-on to be removed from a collection.

DELETE /api/v4/accounts/account/(int:user_id|string:username)/collections/(string: *collection_slug*) **/addons/(int:addon_id|st**

1.2.11 Discovery

Note: These APIs are not frozen and can change at any time without warning. See *the API versions available* for alternatives if you need stability.

Discovery Content

This endpoint allows you to fetch content for the new Discovery Pane in Firefox (about:addons).

Note: If a telemetry client id is passed as a parameter to the discovery pane api endpoint then static curated content is amended with recommendations from the recommendation service. The same number of results will be returned as a standard discovery response and only extensions (not themes) are recommeded. Only valid, publicly available addons are shown.

E.g. a standard discovery pane will display 7 items, 4 extensions and 3 themes. Up to 4 extensions will be replaced with recommendations; the 3 themes will not be replaced. The API will still return a total of 7 items.

GET /api/v4/discovery/

Query Parameters

- **lang** (*string*) – Activate translations in the specific language for that query. (See *translated fields*)
- **edition** (*string*) – Optionally return content for a specific edition of Firefox. Currently only `china` (and the alias `MozillaOnline`) is supported.
- **telemetry-client-id** (*string*) – Optional sha256 hash of the telemetry client ID to be passed to the TAAR service to enable recommendations. Must be the hex value of a sha256 hash, otherwise it will be ignored.

Response JSON Object

- **count** (*int*) – The number of results for this query.
- **results** (*array*) – The array containing the results for this query.
- **results[].heading** (*string*) – The heading for this item. May contain some HTML tags.
- **results[].description** (*string|null*) – The description for this item, if any. May contain some HTML tags.
- **results[].description_text** (*string|null*) – The description for this item, if any. Text-only, content might slightly differ from `description` because of that.
- **results[].is_recommendation** (*boolean*) – If this item was from the recommendation service, rather than static curated content.

- **results[].addon** (*object*) – The *add-on* for this item. Only a subset of fields are present: `id`, `authors`, `average_daily_users`, `current_version` (with only the `id`, `compatibility`, `is_strict_compatibility_enabled` and `files` fields present), `guid`, `icon_url`, `name`, `ratings`, `previews`, `slug`, `theme_data`, `type` and `url`.

Editorial Content

This endpoint allows you to fetch all editorial content for Discovery Pane Recommendations. This is used internally to generate .po files containing the strings defined by the content team. It is also used by TAAR service to obtain a list of appropriate add-ons to recommended.

GET /api/v4/discovery/editorial/

Query Parameters

- **recommended** (*boolean*) – Filter to only add-ons recommended by Mozilla. Only `recommended=true` is supported.

Response JSON Object

- **results** (*array*) – The array containing the results for this query. There is no pagination, all results are returned.
- **results[].addon** (*object*) – A *add-on* object for this item, but only containing one field: `guid`.
- **results[].custom_heading** (*string|null*) – The custom heading for this item, if any.
- **results[].custom_description** (*string|null*) – The custom description for this item, if any.

1.2.12 Download Sources

When requesting an add-on file URL, clients have the option to indicate what is the source of the request. This will then be used to group by downloads by sources in the add-on statistics page.

To indicate the source, add the `src` query parameter to the download URL. The following values are recognized:

Name	Description
api	Add-ons Manager
discovery-promo	Add-ons Manager Promo
discovery-featured	Add-ons Manager Featured
discovery-learnmore	Add-ons Manager Learn More
ss	Search Suggestions
search	Search Results
homepagepromo	Homepage Promo
hp-btn-promo	Homepage Promo
hp-dl-promo	Homepage Promo
hp-hc-featured	Homepage Featured
hp-dl-featured	Homepage Featured
hp-hc-upandcoming	Homepage Up and Coming
hp-dl-upandcoming	Homepage Up and Coming
hp-dl-mostpopular	Homepage Most Popular

continues on next page

Table 2 – continued from previous page

Name	Description
dp-btn-primary	Detail Page
dp-btn-version	Detail Page (bottom)
addondetail	Detail Page
addon-detail-version	Detail Page (bottom)
dp-btn-devchannel	Detail Page (Development Channel)
oftenusedwith	Often Used With
dp-hc-oftenusedwith	Often Used With
dp-dl-oftenusedwith	Often Used With
dp-hc-othersby	Others By Author
dp-dl-othersby	Others By Author
dp-hc-dependencies	Dependencies
dp-dl-dependencies	Dependencies
dp-hc-upsell	Upsell
dp-dl-upsell	Upsell
developers	Meet the Developer
userprofile	User Profile
version-history	Version History
sharingapi	Sharing
category	Category Pages
collection	Collections
cb-hc-featured	Category Landing Featured Carousel
cb-dl-featured	Category Landing Featured Carousel
cb-hc-toprated	Category Landing Top Rated
cb-dl-toprated	Category Landing Top Rated
cb-hc-mostpopular	Category Landing Most Popular
cb-dl-mostpopular	Category Landing Most Popular
cb-hc-recentlyadded	Category Landing Recently Added
cb-dl-recentlyadded	Category Landing Recently Added
cb-btn-featured	Browse Listing Featured Sort
cb-dl-featured	Browse Listing Featured Sort
cb-btn-users	Browse Listing Users Sort
cb-dl-users	Browse Listing Users Sort
cb-btn-rating	Browse Listing Rating Sort
cb-dl-rating	Browse Listing Rating Sort
cb-btn-created	Browse Listing Created Sort
cb-dl-created	Browse Listing Created Sort
cb-btn-name	Browse Listing Name Sort
cb-dl-name	Browse Listing Name Sort
cb-btn-popular	Browse Listing Popular Sort
cb-dl-popular	Browse Listing Popular Sort
cb-btn-updated	Browse Listing Updated Sort
cb-dl-updated	Browse Listing Updated Sort
cb-btn-hotness	Browse Listing Up and Coming Sort
cb-dl-hotness	Browse Listing Up and Coming Sort
find-replacement	Find replacement service for obsolete add-ons

1.2.13 Hero Shelves

Note: These APIs are not frozen and can change at any time without warning. See *the API versions available* for alternatives if you need stability.

Combined Hero Shelves

This convienence endpoint serves a single, randomly selected, primary hero shelf, and a single, randomly selected secondary hero shelf.

GET /api/v4/hero/

Query Parameters

- **lang** (*string*) – Activate translations in the specific language for that query. (See *translated fields*)
- **wrap_outgoing_links** (*string*) – If this parameter is present, wrap outgoing links through `outgoing.prod.mozaws.net` (See *Outgoing Links*)

Response JSON Object

- **primary** (*object*) – A *primary hero shelf*.
- **secondary** (*object*) – A *secondary hero shelf*.

Primary Hero Shelves

This endpoint returns all enabled primary hero shelves. As there will only ever be a small number of shelves this endpoint is not paginated.

GET /api/v4/hero/primary/

Query Parameters

- **lang** (*string*) – Activate translations in the specific language for that query. (See *translated fields*)
- **all** (*boolean*) – return all shelves - both enabled and disabled. To be used internally to generate .po files containing the strings defined by the content team.
- **raw** (*string*) – If this parameter is present, don't localise description or fall-back to addon metadata. To be used internally to generate .po files containing the strings defined by the content team.
- **wrap_outgoing_links** (*string*) – If this parameter is present, wrap outgoing links through `outgoing.prod.mozaws.net` (See *Outgoing Links*)

Response JSON Object

- **results** (*array*) – The array containing the results for this query.
- **results[].gradient** (*object*) – The background colors used for the gradient.
- **results[].gradient.start** (*string*) – The starting color name for gradient - typically top or left. The name is from the photon color variables.
- **results[].gradient.end** (*string*) – The ending color name for gradient - typically bottom or right. The name is from the photon color variables.

- **results[].featured_image** (*string|null*) – The image used to illustrate the item, if set.
- **results[].description** (*string|null*) – The description for this item, if any.
- **results[].addon** (*object*) – The *add-on* for this item if the addon is hosted on AMO. Either this field or `external` will be present. Only a subset of fields are present: `id`, `authors`, `average_daily_users`, `current_version` (with only the `id`, `compatibility`, `is_strict_compatibility_enabled` and `files` fields present), `guid`, `icon_url`, `name`, `ratings`, `previews`, `slug`, `theme_data`, `type` and `url`.
- **results[].external** (*object*) – The *add-on* for this item if the addon is externally hosted. Either this field or `addon` will be present. Only a subset of fields are present: `id`, `guid`, `homepage`, `name` and `type`.

Secondary Hero Shelves

This endpoint returns all enabled secondary hero shelves. As there will only ever be a small number of shelves - and likely only one - this endpoint is not paginated.

GET /api/v4/hero/secondary/

Query Parameters

- **lang** (*string*) – Activate translations in the specific language for that query. (See *translated fields*)
- **all** (*boolean*) – return all shelves - both enabled and disabled. To be used internally to generate .po files containing the strings defined by the content team.
- **wrap_outgoing_links** (*string*) – If this parameter is present, wrap outgoing links through `outgoing.prod.mozaws.net` (See *Outgoing Links*)

Response JSON Object

- **results** (*array*) – The array containing the results for this query.
- **results[].headline** (*string*) – The headline for this item.
- **results[].description** (*string*) – The description for this item.
- **results[].cta** (*object|null*) – The optional call to action link and text to be displayed with the item.
- **results[].cta.url** (*string*) – The url the call to action would link to.
- **results[].cta.text** (*string*) – The call to action text.
- **results[].modules** (*array*) – The modules for this shelf. Should always be 3.
- **results[].modules[].icon** (*string*) – The icon used to illustrate the item.
- **results[].modules[].description** (*string*) – The description for this item.
- **results[].modules[].cta** (*object|null*) – The optional call to action link and text to be displayed with the item.
- **results[].modules[].cta.url** (*string*) – The url the call to action would link to.
- **results[].modules[].cta.text** (*string*) – The call to action text.

1.2.14 Ratings

Note: These APIs are not frozen and can change at any time without warning. See *the API versions available* for alternatives if you need stability. The only authentication method available at the moment is *the internal one*.

List ratings

This endpoint allows you to fetch ratings for a given add-on or user. Either `addon` or `user` query parameters are required, and they can be combined together.

When `addon`, `user` and `version` are passed on the same request, `page_size` will automatically be set to `1`, since an user can only post one rating per version of a given add-on. This can be useful to find out if a user has already posted a rating for the current version of an add-on.

`GET /api/v4/ratings/rating/`

Query Parameters

- **`addon`** (*`string`*) – The *add-on* id, slug, or guid to fetch ratings from. When passed, the ratings shown will always be the latest posted by each user on this particular add-on (which means there should only be one rating per user in the results), unless the `version` parameter is also passed.
- **`exclude_ratings`** (*`string`*) – Exclude ratings by their `id`. Multiple ratings can be specified, separated by comma(s).
- **`filter`** (*`string`*) – The *filter(s)* to apply.
- **`score`** (*`string`*) – Only include ratings that have been given a specific `score`. Multiple scores can be specified, separated by comma(s).
- **`show_flags_for`** (*`string`*) – The user id to show flags for. If the request is made with an authenticated user matching this parameter value, a `flags` property will be added to the response as described below in *ratings*.
- **`show_grouped_ratings`** (*`boolean`*) – Whether or not to show ratings aggregates for this add-on in the response (Use "true"/"1" as truthy values, "0"/"false" as falsy ones).
- **`show_permissions_for`** (*`string`*) – The user id to show permissions for. If the request is made with an authenticated user matching this parameter value, and the `addon` parameter is also present, a `can_reply` property will be added to the response as described below.
- **`user`** (*`int`*) – The user id to fetch ratings from.
- **`version`** (*`int`*) – The version id to fetch ratings from.
- **`page`** (*`int`*) – 1-based page number. Defaults to 1.
- **`page_size`** (*`int`*) – Maximum number of results to return for the requested page. Defaults to 25.

Response JSON Object

- **`count`** (*`int`*) – The number of results for this query.
- **`next`** (*`string`*) – The URL of the next page of results.
- **`previous`** (*`string`*) – The URL of the previous page of results.
- **`results`** (*`array`*) – An array of *ratings*.

- **can_reply** (*boolean*) – Only present if the `addon` query parameter and `show_permissions_for` parameters are present. A boolean indicating if the user that made the ratings list request can reply to ratings in that list.
- **grouped_ratings** (*object*) – Only present if `show_grouped_ratings` query parameter is present. An object with 5 key-value pairs, the keys representing each possible rating (Though a number, it has to be converted to a string because of the JSON formatting) and the values being the number of times the corresponding rating has been posted for this add-on, e.g. `{"1": 4, "2": 8, "3": 15, "4": 16: "5": 23}`.

By default, the rating list API will only return not-deleted ratings, and include ratings without text. You can change that with the `filter` query parameter. You can filter by multiple values, e.g. `filter=with_deleted,without_empty_body,with_yours`

Value	Description
with_deleted	Returns deleted ratings too. This requires the Addons:Edit permission.
without_empty_body	Excludes ratings that only contain a rating, and no textual content.
with_yours	Used in combination *without_empty_body* to include your own ratings, even if they have no text.

Detail

This endpoint allows you to fetch a rating by its id.

Note: Users with `Addons:Edit` permission will be able to see deleted ratings. Use the `is_deleted` property to distinguish those from normal ratings everyone is able to see.

GET /api/v4/ratings/rating/(int: *id*)/

Query Parameters

- **show_flags_for** (*string*) – The user id to show flags for. If the request is made with an authenticated user matching this parameter value, a `flags` property will be added to the response as described below.

Response JSON Object

- **id** (*int*) – The rating id.
- **addon** (*object*) – A simplified *add-on* object that contains only a few properties: `id`, `name`, `icon_url` and `slug`.
- **body** (*string|null*) – The text of the rating.
- **is_deleted** (*boolean*) – Boolean indicating whether the rating has been deleted or not.
- **is_latest** (*boolean*) – Boolean indicating whether the rating is the latest posted by the user on the same add-on.
- **previous_count** (*int*) – The number of ratings posted by the user on the same add-on before this one.
- **flags[]** (*object*) – A list of flags the user requesting has previously applied to this rating (that haven't been processed by moderators already). Only present if `show_flags_for` parameter sent.

- **flags.flag** (*string*) – A *constant* describing the reason behind the flagging.
- **flags.note** (*string|null*) – A note to explain further the reason behind the flagging if `flag` was `rating_flag_reason_other`; null otherwise.
- **score** (*int*) – The score the user gave as part of the rating.
- **reply** (*object|null*) – The rating object containing the developer reply to this rating, if any (The fields `rating`, `reply` and `version` are omitted).
- **version.id** (*int*) – The add-on version id the rating applies to.
- **version.version** (*string*) – The add-on version string the rating applies to.
- **user** (*object*) – Object holding information about the user who posted the rating.
- **user.id** (*string*) – The user id.
- **user.name** (*string*) – The user name.
- **user.url** (*string*) – The user profile URL.
- **user.username** (*string*) – The user username.

Post

This endpoint allows you to post a new rating for a given add-on and version. If successful a *rating object* is returned.

Note: Requires authentication.

POST /api/v4/ratings/rating/

Request JSON Object

- **addon** (*string*) – The add-on id the rating applies to (required).
- **body** (*string|null*) – The text of the rating.
- **score** (*int*) – The score the user wants to give as part of the rating (required).
- **version** (*int*) – The add-on version id the rating applies to (required).

Edit

This endpoint allows you to edit an existing rating by its id. If successful a *rating object* is returned.

Note: Requires authentication and Addons:Edit permissions or the user account that posted the rating.

Only body and score are allowed for modification.

PATCH /api/v4/ratings/rating/(int: *id*)/

Request JSON Object

- **body** (*string|null*) – The text of the rating.
- **score** (*int*) – The score the user wants to give as part of the rating.

Delete

This endpoint allows you to delete an existing rating by its id.

> **Note:** Requires authentication and Addons:Edit permission or the user account that posted the rating. Even with the right permission, users can not delete a rating from somebody else if it was posted on an add-on they are listed as a developer of.

DELETE /api/v4/ratings/rating/(**int:** *id*)**/**

Reply

This endpoint allows you to reply to an existing user rating. If successful a *rating reply object* is returned - a *rating* object but with the fields `rating`, `reply` and `version` omitted.

> **Note:** Requires authentication and either Addons:Edit permission or a user account listed as a developer of the add-on.

POST /api/v4/ratings/rating/(**int:** *id*)**/reply/**

Request JSON Object

- **body** (*string*) – The text of the reply (required).

Flag

This endpoint allows you to flag an existing user rating, to let a moderator know that something may be wrong with it.

> **Note:** Requires authentication and a user account different from the one that posted the rating.

POST /api/v4/ratings/rating/(**int:** *id*)**/flag/**

Request JSON Object

- **flag** (*string*) – A *constant* describing the reason behind the flagging.
- **note** (*string|null*) – A note to explain further the reason behind the flagging. This field is required if the flag is `rating_flag_reason_other`, and passing it will automatically change the flag to that value.

Response JSON Object

- **object** – If successful, an object with a `msg` property containing a success message. If not, an object indicating which fields contain errors.

Available constants for the `flag` property:

Constant	Description
rating_flag_reason_spam	Spam or otherwise non-rating content
rating_flag_reason_language	Inappropriate language/dialog
rating_flag_reason_bug_support	Misplaced bug report or support request
rating_flag_reason_other	Other (please specify)

1.2.15 Reviewers

Note: These APIs are not frozen and can change at any time without warning. See *the API versions available* for alternatives if you need stability. The only authentication method available at the moment is *the internal one*.

Subscribe

This endpoint allows you to subscribe the current user to the notification sent when a new listed version is submitted on a particular add-on.

Note: Requires authentication and the current user to have any reviewer-related permission.

POST /api/v4/reviewers/addon/(**int:** *addon_id*)**/subscribe/**

Unsubscribe

This endpoint allows you to unsubscribe the current user to the notification sent when a new listed version is submitted on a particular add-on.

Note: Requires authentication and the current user to have any reviewer-related permission.

POST /api/v4/reviewers/addon/(**int:** *addon_id*)**/unsubscribe/**

Disable

This endpoint allows you to disable the public listing for an add-on.

Note:

Requires authentication and the current user to have `Reviews:Admin` permission.

POST /api/v4/reviewers/addon/(**int:** *addon_id*)**/disable/**

Enable

This endpoint allows you to re-enable the public listing for an add-on. If the add-on can't be public because it does not have public versions, it will instead be changed to awaiting review or incomplete depending on the status of its versions.

Note: Requires authentication and the current user to have `Reviews:Admin` permission.

POST /api/v4/reviewers/addon/(**int:** *addon_id*)**/enable/**

Flags

This endpoint allows you to manipulate various reviewer-specific flags on an add-on.

Note: Requires authentication and the current user to have `Reviews:Admin` permission.

PATCH /api/v4/reviewers/addon/(int: *addon_id*)/flags/

Response JSON Object

- **auto_approval_disabled** (*boolean*) – Boolean indicating whether auto approval are disabled on an add-on or not. When it's `true`, new versions for this add-on will make it appear in the regular reviewer queues instead of being auto-approved.
- **auto_approval_disabled_until_next_approval** (*boolean*) – Boolean indicating whether auto approval are disabled on an add-on until the next version is approved or not. Has the same effect as `auto_approval_disabled` but is automatically reset to `false` when the latest version of the add-on is manually approved by a human reviewer.
- **auto_approval_delayed_until** (*string|null*) – Date until the add-on auto-approval is delayed.
- **needs_admin_code_review** (*boolean*) – Boolean indicating whether the add-on needs its code to be reviewed by an admin or not.
- **needs_admin_content_review** (*boolean*) – Boolean indicating whether the add-on needs its content to be reviewed by an admin or not.
- **needs_admin_theme_review** (*boolean*) – Boolean indicating whether the theme needs to be reviewed by an admin or not.

Allow resubmission

This endpoint allows you to allow resubmission of an add-on that was previously denied.

Note: Requires authentication and the current user to have `Reviews:Admin` permission.

POST /api/v4/reviewers/addon/(int: *addon_id*)/allow_resubmission/

Status Codes

- 202 Accepted – Success.
- 409 Conflict – The add-on GUID was not previously denied.

Deny resubmission

This endpoint allows you to deny resubmission of an add-on that was not already denied.

Note: Requires authentication and the current user to have `Reviews:Admin` permission.

POST /api/v4/reviewers/addon/(int: *addon_id*)/deny_resubmission/

Status Codes

- 202 Accepted – Success.
- 409 Conflict – The add-on GUID was already denied.

List Versions

This endpoint allows you to list versions that can be used either for *browsing* or diffing versions.

> **Note:** Requires authentication and the current user to have `ReviewerTools:View` permission for listed add-ons as well as `Addons:ReviewUnlisted` for unlisted add-ons. Additionally the current user can also be the owner of the add-on.
>
> This endpoint is not paginated as normal, and instead will return all results without obeying regular pagination parameters.

If the user doesn't have `AddonsReviewUnlisted` permissions only listed versions are shown. Otherwise it can contain mixed listed and unlisted versions.

GET /api/v4/reviewers/addon/(**int:** *addon_id*)**/versions/**

Response JSON Object

- **id** (*int*) – The version id.
- **channel** (*string*) – The version channel, which determines its visibility on the site. Can be either `unlisted` or `listed`.
- **version** (*string*) – The version number string for the version.

Browse

This endpoint allows you to browse through the contents of an Add-on version.

> **Note:** Requires authentication and the current user to have `ReviewerTools:View` permission for listed add-ons as well as `Addons:ReviewUnlisted` for unlisted add-ons. Additionally the current user can also be the owner of the add-on.

GET /api/v4/reviewers/addon/(**int:** *addon_id*)**/versions/**
int: *version_id***/** Inherits the following properties from *version detail*: `id`, `channel`, `reviewed` and `version`.

Parameters

- **file** (*string*) – The specific file in the XPI to retrieve. Defaults to manifest.json, install.rdf or package.json for Add-ons as well as the XML file for search engines.
- **file_only** (*boolean*) – Indicates that the API should only return data for the requested file, and not version data. If this is `true` then the only property returned of those listed below is the `file` property.

Response JSON Object

- **validation_url_json** (*string*) – The absolute url to the addons-linter validation report, rendered as JSON.
- **validation_url** (*string*) – The absolute url to the addons-linter validation report, rendered as HTML.

- **has_been_validated** (*boolean*) – `True` if the version has been validated through addons-linter.
- **addon** (*object*) – A simplified *add-on* object that contains only a few properties: `id`, `name`, `icon_url` and `slug`.
- **file_entries[]** (*array*) – The complete file-tree of the extracted XPI.
- **file_entries[].depth** (*int*) – Level of folder-tree depth, starting with 0.
- **file_entries[].filename** (*string*) – The filename of the file.
- **file_entries[].path** (*string*) – The absolute path (from the root of the XPI) of the file.
- **file_entries[].mime_category** (*string*) – The mime type category of this file. Can be `image`, `directory`, `text` or `binary`.
- **file** (*object*) – The requested file.
- **file.id** (*int*) – The id of the submitted file (i.e., the xpi file).
- **file.content** (*string*) – Raw content of the requested file.
- **file.selected_file** (*string*) – The selected file, either from the `file` parameter or the default (manifest.json, install.rdf or package.json for Add-ons as well as the XML file for search engines).
- **file.download_url** (*string|null*) – The download url of the selected file or `null` in case of a directory.
- **file.mimetype** (*string*) – The determined mimetype of the selected file or `application/octet-stream` if none could be determined.
- **file.sha256** (*string*) – SHA256 hash of the selected file.
- **file.size** (*int*) – The size of the selected file in bytes.
- **file.filename** (*string*) – The filename of the file.
- **file.mime_category** (*string*) – The mime type category of this file. Can be `image`, `directory`, `text` or `binary`.
- **uses_unknown_minified_code** (*boolean*) – Indicates that the selected file could be using minified code.

Compare

This endpoint allows you to compare two Add-on versions with each other.

> **Note:** Requires authentication and the current user to have `ReviewerTools:View` permission for listed add-ons as well as `Addons:ReviewUnlisted` for unlisted add-ons. Additionally the current user can also be the owner of the add-on.

GET /api/v4/reviewers/addon/(**int:** *addon_id*)**/versions/**
int: *base_version_id***/compare_to/int:** *version_id***/**

> **Note:** Contrary to what `git diff` does, this API renders a hunk full of unmodified lines for unmodified files.

Inherits most properties from *browse detail*, except that most of the *file.entries[]* properties and *file.download_url* can be *null* in case of a deleted file.

Properties specific to this endpoint:

Response JSON Object

- **file_entries[]** (*array*) – The complete file-tree of the extracted XPI.
- **file.entries[].status** (*string*) – Status of this file, see https://git-scm.com/docs/git-status#_short_format
- **file_entries[].depth** (*int*) – Level of folder-tree depth, starting with 0.
- **file_entries[].filename** (*string*) – The filename of the file.
- **file_entries[].path** (*string*) – The absolute path (from the root of the XPI) of the file.
- **file_entries[].mime_category** (*string*) – The mime type category of this file. Can be `image`, `directory`, `text` or `binary`.
- **diff** (*object|null*) – See the following output with inline comments for a complete description.
- **base_file** (*object*) – The file attached to the base version you're comparing against.
- **base_file.id** (*object*) – The id of the base file.
- **uses_unknown_minified_code** (*boolean*) – Indicates that the selected file in either the current or the parent version could be using minified code.

Git patch we're talking about:

```
diff --git a/README.md b/README.md
index a37979d..b12683c 100644
--- a/README.md
+++ b/README.md
@@ -1 +1 @@
-# beastify
+Updated readme
diff --git a/manifest.json b/manifest.json
index aba695f..24f385f 100644
--- a/manifest.json
+++ b/manifest.json
@@ -1,36 +1 @@
-{
-
-  "manifest_version": 2,
-  "name": "Beastify",
-  "version": "1.0",
-
-  "permissions": [
-    "http://*/*",
-    "https://*/*",
-    "bookmarks",
-    "made up permission",
-    "https://google.com/"
-  ],
-
-  "content_scripts": [
-  {
```

(continues on next page)

(continued from previous page)

```
-    "matches": ["*://*.mozilla.org/*"],
-    "js": ["borderify.js"]
-  },
-  {
-    "matches": ["*://*.mozilla.com/*", "https://*.mozillians.org/*"],
-    "js": ["borderify.js"]
-  }
-  ],
-
-  "browser_action": {
-    "default_icon": "button/beasts.png",
-    "default_title": "Beastify",
-    "default_popup": "popup/choose_beast.html"
-  },
-
-  "web_accessible_resources": [
-    "beasts/*.jpg"
-  ]
-
-}
+{"id": "random"}
```

The following represents the git patch from above.

```
"diff": {
    "path": "README.md",
    "old_path": "README.md",
    "size": 15,  // Size in bytes
    "lines_added": 1,  // How many lines got added
    "lines_deleted": 1,  // How many lines got deleted
    "is_binary": false,  // Is this a binary file (as determined by git)
    "mode": "M",  // Status of this file, see https://git-scm.com/docs/git-status
↪#_short_format
    "hunks": [
        {
            "header": "@@ -1 +1 @@\\n",
            "old_start": 1,
            "new_start": 1,
            "old_lines": 1,
            "new_lines": 1,
            "changes": [
                {
                    "content": "# beastify\\n",
                    "type": "delete",
                    "old_line_number": 1,
                    "new_line_number": -1
                },
                {
                    "content": "Updated readme\\n",
                    "type": "insert",
                    "old_line_number": -1,
                    "new_line_number": 1
                }
            ]
        }
    ],
    "parent": "075c5755198be472522477a1b396951b3b68ac18",
```

(continues on next page)

(continued from previous page)

```
    "hash": "00161dcf22afb7bab23cf205f0c903eb5aad5431"
}
```

Canned Responses

This endpoint allows you to retrieve a list of canned responses.

Note: Requires authentication and the current user to have any reviewer-related permission.

GET /api/v4/reviewers/canned-responses/
Retrieve canned responses

Note: Because this endpoint is not returning too much data it is not paginated as normal, and instead will return all results without obeying regular pagination parameters.

Response JSON Object

- **id** (*int*) – The canned response id.
- **title** (*string*) – The title of the canned response.
- **response** (*string*) – The text that will be filled in as the response.
- **category** (*string*) – The category of the canned response. For example, "Other", "Privacy reasons" etc.

Drafting Comments

These endpoints allow you to draft comments that can be submitted through the regular reviewer pages.

Note: Requires authentication and the current user to have `ReviewerTools:View` permission for listed add-ons as well as `Addons:ReviewUnlisted` for unlisted add-ons. Additionally the current user can also be the owner of the add-on.

GET /api/v4/reviewers/addon/(int: *addon_id*) **/versions/**
int: *version_id***/draft_comments/** Retrieve existing draft comments for a specific version. See *pagination* for more details.

Response JSON Object

- **count** (*int*) – The number of comments for this version.
- **next** (*string*) – The URL of the next page of results.
- **previous** (*string*) – The URL of the previous page of results.
- **results** (*array*) – An array of *comments*.

GET /api/v4/reviewers/addon/(int: *addon_id*) **/versions/**
int: *version_id***/draft_comments/int:** *comment_id***/**

Response JSON Object

- **id** (*int*) – The id of the draft comment object.
- **comment** (*string*) – The comment that is being drafted as part of a review. Specific to a line in a file.
- **filename** (*string|null*) – The full file path a specific comment is related to. Can be `null` in case a comment doesn't belong to a specific file but the whole version.
- **lineno** (*int|null*) – The line number a specific comment is related to. Please make sure that in case of comments for git diffs, that the *lineno* used here belongs to the file in the version that belongs to *version_id* and not it's parent. Can be `null` in case a comment belongs to the whole file and not to a specific line.
- **version_id** (*int*) – The id of the version.
- **user.id** (*int*) – The id for an author.
- **user.name** (*string*) – The name for an author.
- **user.username** (*string*) – The username for an author.
- **user.url** (*string|null*) – The link to the profile page for an author, if the author's profile is public.
- **canned_response** (*object|null*) – Object holding the *canned response* if set.

POST /api/v4/reviewers/addon/(**int:** *addon_id*)**/versions/**
int: *version_id***/draft_comments/** Create a draft comment for a specific version.

Request JSON Object

- **comment** (*string*) – The comment that is being drafted as part of a review.
- **filename** (*string*) – The full file path this comment is related to. This must represent the full path, including sub-folders and relative to the root. E.g `lib/scripts/background.js`
- **lineno** (*int*) – The line number this comment is related to (optional). Please make sure that in case of comments for git diffs, that the *lineno* used here belongs to the file in the version that belongs to *version_id* and not it's parent.
- **canned_response** (*int*) – The id of the *canned response* (optional).

Status Codes

- 201 Created – New comment has been created.
- 400 Bad Request – An error occurred, check the *error* value in the JSON.
- 403 Forbidden – The user doesn't have the permission to create a comment. This might happen (among other cases) when someone without permissions for unlisted versions tries to add a comment for an unlisted version (which shouldn't happen as the user doesn't see unlisted versions, but it's blocked here too).

Response In case of successful creation, the response is a *draft comment object*.

DELETE /api/v4/reviewers/addon/(**int:** *addon_id*)**/versions/**
int: *version_id***/draft_comments/int:** *comment_id***/** Delete a draft comment.

Status Codes

- 204 No Content – The comment has been deleted successfully.
- 404 Not Found – The user doesn't have the permission to delete. This might happen when someone tries to delete a comment created by another reviewer or author.

PATCH /api/v4/reviewers/addon/(**int:** *addon_id*)**/versions/**
int: *version_id***/draft_comments/int:** *comment_id* Update a comment, it's filename or line number.

Request JSON Object

- **comment** (*string*) – The comment that is being drafted as part of a review.
- **filename** (*string*) – The full file path this comment is related to. This must represent the full path, including sub-folders and relative to the root. E.g `lib/scripts/background.js`
- **lineno** (*int*) – The line number this comment is related to. Please make sure that in case of comments for git diffs, that the *lineno* used here belongs to the file in the version that belongs to *version_id* and not it's parent.
- **canned_response** (*int*) – The id of the *canned response* (optional).

Status Codes

- 200 OK – The comment has been updated.
- 400 Bad Request – An error occurred, check the *error* value in the JSON.

Response In case of successful creation, the response is a *draft comment object*.

1.2.16 Scanners

Note: These APIs are subject to change at any time and are for internal use only.

Scanner Results

This endpoint returns a list of labelled scanner results.

Note: Requires authentication and the current user to have read access to the scanner results.

GET /api/v4/scanner/results/

Query Parameters

- **label** (*string*) – Filter by label.
- **scanner** (*string*) – Filter by scanner name.

Response JSON Object

- **id** (*int*) – The scanner result ID.
- **scanner** (*string*) – The scanner name.
- **label** (*string*) – Either `good` or `bad`.
- **results** (*object*) – The scanner (raw) results.
- **created** (*string*) – The date the result was created, formatted with this format.
- **model_version** (*string|null*) – The model version when applicable, `null` otherwise.

1.2.17 Signing

Note: These APIs are not frozen and can change at any time without warning. See *the API versions available* for alternatives if you need stability.

The following API endpoints help you get your add-on signed by Mozilla so it can be installed into Firefox without error. See extension signing for more details about Firefox's signing policy.

Client Libraries

The following libraries will make it easier to use the signing API:

- sign-addon, for general programattic use in NodeJS
- web-ext sign, for developing Web Extensions

If you are using `curl` to interact with the API you should be sure to pass the `-g` flag to skip "URL globbing" which won't interact well with add-on Ids that have { } characters in them.

Uploading a version

You can upload a new version for signing by issuing a `PUT` request and including the contents of your add-on in the `upload` parameter as multi-part formdata. This will create a pending version on the add-on and will prevent future submissions to this version unless validation or review fails.

If the upload succeeded then it will be submitted for validation and you will be able to check its status.

PUT /api/v4/addons/(string: *guid*)/versions/string: *version*/ Request:

```
curl "https://addons.mozilla.org/api/v4/addons/@my-addon/versions/1.0/"
    -g -XPUT --form "upload=@build/my-addon.xpi"
    -H "Authorization: JWT <jwt-token>"
```

Parameters

- **guid** – The add-on extension identifier.
- **version** – The version of the add-on.

Form Parameters

- **upload** – The add-on file being uploaded.
- **channel** – (optional) The channel this version should be uploaded to, which determines its visibility on the site. It can be either `unlisted` or `listed`. See the note below.

Request Headers

- Content-Type – multipart/form-data

Note: `channel` is only valid for new versions on existing add-ons. If the add-on is new then the version will be created as `unlisted`. If the parameter isn't supplied then the channel of the most recent version (submitted either via this API or the website) will be assumed. For example, if you submit a version as `listed` then the next version will be `listed` if you don't specify the channel.

Response:

The response body will be the same as the *Checking the status of your upload* response.

Status Codes

- 201 Created – new add-on and version created.
- 202 Accepted – new version created.
- 400 Bad Request – an error occurred, check the *error* value in the JSON.
- 401 Unauthorized – authentication failed.
- 403 Forbidden – you do not own this add-on.
- 409 Conflict – version already exists.

Uploading without an ID

Note: This is only valid for WebExtensions. All other add-on types require an add-on ID and have to use the regular endpoint to *upload a version*.

`POST /api/v4/addons/`

Request:

```
curl "https://addons.mozilla.org/api/v4/addons/"
    -g -XPOST -F "upload=@build/my-addon.xpi" -F "version=1.0"
    -H "Authorization: JWT <jwt-token>"
```

Form Parameters

- **`upload`** – The add-on file being uploaded.
- **`version`** – The version of the add-on.

Request Headers

- Content-Type – multipart/form-data

Response:

The response body will be the same as the *Checking the status of your upload* response.

Status Codes

- 201 Created – new add-on and version created.
- 202 Accepted – new version created.
- 400 Bad Request – an error occurred, check the *error* value in the JSON.
- 401 Unauthorized – authentication failed.
- 403 Forbidden – you do not own this add-on.
- 409 Conflict – version already exists.

Creating an add-on

If this is the first time that your add-on's UUID has been seen then the add-on will be created as an unlisted add-on when the version is uploaded.

Checking the status of your upload

You can check the status of your upload by issuing a `GET` request. There are a few things that will happen once a version is uploaded and the status of those events is included in the response.

Once validation is completed (whether it passes or fails) then the `processed` property will be `true`. You can check if validation passed using the `valid` property and check the results with `validation_results`.

If validation passed then your add-on will be submitted for automated or manual review. Once review is complete then then `reviewed` property will be set and you can check the results with the `passed_review` property.

GET /api/v4/addons/(string: *guid*)/versions/(string: *version*)/[uploads/(string:upload-pk)/] Request:

```
curl "https://addons.mozilla.org/api/v4/addons/@my-addon/versions/1.0/"
    -g -H "Authorization: JWT <jwt-token>"
```

Parameters

- **guid** – The add-on extension identifier.
- **version** – the version of the add-on.
- **upload-pk** – (optional) the pk for a specific upload.

Response:


```
{
    "guid": "420854ee-7a85-42b9-822f-8e03dc5f6de9",
    "active": true,
    "automated_signing": true,
    "files": [
        {
            "download_url": "https://addons.mozilla.org/api/v4/downloads/file/100/
↪example-id.0-fx+an.xpi?src=api",
            "hash":
↪"sha256:1bb945266bf370170a656350d9b640cbcaf70e671cf753c410e604219cdd9267",
            "signed": true
        }
    ],
    "passed_review": true,
    "pk": "f68abbb3b1624c098fe979a409fe3ce9",
    "processed": true,
    "reviewed": true,
    "url": "https://addons.mozilla.org/api/v4/addons/@example-id.0/uploads/
↪f68abbb3b1624c098fe979a409fe3ce9/",
    "valid": true,
    "validation_results": {},
    "validation_url": "https://addons.mozilla.org/en-US/developers/upload/
↪f68abbb3b1624c098fe979a409fe3ce9",
    "version": "1.0"
}
```

Response JSON Object

- **guid** – The GUID of the addon.
- **active** – version is active.
- **automated_signing** – If true, the version will be signed automatically. If false it will end up in the manual review queue when valid.
- **files[].download_url** – URL to *download the add-on file*.
- **files[].hash** – Hash of the file contents, prefixed by the hashing algorithm used. Example: `sha256:1bb945266bf3701...` . In the case of signed files, the hash will be that of the final signed file, not the original unsigned file.
- **files[].signed** – if the file is signed.
- **passed_review** – if the version has passed review.
- **pk** – the pk for this upload.
- **processed** – if the version has been processed by the validator.
- **reviewed** – if the version has been reviewed.
- **url** – URL to check the status of this upload.
- **valid** – if the version passed validation.
- **validation_results** – the validation results (removed from the example for brevity).
- **validation_url** – a URL to the validation results in HTML format.
- **version** – the version.

Status Codes

- 200 OK – request successful.
- 401 Unauthorized – authentication failed.
- 403 Forbidden – you do not own this add-on.
- 404 Not Found – add-on or version not found.

Downloading signed files

When checking on your *request to sign a version*, a successful response will give you an API URL to download the signed files. This endpoint returns the actual file data for download.

GET /api/v4/file/[int:file_id]/[string:base_filename]

Request:

```
curl "https://addons.mozilla.org/api/v4/file/123/some-addon.xpi?src=api"
    -g -H "Authorization: JWT <jwt-token>"
```

Parameters

- **file_id** – the primary key of the add-on file.
- **base_filename** – the base filename. This is just a convenience for clients so that they write meaningful file names to disk.

Response:

There are two possible responses:

- Binary data containing the file
- A header that redirects you to a mirror URL for the file. In this case, the initial response will include a `SHA-256` hash of the file in the header `X-Target-Digest`. Clients should check that the final downloaded file matches this hash.

Status Codes

- 200 OK – request successful.
- 302 Found – file resides at a mirror URL
- 401 Unauthorized – authentication failed.
- 404 Not Found – file does not exist or requester does not have access to it.

1.3 Server Install

The following documentation covers how to install and develop the server using Docker.

1.3.1 Install with Docker

Want the easiest way to start contributing to AMO? Try our Docker-based development environment.

First you'll need to install Docker. Please read their docs for the installation steps specific to your operating system.

There are two options for running docker depending on the platform you are running.

- **Run docker on the host machine directly (recommended)**
  - Update default settings in Docker Desktop - we suggest increasing RAM limit to at least 4 GB in the Resources/Advanced section and click on "Apply and Restart".
- Run docker-machine which will run docker inside a virtual-machine

Historically Mac and Windows could only run Docker via a vm. That has recently changed with the arrival of docker-for-mac and docker-for-windows.

If your platform can run Docker directly either on Linux, with docker-for-mac or docker-for-windows then this is the easiest way to run `addons-server`.

If you have problems, due to not meeting the minimum specifications for docker-for-windows or you'd prefer to keep running docker-machine vms then docker-machine will still work just fine. See the docs for creating the vm here *Creating the docker-machine vm*

Note: If you're on a Mac and already have a working docker-machine setup you can run that and docker-for-mac (*but not docker-for-windows*) side by side. The only caveat is it's recommended that you keep the versions of Docker on the vm and the host in-sync to ensure compatibility when you switch between them.

Setting up the containers

Note: docker-toolbox, docker-for-mac and docker-for-windows will install `docker-compose` for you. If you're on Linux and you need it, you can install it manually with:

```
pip install docker-compose
```

Note: On Windows, ensure that Docker Desktop is running as Linux Container. You can double-check this by ensuring you see "Switch to Windows containers..." in the resulting context menu when right-clicking on the Docker Desktop icon in the task-bar.

For more information see switching docker containers.

Failure to do so will result in errors in later steps like `make initialize`:

```
ValueError: Unable to configure handler 'statsd': [Errno -2] Name or service not known
Makefile-docker:71: recipe for target 'initialize_db' failed
make: *** [initialize_db] Error 1
```

Next once you have Docker up and running follow these steps on your host machine:

```
# Checkout the addons-server sourcecode.
git clone git://github.com/mozilla/addons-server.git
cd addons-server
# Download the containers
docker-compose pull  # Can take a while depending on your internet bandwidth.
# Start up the containers
docker-compose up -d
make initialize  # Answer yes, and create your superuser when asked.
# On Windows you can substitute `make initialize` for the command:
docker-compose exec web make initialize
```

Note: Docker requires the code checkout to exist within your home directory so that Docker can mount the source-code into the container.

Accessing the web server

By default our docker-compose config exposes the web-server on port 80 of localhost.

We use `olympia.test` as the default hostname to access your container server (e.g. for Firefox Accounts). To be able access the development environment using `http://olympia.test` you'll need to edit your `/etc/hosts` file on your native operating system. For example:

```
[ip-address]  olympia.test
```

Typically the IP address is localhost (127.0.0.1) but if you're using docker-machine see *Accessing the web-server with docker-machine* for details of how to get the ip of the Docker vm.

By default we configure *OLYMPIA_SITE_URL* to point to *http://olympia.test*.

If you choose a different hostname you'll need to set that environment variable and restart the Docker containers:

```
docker-compose stop # only needed if running
export OLYMPIA_SITE_URL=http://[YOUR_HOSTNAME}
docker-compose up -d
```

Running common commands

Run the tests using `make`, *outside* of the Docker container:

```
make test
# or
docker-compose exec web pytest src/olympia/
```

You can run commands inside the Docker container by `ssh`ing into it using:

```
make shell
# or
docker-compose exec web bash
```

Then to run the tests inside the Docker container you can run:

```
pytest
```

You can also run single commands from your host machine without opening a shell on each container. Here is an example of running the `pytest` command on the `web` container:

```
docker-compose run web pytest
```

If you'd like to use a python debugger to interactively debug Django view code, check out the *Debugging* section.

Note: If you see an error like `No such container: addonsserver_web_1` and your containers are running you can overwrite the base name for docker containers with the `COMPOSE_PROJECT_NAME` environment variable. If your container is named `localaddons_web_1` you would set `COMPOSE_PROJECT_NAME=localaddons`.

Updating your containers

Any time you update Olympia (e.g., by running `git pull`), you should make sure to update your Docker image and database with any new requirements or migrations:

```
docker-compose stop
docker-compose pull
docker-compose up -d
make update_docker  # Runs database migrations and rebuilds assets.
# On Windows you can substitute `make update_docker` for the following two commands:
docker-compose exec worker make update_deps
docker-compose exec web make update
```

Gotchas!

Here's a list of a few of the issues you might face when using Docker.

Can't access the web server?

Check you've created a hosts file entry pointing `olympia.test` to the relevant IP address.

If containers are failing to start use `docker-compose ps` to check their running status.

Another way to find out what's wrong is to run `docker-compose logs`.

Getting "Programming error [table] doesn't exist"?

Make sure you've run the `make initialize` step as detailed in the initial setup instructions.

ConnectionError during initialize (elasticsearch container fails to start)

When running `make initialize` without a working elasticsearch container, you'll get a ConnectionError. Check the logs with `docker-compose logs`. If elasticsearch is complaining about `vm.max_map_count`, run this command on your computer or your docker-machine VM:

```
sudo sysctl -w vm.max_map_count=262144
```

This allows processes to allocate more memory map areas.

Connection to elasticsearch timed out (elasticsearch container exits with code 137)

`docker-compose up -d` brings up all containers, but running `make initialize` causes the elasticsearch container to go down. Running `docker-compose ps` shows `Exited (137)` against it.

Update default settings in Docker Desktop - we suggest increasing RAM limit to at least 4 GB in the Resources/Advanced section and click on "Apply and Restart".

Port collisions (nginx container fails to start)

If you're already running a service on port 80 or 8000 on your host machine, the `nginx` container will fail to start. This is because the `docker-compose.override.yml` file tells `nginx` to listen on port 80 and the web service to listen on port 8000 by default.

This problem will manifest itself by the services failing to start. Here's an example for the most common case of `nginx` not starting due to a collision on port 80:

```
ERROR: for nginx  Cannot start service nginx:.....
...Error starting userland proxy: Bind for 0.0.0.0:80: unexpected error (Failure␣
↪EADDRINUSE)
ERROR: Encountered errors while bringing up the project.
```

You can check what's running on that port by using (sudo is required if you're looking at port < 1024):

```
sudo lsof -i :80
```

We specify the ports `nginx` listens on in the `docker-compose.override.yml` file. If you wish to override the ports you can do so by creating a new `docker-compose` config and starting the containers using that config alongside the default config.

For example if you create a file called `docker-compose-ports.yml`:

```
nginx:
  ports:
    - 8880:80
```

Next you would stop and start the containers with the following:

```
docker-compose stop # only needed if running
docker-compose -f docker-compose.yml -f docker-compose-ports.yml up -d
```

Now the container `nginx` is listening on 8880 on the host. You can now proxy to the container `nginx` from the host `nginx` with the following `nginx` config:

```
server {
    listen       80;
    server_name  olympia.test;
    location / {
        proxy_pass   http://olympia.test:8880;
    }
}
```

Persisting changes

Please note: any command that would result in files added or modified outside of the `addons-server` folder (e.g. modifying pip or npm dependencies) won't persist, and thus won't survive after the running container exits.

Note: If you need to persist any changes to the image, they should be carried out via the `Dockerfile`. Commits to master will result in the Dockerfile being rebuilt on the Docker hub.

Restarting docker-machine vms following a reboot

If you quit docker-machine, or restart your computer, docker-machine will need to start again using:

```
docker-machine start addons-dev
```

You'll then need to *export the variables* again, and start the services:

```
docker-compose up -d
```

Hacking on the Docker image

If you want to test out changes to the Olympia Docker image locally, use the normal Docker commands such as this to build a new image:

```
cd addons-server
docker build -t addons/addons-server .
docker-compose up -d
```

After you test your new image, commit to master and the image will be published to Docker Hub for other developers to use after they pull image changes.

1.3.2 Installation with Docker machine

Creating the docker-machine vm

Your first step is to create a vm - this step assumes we're using virtualbox as the driver:

```
docker-machine create --driver=virtualbox addons-dev
```

Then you can export the variables so that docker-compose can talk to the docker service. This command will tell you how to do that:

```
docker-machine env addons-dev
```

On a mac it's a case of running:

```
eval $(docker-machine env addons-dev)
```

Now you have the vm running you can follow the standard docker instructions: *Install with Docker*

Accessing the web-server with docker-machine

If you're using docker-machine, you can get the ip like so:

```
docker-machine ip addons-dev
```

Note: If you're still using boot2docker then the command is *boot2docker ip*. At this point you can look at installing docker-toolbox and migrating your old boot2docker vm across to running via docker-machine. See docker-toolbox for more info.

Now you can connect to port 80 of that ip address. Here's an example (your ip might be different):

```
http://192.168.99.100/
```

Note: `docker-machine` hands out IP addresses as each VM boots; your IP addresses can change over time. You can find out which IP is in use using `docker-machine ip [machine name]`

The following documentation covers how to install the individual components, this documentation is deprecated in favour of using Docker. This documentation may be out of date or incomplete.

1.3.3 Install Olympia manually (deprecated)

The approved installation is *via Docker*. The following pages might be helpful but are deprecated and may be out of date.

Installing Olympia the long way

Note: The following documentation is deprecated. The approved installation is *via Docker*.

The following instructions walk you through installing and configuring all required services from scratch.

We're going to use all the hottest tools to set up a nice environment. Skip steps at your own peril. Here we go!

Requirements

To get started, you'll need:

- Python 2.7 (2.7 -> 2.7.10)
- Node 0.10.x or higher
- MySQL
- ElasticSearch
- libxml2 (for building lxml, used in tests)

OS X and *Ubuntu* instructions follow.

There are a lot of advanced dependencies we're going to skip for a fast start. They have their own *section*.

If you're on a Linux distro that splits all its packages into `-dev` and normal stuff, make sure you're getting all those `-dev` packages.

On Ubuntu

The following command will install the required development files on Ubuntu or, if you're running a recent version, you can install them automatically:

```
sudo apt-get install python-dev python-virtualenv libxml2-dev libxslt1-dev␣
↪libmysqlclient-dev memcached libssl-dev swig openssl curl libjpeg-dev zlib1g-dev␣
↪libsasl2-dev nodejs nodejs-legacy
```

Note: As of writing, M2Crypto is only compatible with swig <=3.0.4 version's. So, if you encounter a libssl exception while running `make full_init`, you might have to downgrade swig to version <=3.0.4.

On OS X

The best solution for installing UNIX tools on OS X is Homebrew.

The following packages will get you set for olympia:

```
brew install python libxml2 mysql libmemcached openssl swig jpeg
```

Note: As of writing, M2Crypto is only compatible with swig <=3.0.4 version's. So, if you encounter a libssl exception while running `make full_init`, you might have to downgrade swig to version <=3.0.4.

MySQL

You'll probably need to *configure MySQL after install* (especially on Mac OS X) according to advanced installation.

See *Database* for creating and managing the database.

Elasticsearch

You'll need an Elasticsearch server up and running during the init script. See *Elasticsearch* for more instructions.

Use the Source

Grab olympia from github with:

```
git clone git://github.com/mozilla/olympia.git
cd olympia
```

`olympia.git` is all the source code. *Updating* is detailed later on.

virtualenv and virtualenvwrapper

virtualenv is a tool to create isolated Python environments. This will let you put all of Olympia's dependencies in a single directory rather than your global Python directory. For ultimate convenience, we'll also use virtualenvwrapper which adds commands to your shell.

Are you ready to bootstrap virtualenv and virtualenvwrapper? Since each shell setup is different, you can install everything you need and configure your shell using the virtualenv-burrito. Type this:

```
curl -sL https://raw.github.com/brainsik/virtualenv-burrito/master/virtualenv-burrito.
↪sh | $SHELL
```

Open a new shell to test it out. You should have the `workon` and `mkvirtualenv` commands.

virtualenvwrapper Hooks (optional)

virtualenvwrapper lets you run hooks when creating, activating, and deleting virtual environments. These hooks can change settings, the shell environment, or anything else you want to do from a shell script. For complete hook documentation, see http://www.doughellmann.com/docs/virtualenvwrapper/hooks.html.

You can find some lovely hooks to get started at http://gist.github.com/536998. The hook files should go in `$WORKON_HOME` (`$HOME/Envs` from above), and `premkvirtualenv` should be made executable.

Getting Packages

Now we're ready to go, so create an environment for olympia:

```
mkvirtualenv olympia
```

That creates a clean environment named olympia using your default python. You can get out of the environment by restarting your shell or calling `deactivate`.

To get back into the olympia environment later, type:

```
workon olympia  # requires virtualenvwrapper
```

Note: Olympia requires Python 2.7.

Note: If you want to use a different Python binary, pass the name (if it is on your path) or the full path to mkvirtualenv with `--python`:

```
mkvirtualenv --python=/usr/local/bin/python2.7 olympia
```

Finish the install

First make sure you have a recent pip for security reasons:

```
pip install --upgrade pip
```

From inside your activated virtualenv, install the required python packages, initialize the database, create a super user, compress the assets, . . . :

```
make full_init
```

Settings

Most of olympia is already configured in `settings.py`, but there's some things you may want to configure locally. All your local settings go into `local_settings.py`. The settings template for developers, included below, is at docs/settings/local_settings.dev.py.

```
from settings import *  # noqa

INTERNAL_IPS = ('127.0.0.1', )
```

I'm extending `INSTALLED_APPS` and `MIDDLEWARE_CLASSES` to include the Django Debug Toolbar. It's awesome, you want it.

The file `local_settings.py` is for local use only; it will be ignored by git.

Database

By default, Olympia connects to the `olympia` database running on `localhost` as the user `root`, with no password. To create a database, run:

```
$ mysql -u root -p
mysql> CREATE DATABASE olympia CHARACTER SET utf8 COLLATE utf8_unicode_ci;
```

If you want to change settings, you can either add the database settings in your *local_settings.py* or set the environment variable `DATABASE_URL`:

```
export DATABASES_DEFAULT_URL=mysql://<user>:<password>@<hostname>/<database>
```

If you've changed the user and password information, you need to grant permissions to the new user:

```
$ mysql -u root -p
mysql> GRANT ALL ON olympia.* TO <YOUR_USER>@localhost IDENTIFIED BY '<YOUR_PASSWORD>
↪';
```

Finally, to run the test suite, you'll need to add an extra grant in MySQL for your database user:

```
$ mysql -u root -p
mysql> GRANT ALL ON test_olympia.* TO <YOUR_USER>@localhost IDENTIFIED BY '<YOUR_
↪PASSWORD>';
```

Warning: Don't forget to change `<YOUR_USER>` and `<YOUR_PASSWORD>` to your actual database credentials.

The database is initialized automatically using the `make full_init` command you saw earlier.

Database Migrations

Each incremental change we add to the database is done with a versioned SQL (and sometimes Python) file. To keep your local DB fresh and up to date, run migrations like this:

```
$ schematic migrations/
```

If, at some point, you want to start from scratch and recreate the database, you can just run the `make initialize_db` command. This will also fake all the schematic migrations, and allow you to create a superuser.

Run the Server

If you've gotten the system requirements, downloaded `olympia`, set up your virtualenv with the compiled packages, and configured your settings and database, you're good to go.

```
./manage.py runserver
```

Note: If you don't have a LESS compiler already installed, opening http://localhost:8000 in your browser will raise a 500 server error. If you don't want to run through the *Manual installation* documentation just right now, you can disable all LESS pre-processing by adding the following line to your `local_settings.py` file:

```
LESS_PREPROCESS = False
```

Be aware, however, that this will make the site VERY slow, as a huge amount of LESS files will be served to your browser on EACH request, and each of those will be compiled on the fly by the LESS javascript compiler.

Create an Admin User

To connect to the site, you first need to register a new user "the standard way" by filling in the registration form.

Once this is done, you can either activate this user using the link in the confirmation email sent (it's displayed in the console, check your server logs), or use the following handy management command:

```
./manage.py activate_user <email of your user>
```

If you want to grant yourself admin privileges, pass in the `--set-admin` option:

```
./manage.py activate_user --set-admin <email of your user>
```

Updating

To run a full update of olympia (including source files, pip requirements and database migrations):

```
make full_update
```

If you want to do it manually, then check the Makefile.

The *Contributing* page has more on managing branches.

Contact

Come talk to us on https://chat.mozilla.org/#/room/#amo:mozilla.org if you have questions, issues, or compliments.

Submitting a Patch

See the *Contributing* page.

Advanced Installation

In production we use things like memcached, rabbitmq + celery, elasticsearch, LESS, and Stylus. Learn more about installing these on the *Manual installation* page.

Note: Although we make an effort to keep advanced items as optional installs you might need to install some components in order to run tests or start up the development server.

Manual installation

Note: The following documentation is deprecated. The approved installation is *via Docker*.

Getting Fancy

MySQL

On your dev machine, MySQL probably needs some tweaks. Locate your my.cnf (or create one) then, at the very least, make UTF8 the default encoding:

```
[mysqld]
character-set-server=utf8
```

Here are some other helpful settings:

```
[mysqld]
default-storage-engine=innodb
character-set-server=utf8
skip-sync-frm=OFF
innodb_file_per_table
```

On Mac OS X with homebrew, put my.cnf in `/usr/local/Cellar/mysql/5.5.15/my.cnf` then restart like:

```
launchctl unload -w ~/Library/LaunchAgents/com.mysql.mysqld.plist
launchctl load -w ~/Library/LaunchAgents/com.mysql.mysqld.plist
```

Note: some of the options above were renamed between MySQL versions

---

Here are more tips for optimizing MySQL on your dev machine.

Memcached

We slipped this in with the basic install. The package was `memcached` on Ubuntu and `libmemcached` on OS X. Your default settings already use the following, so you shouldn't need to change anything:

```
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': ['localhost:11211'],
        'TIMEOUT': 500,
    }
}
```

RabbitMQ and Celery

See the *Celery* page for installation instructions. The *example settings* set `CELERY_TASK_ALWAYS_EAGER = True`. If you're setting up Rabbit and want to use `celery`, make sure you remove that line from your `local_settings.py`.

Elasticsearch

See *Elasticsearch* for more instructions.

Redis

On OS X the package is called `redis`. Get it running with the `launchctl` script included in homebrew. To let olympia know about Redis, add this to `local_settings.py`:

```
CACHE_MACHINE_USE_REDIS = True
REDIS_BACKEND = 'redis://'
```

The `REDIS_BACKEND` is parsed like `CACHE_BACKEND` if you need something other than the default settings.

Node.js

Node.js is needed for Stylus and LESS, which in turn are needed to precompile the CSS files.

If you want to serve the CSS files from another domain than the webserver, you will need to precompile them. Otherwise you can have them compiled on the fly, using javascript in your browser, if you set `LESS_PREPROCESS = False` in your local settings.

First, we need to install node and npm:

```
brew install node
curl https://www.npmjs.org/install.sh | sh
```

Optionally make the local scripts available on your path if you don't already have this in your profile:

```
export PATH="./node_modules/.bin/:${PATH}"
```

Not working?

- If you're having trouble installing node, try http://shapeshed.com/journal/setting-up-nodejs-and-npm-on-mac-osx/. You need brew, which we used earlier.
- If you're having trouble with npm, check out the README on https://github.com/isaacs/npm

Stylus CSS

Learn about Stylus at http://learnboost.github.com/stylus/

```
cd olympia
npm install
```

In your `local_settings.py` ensure you are pointing to the correct executable for `stylus`:

```
STYLUS_BIN = path('node_modules/stylus/bin/stylus')
```

LESS CSS

We're slowing switching over from regular CSS to LESS. You can learn more about LESS at http://lesscss.org.

If you already ran `npm install` you don't need to do anything more.

In your `local_settings.py` ensure you are pointing to the correct executable for `less`:

```
LESS_BIN = path('node_modules/less/bin/lessc')
```

You can make the CSS live refresh on save by adding `#!watch` to the URL or by adding the following to your `local_settings.py`:

```
LESS_LIVE_REFRESH = True
```

If you want syntax highlighting, try:

- vim: http://leafo.net/lessphp/vim/
- emacs: http://jdhuntington.com/emacs/less-css-mode.el
- TextMate: https://github.com/appden/less.tmbundle
- Coda: http://groups.google.com/group/coda-users/browse_thread/thread/b3327b0cb893e439?pli=1

Generating additional add-ons

Note: If you previously used the `make full_init` command during the *Installing Olympia the long way* process, it's not necessary to generate additional add-ons for initialisation/development purpose.

If you need more add-ons, you can generate additional ones using the following command:

```
python manage.py generate_addons <num_addons> [--owner <email>] [--app <application>]
```

where `num_addons` is the number of add-ons that you want to generate, `email` (optional) is the email address of the owner of the generated add-ons and `application` (optional) the name of the application (either `firefox`, `thunderbird`, `seamonkey` or `android`).

By default the `email` will be `nobody@mozilla.org` and the `application` will be `firefox` if not specified.

Add-ons will have 1 preview image, 2 translations (French and Spanish), 5 ratings and might be featured randomly.

If you didn't run the `make full_init` command during the *Installing Olympia the long way* process, categories from production (Alerts & Updates, Appearance, and so on) will be created and randomly populated with generated add-ons. Otherwise, the existing categories will be filled with newly generated add-ons.

Celery

Note: The following documentation is deprecated. The approved installation is *via Docker*.

Celery is a task queue powered by RabbitMQ. You can use it for anything that doesn't need to complete in the current request-response cycle. Or use it wherever Les tells you to use it.

For example, each addon has a `current_version` cached property. This query on initial run causes strain on our database. We can create a denormalized database field called `current_version` on the `addons` table.

We'll need to populate regularly so it has fairly up-to-date data. We can do this in a process outside the request-response cycle. This is where Celery comes in.

Installation

RabbitMQ

Celery depends on RabbitMQ. If you use `homebrew` you can install this:

```
brew install rabbitmq
```

Setting up rabbitmq invovles some configuration. You may want to define the following

```
# On a Mac, you can find this in System Preferences > Sharing
export HOSTNAME='<laptop name>.local'
```

Then run the following commands:

```
# Set your host up so it's semi-permanent
sudo scutil --set HostName $HOSTNAME

# Update your hosts by either:
# 1) Manually editing /etc/hosts
# 2) `echo 127.0.0.1 $HOSTNAME >> /etc/hosts`

# RabbitMQ insists on writing to /var
sudo rabbitmq-server -detached

# Setup rabitty things (sudo is required to read the cookie file)
sudo rabbitmqctl add_user olympia olympia
sudo rabbitmqctl add_vhost olympia
sudo rabbitmqctl set_permissions -p olympia olympia ".*" ".*" ".*"
```

Back in safe and happy django-land you should be able to run:

```
celery -A olympia worker -E
```

Celery understands python and any tasks that you have defined in your app are now runnable asynchronously.

Celery Tasks

Any python function can be set as a celery task. For example, let's say we want to update our `current_version` but we don't care how quickly it happens, just that it happens. We can define it like so:

```
@task(rate_limit='2/m')
def _update_addons_current_version(data, **kw):
    task_log.debug("[%s@%s] Updating addons current_versions." %
                   (len(data), _update_addons_current_version.rate_limit))
    for pk in data:
        try:
            addon = Addon.objects.get(pk=pk)
            addon.update_version()
        except Addon.DoesNotExist:
            task_log.debug("Missing addon: %d" % pk)
```

`@task` is a decorator for Celery to find our tasks. We can specify a `rate_limit` like `2/m` which means `celery` will only run this command 2 times a minute at most. This keeps write-heavy tasks from killing your database.

If we run this command like so:

```
from celery.task.sets import TaskSet

all_pks = Addon.objects.all().values_list('pk', flat=True)
ts = [_update_addons_current_version.subtask(args=[pks])
      for pks in amo.utils.chunked(all_pks, 300)]
TaskSet(ts).apply_async()
```

All the Addons with ids in `pks` will (eventually) have their `current_versions` updated.

Cron Jobs

This is all good, but let's automate this. In Olympia we can create cron jobs like so:

```
@cronjobs.register
def update_addons_current_version():
    """Update the current_version field of the addons."""
    d = Addon.objects.valid().exclude(
          type=amo.ADDON_PERSONA).values_list('id', flat=True)

    with establish_connection() as conn:
        for chunk in chunked(d, 1000):
            print chunk
            _update_addons_current_version.apply_async(args=[chunk],
                                                       connection=conn)
```

This job will hit all the addons and run the task we defined in small batches of 1000.

We'll need to add this to both the `prod` and `preview` crontabs so that they can be run in production.

Better than Cron

Of course, cron is old school. We want to do better than cron, or at least not rely on brute force tactics.

For a surgical strike, we can call `_update_addons_current_version` any time we add a new version to that addon. Celery will execute it at the prescribed rate, and your data will be updated ... eventually.

During Development

`celery` only knows about code as it was defined at instantiation time. If you change your `@task` function, you'll need to `HUP` the process.

However, if you've got the `@task` running perfectly you can tweak all the code, including cron jobs that call it without need of restart.

Elasticsearch

Note: The following documentation is deprecated. The approved installation is *via Docker*.

Elasticsearch is a search server. Documents (key-values) get stored, configurable queries come in, Elasticsearch scores these documents, and returns the most relevant hits.

Also check out elasticsearch-head, a plugin with web front-end to elasticsearch that can be easier than talking to elasticsearch over curl, or Marvel, which includes a query editors with autocompletion.

Installation

Elasticsearch comes with most package managers.:

```
brew install elasticsearch  # or whatever your package manager is called.
```

If Elasticsearch isn't packaged for your system, you can install it manually, here are some good instructions on how to do so.

On Ubuntu, you should just download and install a .deb from the download page.

Launching and Setting Up

Launch the Elasticsearch service. If you used homebrew, `brew info elasticsearch` will show you the commands to launch. If you used aptitude, Elasticsearch will come with a start-stop daemon in /etc/init.d. On Ubuntu, if you have installed from a .deb, you can type:

    sudo service elasticsearch start

Olympia has commands that sets up mappings and indexes objects such as add-ons and apps for you. Setting up the mappings is analogous to defining the structure of a table, indexing is analogous to storing rows.

For AMO, this will set up all indexes and start the indexing processeses:

```
./manage.py reindex
```

Or you could use the makefile target:

```
make reindex
```

If you need to add arguments:

```
make ARGS='--with-stats --wipe --force' reindex
```

Indexing

Olympia has other indexing commands. It is worth noting that the index is maintained incrementally through post_save and post_delete hooks:

```
./manage.py cron reindex_addons  # Index all the add-ons.

./manage.py index_stats  # Index all the update and download counts.

./manage.py cron reindex_collections  # Index all the collections.

./manage.py cron reindex_users  # Index all the users.

./manage.py cron compatibility_report  # Set up the compatibility index.

./manage.py weekly_downloads # Index weekly downloads.
```

Querying Elasticsearch in Django

For now, we have our own query builder (which is an historical clone of elasticutils), but we will switch to the official one very soon.

We attach elasticutils to Django models with a mixin. This lets us do things like `.search()` which returns an object which acts a lot like Django's ORM's object manager. `.filter(**kwargs)` can be run on this search object:

```
query_results = list(
    MyModel.search().filter(my_field=a_str.lower())
    .values_dict('that_field'))
```

Testing with Elasticsearch

All test cases using Elasticsearch should inherit from `amo.tests.ESTestCase`. All such tests are marked with the `es_tests` pytest marker. To run only those tests:

```
pytest -m es_tests
```

or

```
make test_es
```

Troubleshooting

I got a CircularReference error on .search() - check that a whole object is not being passed into the filters, but rather just a field's value.

I indexed something into Elasticsearch, but my query returns nothing - check whether the query contains upper-case letters or hyphens. If so, try lowercasing your query filter. For hyphens, set the field's mapping to not be analyzed:

```
'my_field': {'type': 'string', 'index': 'not_analyzed'}
```

Try running .values_dict on the query as mentioned above.

Trouble-shooting the development installation

Note: The following documentation is deprecated. The approved installation is *via Docker*.

Image processing isn't working

If adding images to apps or extensions doesn't seem to work then there's a couple of settings you should check.

Checking your image settings

Check that `CELERY_TASK_ALWAYS_EAGER` is set to `True` in your settings file. This means it will process tasks without a celery worker running:

```
CELERY_TASK_ALWAYS_EAGER = True
```

If that yields no joy you can try running a celery worker in the foreground, set `CELERY_TASK_ALWAYS_EAGER = False` and run:

```
celery -A olympia worker -E
```

Note: This requires a RabbitMQ server to be set up as detailed in the *Celery* instructions.

This may help you to see where the image processing tasks are failing. For example it might show that Pillow is failing due to missing dependencies.

Checking your Pillow installation (Ubuntu)

When you run you should see at least JPEG and ZLIB are supported

If that's the case you should see this in the output of `pip install -I Pillow`:

```
--------------------------------------------------------------------
*** TKINTER support not available
--- JPEG support available
--- ZLIB (PNG/ZIP) support available
*** FREETYPE2 support not available
```

(continues on next page)

(continued from previous page)

```
*** LITTLECMS support not available
--------------------------------------------------------------------
```

If you don't then this suggests Pillow can't find your image libraries:

To fix this double-check you have the necessary development libraries installed first (e.g: `sudo apt-get install libjpeg-dev zlib1g-dev`)

Now run the following for 32bit:

```
sudo ln -s /usr/lib/i386-linux-gnu/libz.so /usr/lib
sudo ln -s /usr/lib/i386-linux-gnu/libjpeg.so /usr/lib
```

Or this if your running 64bit:

```
sudo ln -s /usr/lib/x86_64-linux-gnu/libz.so /usr/lib
sudo ln -s /usr/lib/x86_64-linux-gnu/libjpeg.so /usr/lib
```

Note: If you don't know what arch you are running run `uname -m` if the output is `x86_64` then it's 64-bit, otherwise it's 32bit e.g. `i686`

Now re-install Pillow:

```
pip install -I Pillow
```

And you should see the necessary image libraries are now working with Pillow correctly.

ES is timing out

This can be caused by `number_of_replicas` not being set to 0 for the local indexes.

To check the settings run:

```
curl -s localhost:9200/_cluster/state\?pretty | fgrep number_of_replicas -B 5
```

If you see any that aren't 0 do the following:

Set `ES_DEFAULT_NUM_REPLICAS` to `0` in your local settings.

To set them to zero immediately run:

```
curl -XPUT localhost:9200/_settings -d '{ "index" : { "number_of_replicas" : 0 } }'
```

1.4 Development

1.4.1 Running Tests

Run tests from outside the docker container with `make`:

```
make test
```

Other, more niche, test commands:

```
make test_failed # rerun the failed tests from the previous run
make test_force_db # run the entire test suite with a new database
make tdd # run the entire test suite, but stop on the first error
```

Using pytest directly

For advanced users. To run the entire test suite you never need to use `pytest` directly.

You can connect to the docker container using `make shell`; then use `pytest` directly, which allows for finer-grained control of the test suite.

Run your tests like this:

```
pytest
```

For running subsets of the entire test suite, you can specify which tests run using different methods:

- *pytest -m es_tests* to run the tests that are marked as *es_tests*
- *pytest -k test_no_license* to run all the tests that have *test_no_license* in their name
- *pytest src/olympia/addons/tests/test_views.py::TestLicensePage::test_no_license* to run this specific test

For more, see the Pytest usage documentation.

1.4.2 Debugging

The *docker setup* uses supervisord to run the django runserver. This means if you want to access the management server from a shell to run things like ipdb you still can.

Using ipdb

As with ipdb normally just add a line in your code at the relevant point:

```
import ipdb; ipdb.set_trace()
```

Next connect to the running web container:

```
make debug
```

This will bring the Django management server to the foreground and you can interact with ipdb as you would normally. To quit you can just type `Ctrl+c`.

All being well it should look like this:

```
$ make debug
docker exec -t -i olympia_web_1 supervisorctl fg olympia
:/opt/rh/python27/root/usr/lib/python2.7/site-packages/celery/utils/__init__.py:93
11:02:08 py.warnings:WARNING /opt/rh/python27/root/usr/lib/python2.7/site-packages/
↪jwt/api_jws.py:118: DeprecationWarning: The verify parameter is deprecated. Please␣
↪use options instead.
'Please use options instead.', DeprecationWarning)
:/opt/rh/python27/root/usr/lib/python2.7/site-packages/jwt/api_jws.py:118
[21/Oct/2015 11:02:08] "PUT /en-US/firefox/api/v4/addons/%40unlisted/versions/0.0.5/␣
↪HTTP/1.1" 400 36
Validating models...
```

(continues on next page)

(continued from previous page)

```
0 errors found
October 21, 2015 - 13:52:07
Django version 1.6.11, using settings 'settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
[21/Oct/2015 13:57:56] "GET /static/img/app-icons/16/sprite.png HTTP/1.1" 200 3810
13:58:01 py.warnings:WARNING /opt/rh/python27/root/usr/lib/python2.7/site-packages/
↪celery/task/sets.py:23: CDeprecationWarning:
    celery.task.sets and TaskSet is deprecated and scheduled for removal in
    version 4.0. Please use "group" instead (see the Canvas section in the userguide)

""")
:/opt/rh/python27/root/usr/lib/python2.7/site-packages/celery/utils/__init__.py:93
> /code/src/olympia/browse/views.py(148)themes()
    147     import ipdb;ipdb.set_trace()
--> 148     TYPE = amo.ADDON_THEME
    149     if category is not None:

ipdb> n
> /code/src/olympia/browse/views.py(149)themes()
    148     TYPE = amo.ADDON_THEME
--> 149     if category is not None:
    150         q = Category.objects.filter(application=request.APP.id, type=TYPE)

ipdb>
```

Logging

Logs for the celery and Django processes can be found on your machine in the *logs* directory.

Using the Django Debug Toolbar

The Django Debug Toolbar is very powerful and useful when viewing pages from the website, to check the view used, its parameters, the SQL queries, the templates rendered and their context.

To use it please see the official getting started docs: https://django-debug-toolbar.readthedocs.io/en/1.4/installation.html#quick-setup

---

Note: You must know that using the Django Debug Toolbar will slow the website quite a lot. You can mitigate this by deselecting the checkbox next to the `SQL` panel.

Also, please note that you should only use the Django Debug Toolbar if you need it, as it makes CSP report only for your local dev.

Note: You might have to disable CSP by setting *CSP_REPORT_ONLY = True* in your local settings because django debug toolbar uses "data:" for its logo, and it uses "unsafe eval" for some panels like the templates or SQL ones.

1.4.3 Adding Python Dependencies

To add a new dependency you'll to carry out the following. First install hashin:

```
pip install hashin
```

Next add the dependency you want to add to the relevant requirements file.

Note: If you add just the package name the script will automatically get the latest version for you.

Once you've done that you can run the requirements script:

```
hashin -r <requirements file> <dependency>
```

This will add hashes and sort the requirements for you adding comments to show any package dependencies.

When it's run check the diff and make edits to fix any issues before submitting a PR with the additions.

1.4.4 Error Pages

When running Django locally you get verbose error pages instead of the standard ones. To access the HTML for the standard error pages, you can access the urls:

```
/services/403
/services/404
/services/500
```

1.4.5 Testing

We're using a mix of Django's Unit Testing and pytest with pytest-django for our automated testing. This gives us a lot of power and flexibility to test all aspects of the site.

Configuration

Configuration for your unit tests is handled automatically. The only thing you'll need to ensure is that the database credentials in your settings has full permissions to modify a database with `test_` prepended to it. By default the database name is `olympia`, so the test database is `test_olympia`. Optionally, in particular if the code you are working on is related to search, you'll want to run Elasticsearch tests. Obviously, you need Elasticsearch to be installed. See *Elasticsearch* page for details.

If you don't want to run the Elasticsearch tests, you can use the `test_no_es` target in the Makefile:

```
make test_no_es
```

On the contrary, if you only want to run Elasticsearch tests, use the `test_es` target:

```
make test_es
```

Running Tests

To run the whole test suite use:

```
pytest
```

There are a lot of options you can pass to adjust the output. Read pytest and pytest-django docs for the full set, but some common ones are:

- `-v` to provide more verbose information about the test run
- `-s` tells pytest to not capture the logging output
- `--create-db` tells pytest-django to recreate the database instead of reusing the one from the previous run
- `-x --pdb` to stop on the first failure, and drop into a python debugger
- `--lf` to re-run the last test failed
- `-m test_es` will only run tests that are marked with the `test_es` mark
- `-k foobar` will only run tests that contain `foobar` in their name

There are a few useful makefile targets that you can use:

Run all the tests:

```
make test
```

If you need to rebuild the database:

```
make test_force_db
```

To fail and stop running tests on the first failure:

```
make tdd
```

If you wish to add arguments, or run a specific test, overload the variables (check the Makefile for more information):

```
make test ARGS='-v src/olympia/amo/tests/test_url_prefix.py::MiddlewareTest::test_get_
↪app'
```

If you wish to re-run only the tests failed from the previous run:

```
make test_failed
```

Database Setup

Our test runner is configured by default to reuse the database between each test run. If you really want to make a new database (e.g. when models have changed), use the `--create-db` parameter:

```
pytest --create-db
```

or

```
make test_force_db
```

Writing Tests

We support two types of automated tests right now and there are some details below but remember, if you're confused look at existing tests for examples.

Also, take some time to get familiar with pytest way of dealing with dependency injection, which they call fixtures (which should not be confused with Django's fixtures). They are very powerful, and can make your tests much more independent, cleaner, shorter, and more readable.

Unit/Functional Tests

Most tests are in this category. Our test classes extend `django.test.TestCase` and follow the standard rules for unit tests. We're using JSON fixtures for the data.

External calls

Connecting to remote services in tests is not recommended, developers should mock out those calls instead.

Why Tests Fail

Tests usually fail for one of two reasons: The code has changed or the data has changed. An third reason is **time**. Some tests have time-dependent data usually in the fixtures. For example, some featured items have expiration dates.

We can usually save our future-selves time by setting these expirations far in the future.

Localization Tests

If you want test that your localization works then you can add in locales in the test directory. For an example see `devhub/tests/locale`. These locales are not in the normal path so should not show up unless you add them to the *LOCALE_PATH*. If you change the .po files for these test locales, you will need to recompile the .mo files manually, for example:

```
msgfmt --check-format -o django.mo django.po
```

1.4.6 Style Guide

Writing code for olympia? Awesome! Please help keep our code readable by, whenever possible, adhering to these style conventions.

Python

- see https://wiki.mozilla.org/Webdev:Python

Markup

- `<!DOCTYPE html>`
- double-quote attributes
- Soft tab (2 space) indentation
- Title-Case `<label>` tags - “Display Name” vs “Display name”
- to clearfix, use the class `c` on an element

JavaScript

- Soft tabs (4 space) indentation
- Single quotes around strings (unless the string contains single quotes)
- variable names for jQuery objects start with $. for example:
  - `var $el = $(el);`
- Element IDs and classes that are not generated by Python should be separated by hyphens, eg: #some-module.
- Protect all functions and objects from global scope to avoid potential name collisions. When exposing functions to other scripts use the `z` namespace.
- Always protect code from loading on pages it’s not supposed to load on. For example:

```
$(document).ready(function() {
    if ($('#something-on-your-page').length) {
        initYourCode();
    }

    function initYourCode() {
        // ...
    }
});
```

1.4.7 Contributing

If you’re not sure this is the correct place to file an issue then please file an issue on the mozilla/addons project instead.

Before contributing code, please note:

- You agree to license your contributions under the license.
- Please ask on the dev-addons mailing list before submitting pull-requests for new features or large changes that are not related to existing issues.
- Follow PEP8, jshint and our other style guide conventions.
- Please write tests and read the docs on addons-server.

Ready to get started? Follow these steps.

Note to staff: If you come across a potential “good first bug” for contributors, please tag it with “**maybe good first bug**”. The community team triages these every other week to ensure they have mentors assigned, onboarding information, and basic steps to get started. This gives new contributors a better experience when they pick a “good first bug” to work on.

Thank you for contributing!

1.4.8 Push From Master

We deploy from the master branch once a month. If you commit something to master that needs additional QA time, be sure to use a waffle feature flag.

Local Branches

Most new code is developed in local one-off branches, usually encompassing one or two patches to fix a bug. Upstream doesn't care how you do local development, but we don't want to see a merge commit every time you merge a single patch from a branch. Merge commits make for a noisy history, which is not fun to look at and can make it difficult to cherry-pick hotfixes to a release branch. We do like to see merge commits when you're committing a set of related patches from a feature branch. The rule of thumb is to rebase and use fast-forward merge for single patches or a branch of unrelated bug fixes, but to use a merge commit if you have multiple commits that form a cohesive unit.

Here are some tips on Using topic branches and interactive rebasing effectively.

1.4.9 Using the VPN with docker on OSX

If you need to access services behind a VPN, the docker setup should by default allow outgoing traffic over the VPN as it does for your host. If this isn't working you might find that it will work if you start up the vm *after* you have started the VPN connection.

To do this simply stop the containers:

```
docker-compose stop
```

Stop the docker-machine vm:

```
# Assumes you've called the vm 'addons-dev'
docker-machine stop addons-dev
```

Then connect to your VPN and restart the docker vm:

```
docker-machine start addons-dev
```

and fire up the env containers again:

```
docker-compose up -d
```

1.4.10 Access Control Lists

ACL versus Django Permissions

Currently we use the `is_superuser` flag in the `User` model to indicate that a user can access the admin site.

Outside of that we use the `access.models.GroupUser` to define what access groups a user is a part of.

How permissions work

Permissions that you can use as filters can be either explicit or general.

For example `Admin:EditAddons` means only someone with that permission will validate.

If you simply require that a user has *some* permission in the *Admin* group you can use `Admin:%`. The `%` means "any."

Similarly a user might be in a group that has explicit or general permissions. They may have `Admin:EditAddons` which means they can see things with that same permission, or things that require `Admin:%`.

If a user has a wildcard, they will have more permissions. For example, `Admin:*` means they have permission to see anything that begins with `Admin:`.

The notion of a superuser has a permission of `*:*` and therefore they can see everything.

1.4.11 Logging

Logging is fun. We all want to be lumberjacks. My muscle-memory wants to put `print` statements everywhere, but it's better to use `log.debug` instead. `print` statements make mod_wsgi sad, and they're not much use in production. Plus, `django-debug-toolbar` can hijack the logger and show all the log statements generated during the last request. When `DEBUG = True`, all logs will be printed to the development console where you started the server. In production, we're piping everything into `mozlog`.

Configuration

The root logger is set up from `settings_base` in the `src/olympia/lib` of addons-server. It sets up sensible defaults, but you can tweak them to your liking:

Log level

There is no unified log level, instead every logger has it's own log level because it depends on the context they're used in.

LOGGING

See PEP 391 for formatting help. Messages will not propagate through a logger unless `propagate: True` is set.

```
LOGGING = {
    'loggers': {
        'caching': {'handlers': ['null']},
    },
}
```

If you want to add more to this do something like this:

```
LOGGING['loggers'].update({
    'z.paypal': {
        'level': logging.DEBUG,
    },
    'z.es': {
        'handlers': ['null'],
    },
})
```

Using Loggers

The `olympia.core.logger` package uses global objects to make the same logging configuration available to all code loaded in the interpreter. Loggers are created in a pseudo-namespace structure, so app-level loggers can inherit settings from a root logger. olympia's root namespace is just `"z"`, in the interest of brevity. In the caching package, we create a logger that inherits the configuration by naming it `"z.caching"`:

```
import olympia.core.logger

log = olympia.core.logger.getLogger('z.caching')

log.debug("I'm in the caching package.")
```

Logs can be nested as much as you want. Maintaining log namespaces is useful because we can turn up the logging output for a particular section of olympia without becoming overwhelmed with logging from all other parts.

olympia.core.logging vs. logging

`olympia.core.logger.getLogger` should be used everywhere. It returns a `LoggingAdapter` that inserts the current user's IP address and username into the log message. For code that lives outside the request-response cycle, it will insert empty values, keeping the message formatting the same.

Complete logging docs: http://docs.python.org/library/logging.html

1.4.12 Services

Services contain a couple of scripts that are run as separate wsgi instances on the services. Usually they are hosted on separate domains. They are stand alone wsgi scripts. The goal is to avoid a whole pile of Django imports, middleware, sessions and so on that we really don't need.

To run the scripts you'll want a wsgi server. You can do this using gunicorn, for example:

```
pip install gunicorn
```

Then you can do:

```
cd services
gunicorn --log-level=DEBUG -c wsgi/receiptverify.py -b 127.0.0.1:9000 --debug␣
↪verify:application
```

To test:

```
curl -d "this is a bogus receipt" http://127.0.0.1:9000/verify/123
```

1.4.13 Translating Fields on Models

The `olympia.translations` app defines a `olympia.translations.models.Translation` model, but for the most part, you shouldn't have to use that directly. When you want to create a foreign key to the `translations` table, use `olympia.translations.fields.TranslatedField`. This subclasses Django's `django.db.models.ForeignKey` to make it work with our special handling of translation rows.

A minimal model with translations in addons-server would look like this:

```
from django.db import models

from olympia.amo.models import ModelBase
from olympia.translations.fields import TranslatedField, save_signal

class MyModel(ModelBase):
    description = TranslatedField()

models.signals.pre_save.connect(save_signal,
                                sender=MyModel,
                                dispatch_uid='mymodel_translations')
```

How it works behind the scenes

As mentioned above, a `TranslatedField` is actually a `ForeignKey` to the `translations` table. However, to support multiple languages, we use a special feature of MySQL allowing you to have a `ForeignKey` pointing to multiple rows.

When querying

Our base manager has a `_with_translations()` method that is automatically called when you instanciate a queryset. It does 2 things:

- Stick an extra lang=lang in the query to prevent query caching from returning objects in the wrong language
- Call `olympia.translations.transformers.get_trans()` which does the black magic.

`get_trans()` is called, and calls in turn `olympia.translations.transformer.build_query()` and builds a custom SQL query. This query is the heart of the magic. For each field, it setups a join on the translations table, trying to find a translation in the current language (using `olympia.translation.get_language()`) and then in the language returned by `get_fallback()` on the instance (for addons, that's `default_locale`; if the `get_fallback()` method doesn't exist, it will use `settings.LANGUAGE_CODE`, which should be `en-US` in addons-server).

Only those 2 languages are considered, and a double join + `IF` / `ELSE` is done every time, for each field.

This query is then ran on the slave (`get_trans()` gets a cursor using `connections[multidb.get_replica()]`) to fetch the translations, and some Translation objects are instantiated from the results and set on the instance(s) of the original query.

To complete the mechanism, `TranslationDescriptor.__get__` returns the `Translation`, and `Translations.__unicode__` returns the translated string as you'd expect, making the whole thing transparent.

When setting

Everytime you set a translated field to a string value, `TranslationDescriptor` `__set__` method is called. It determines which method to call (because you can also assign a dict with multiple translations in multiple languages at the same time). In this case, it calls `translation_from_string()` method, still on the "hidden" `TranslationDescriptor` instance. The current language is passed at this point, using `olympia.translation_utils.get_language()`.

From there, `translation_from_string()` figures out whether it's a new translation of a field we had no translation for, a new translation of a field we already had but in a new language, or an update to an existing translation.

It instantiates a new `Translation` object with the correct values if necessary, or just updates the correct one. It then places that object in a queue of Translation instances to be saved later.

When you eventually call `obj.save()`, the `pre_save` signal is sent. If you followed the example above, that means `olympia.translations.fields.save_signal` is then called, and it unqueues all Translation objects and saves them. It's important to do this on `pre_save` to prevent foreign key constraint errors.

When deleting

Deleting all translations for a field is done using `olympia.translations.models.delete_translation()`. It sets the field to `NULL` and then deletes all the attached translations.

Deleting a *specific* translation (like a translation in spanish, but keeping the english one intact) is implemented but not recommended at the moment. The reason why is twofold:

1. MySQL doesn't let you delete something that still has a FK pointing to it, even if there are other rows that match the FK. When you call `delete()` on a translation, if it was the last translation for that field, we set the FK to `NULL` and delete the translation normally. However, if there were any other translations, instead we temporarily disable the constraints to let you delete just the one you want.
2. Remember how fetching works? If you deleted a translation that is part of the fallback, then when you fetch that object, depending on your locale you'll get an empty string for that field, even if there are `Translation` objects in other languages available!

For additional discussion on this topic, see https://bugzilla.mozilla.org/show_bug.cgi?id=902435

Ordering by a translated field

`olympia.translations.query.order_by_translation` allows you to order a `QuerySet` by a translated field, honoring the current and fallback locales like it's done when querying.

1.4.14 How does search on AMO work?

Note: This is documenting our current state of how search is implemented in addons-server. We will be using this to plan future improvements so please note that we are aware that the process written below is not perfect and has bugs here and there.

Please see https://github.com/orgs/mozilla/projects/17#card-10287357 for more planning.

General structure

Our Elasticsearch cluster contains Add-ons (`addons` index) and statistics data. The purpose of that document is to describe the add-ons part only though. We store two kinds of data for add-ons: indexed fields that are used for search purposes, and non-indexed fields that are meant to be returned (often as-is with no transformations) by the search API (allowing us to return search results data without hitting the database). The latter is not relevant to this document.

Our search can be reached either via the API through */api/v4/addons/search/* or */api/v4/addons/autocomplete/* which are used by our addons-frontend as well as via our legacy pages (which are going away and off-topic here).

Indexing

The key fields we search against are `name`, `summary` and `description`. Because all can be translated, we index twice: - Once with the translation in the language-specific analyzer if supported, under `{field}_l10n_{analyzer}` - Once with the translation in the default locale of the add-on, under `{field}`, analyzed with just the `snowball` analyzer for `description` and `summary`, and a custom analyzer for `name` that applies the following filters: `standard`, `word_delimiter` (a custom version with `preserve_original` set to `true`), `lowercase`, `stop`, and `dictionary_decompounder` (with a specific word list) and `unique`.

In addition, for the name, both fields also contains a subfield called `raw` that holds a non-analyzed variant for exact matches in the corresponding language (stored as a `keyword`, with a `lowercase` normalizer). We also have a `name.trigram` variant for the field in the default language, which is using a custom analyzer that depends on a `ngram` tokenizer (with `min_gram=3`, `max_gram=3` and `token_chars=["letter", "digit"]`)

Flow of a search query through AMO

Let's assume we search on addons-frontend (not legacy) the search query hits the API and gets handled by `AddonSearchView`, which directly queries ElasticSearch and doesn't involve the database at all.

There are a few filters that are described in the */api/v4/addons/search/ docs* but most of them are not very relevant for raw search queries. Examples are filters by guid, platform, category, add-on type or appversion (application version compatibility). Those filters are applied using a `filter` clause and shouldn't affect scoring.

Much more relevant for raw add-on searches (and this is primarily used when you use the search on the frontend) is `SearchQueryFilter`.

It composes various rules to define a more or less usable ranking:

Primary rules

These are the ones using the strongest boosts, so they are only applied to the add-on name.

Applied rules (merged via `should`):

1. A `dis_max` query with `term` matches on `name_l10n_{analyzer}.raw` and `name.raw` if the language of the request matches a known language-specific analyzer, or just a `term` query on `name.raw` (`boost=100.0`) otherwise - our attempt to implement exact matches
2. If we have a matching language-specific analyzer, we add a `match` query to `name_l10n_{analyzer}` (`boost=5.0`, `operator=and`)
3. A `phrase` match on `name` that allows swapped terms (`boost=8.0`, `slop=1`)
4. A `match` on `name`, using the standard text analyzer (`boost=6.0`, `analyzer=standard`, `operator=and`)
5. A `prefix` match on `name` (`boost=3.0`)
6. If a query is < 20 characters long, a `dis_max` query (`boost=4.0`) composed of a fuzzy match on `name` (`boost=4.0`, `prefix_length=2`, `fuzziness=AUTO`, `minimum_should_match=2<2 3<-25%`) and a `match` query on `name.trigram`, with a `minimum_should_match=66%` to avoid noise.

Secondary rules

These are the ones using the weakest boosts, they are applied to fields containing more text like description, summary and tags.

Applied rules (merged via `should`):

1. Look for matches inside the summary (`boost=3.0`, `operator=and`)
2. Look for matches inside the description (`boost=2.0`, `operator=and`)

If the language of the request matches a known language-specific analyzer, those are made using a `multi_match` query using `summary` or `description` and the corresponding `{field}_l10n_{analyzer}`, similar to how exact name matches are performed above, in order to support potential translations.

Rescoring rules

On top of the two sets of rules above, a `rescore` query is applied with a `window_size` of `10`. In production, we have 5 shards, so that should re-adjust the score of the top 50 results returned only. The rules used for rescoring are the same used in the secondary rules above, with just one difference: it's using `match_phrase` instead of `match`, with a slop of `10`.

General query flow:

1. Fetch current translation
2. Fetch locale specific analyzer (List of analyzers)
3. Merge primary and secondary *should* rules
4. Create a `function_score` query that uses a `field_value_factor` function on `average_daily_users` with a `log2p` modifier, as well as a `4.0` weight if the add-on is public & non-experimental.
5. Add the `rescore` query to the mix

1.4.15 Building Docs

To simply build the docs:

```
docker-compose run web make docs
```

If you're working on the docs, use `make loop` to keep your built pages up-to-date:

```
make shell
cd docs
make loop
```

Open `docs/_build/html/index.html` in a web browser.

1.4.16 Add-ons Server Documentation

This is the documentation for the use of the addons-server and its services. All documentation is in plain text files using reStructuredText and Sphinx.

To build the documentation, you need the dependencies from `requirements/docs.txt`. Those are automatically installed together with `make update_deps`, so if you've installed that already (following the *Installing Olympia the long way* page), you're all set.

If you're unsure, activate your `virtualenv` and run:

```
make update_deps
```

The documentation is viewable at http://addons-server.readthedocs.io/, and covers development using Add-ons Server, the source code for Add-ons.

Its source location is in the /docs folder.

Note: this project was once called *olympia*, this documentation often uses that term.

Build the documentation

This is as simple as running:

```
make docs
```

This is the same as `cd`'ing to the `docs` folder, and running `make html` from there.

We include a daemon that can watch and regenerate the built HTML when documentation source files change. To use it, go to the `docs` folder and run:

```
python watcher.py 'make html' $(find . -name '*.rst')
```

Once done, check the result by opening the following file in your browser:

    /path/to/olympia/docs/_build/html/index.html

1.5 Third-Party Usage

Running your own Add-ons server will likely require a few changes. There is currently no easy way to provide custom templates and since Firefox Accounts is used for authentication there is no way to authenticate a user outside of a Mozilla property.

If you would like to run your own Add-ons server you may want to update addons-server to support custom templates and move the Firefox Accounts management to a django authentication backend.

Another option would be to add any APIs that you required and write a custom frontend. This work is already underway and should be completed at some point but help is always welcome. You can find the API work in this project and the frontend work in addons-frontend.

1.6 Basket Synchronisation

This documents what data we synchronize with Basket and how.

1.6.1 Triggers

Every time a field that we're meant to synchronize on an object changes, a full sync of the object is triggered.

A consequence of this is, since the relation between an addon and an user is part of the add-on object, when a new user is added as an author, or an existing author is removed from an add-on, that triggers a full sync of the add-on, including user account information for all its authors.

1.6.2 Objects

We're synchronizing 2 types of objects:

- User Accounts
- Add-ons

User Accounts

Note: Newsletter opt-in information is not stored by AMO, and therefore not synchronized with the rest. It's sent to basket separately directly whenever it changes, through basket's Newsletter API `subscribe` and `unsubscribe` endpoints.

POST https://basket.mozilla.org/amo-sync/userprofile/

Request JSON Object

- **id** (*int*) – The numeric user id.
- **deleted** (*boolean*) – Is the account deleted.
- **display_name** (*string|null*) – The name chosen by the user.
- **homepage** (*string|null*) – The user's website.
- **fxa_id|null** (*string*) – The user FxA Identifier
- **last_login** (*string|null*) – The date of the last successful log in to the website.
- **location** (*string|null*) – The location of the user.

Add-ons

POST https://basket.mozilla.org/amo-sync/addon/

Request JSON Object

- **id** (*int*) – The add-on id on AMO.
- **authors** (*array*) – Array holding information about the authors for the add-on.
- **guid** (*string*) – The add-on extension identifier.
- **name** (*string*) – The add-on name (in add-on's default locale)

- **default_locale** (*string*) – The add-on default locale for translations.
- **slug** (*string*) – The add-on slug.
- **type** (*string*) – The *add-on type*.
- **status** (*string*) – The *add-on status*.
- **is_disabled** (*boolean*) – Whether the add-on is disabled or not.
- **latest_unlisted_version** (*object|null*) – Object holding the latest unlisted *version* of the add-on. Only the `'id`, `compatibility`, `is_strict_compatibility_enabled` and `version` fields are present.
- **current_version** (*object*) – Object holding the current *version* of the add-on. Only the `'id`, `compatibility`, `is_strict_compatibility_enabled` and `version` fields are present.
- **last_updated** (*string*) – The date of the last time the add-on was updated by its developer(s).
- **ratings** (*object*) – Object holding ratings summary information about the add-on.
- **ratings.count** (*int*) – The total number of user ratings for the add-on.
- **ratings.text_count** (*int*) – The number of user ratings with review text for the add-on.
- **ratings.average** (*float*) – The average user rating for the add-on.
- **ratings.bayesian_average** (*float*) – The bayesian average user rating for the add-on.
- **categories** (*object*) – Object holding the categories the add-on belongs to.
- **categories[app_name]** (*array*) – Array holding the *category slugs* the add-on belongs to for a given *add-on application*, referenced by its `app_name`. (Combine with the add-on `type` to determine the name of the category).
- **average_daily_users** (*int*) – The average number of users for the add-on.
- **is_recommended** (*boolean*) – Whether the add-on is recommended by Mozilla or not.

Example data

Here is an example of the full json that would be sent for an add-on:

```
{
    "authors": [
        {
            "id": 11263,
            "deleted": false,
            "display_name": "qwerty",
            "fxa_id": "1209e9bf1eeb59d0934579b6db0ccad1",
            "homepage": "",
            "last_login": "2019-08-06T10:39:44Z",
            "location": ""
        }
    ],
    "average_daily_users": 0,
    "categories": {
        "firefox": [
```

(continues on next page)

(continued from previous page)


```
            "games-entertainment"
        ]
    },
    "current_version": {
        "id": 35900,
        "compatibility": {
            "firefox": {
                "min": "48.0",
                "max": "*"
            }
        },
        "is_strict_compatibility_enabled": false,
        "version": "2.0"
    },
    "default_locale": "en-US",
    "guid": "{85ee4a2a-51b6-4f5e-a99c-6d9abcf6782d}",
    "id": 35896,
    "is_disabled": false,
    "is_recommended": false,
    "last_updated": "2019-06-26T11:38:13Z",
    "latest_unlisted_version": {
        "id": 35899,
        "compatibility": {
            "firefox": {
                "min": "48.0",
                "max": "*"
            }
        },
        "is_strict_compatibility_enabled": false,
        "version": "1.0"
    },
    "name": "Ibird Jelewt Boartrica",
    "ratings": {
        "average": 4.1,
        "bayesian_average": 4.2,
        "count": 43,
        "text_count": 40
    },
    "slug": "ibird-jelewt-boartrica",
    "status": "nominated",
    "type": "extension"
}
```


Here is an example of the full json that would be sent for an user:


```
{
    "id": 11263,
    "deleted": false,
    "display_name": "serses",
    "email": "mozilla@virgule.net",
    "homepage": "",
    "last_login": "2019-08-06T10:39:44Z",
    "location": ""
}
```

1.7 AMO Blocklist

This is a high-level overview of the addons-server implementation of the addons blocklist.

1.7.1 Full-Stack Overview

Firefox determines which add-ons are unsafe to allow to continue to be enabled by checking a blocklist. With v1 and v2 blocklist this is literally a list of addon guids, plus other metadata, that should be blocked from executing (v1 is XML format, v2 is JSON format); with v3 blocklist this is a bloomfilter that is queried - if the addon guid and xpi version is present then it's in the blocklist so should be blocked by Firefox.

The blocklists are all served via Firefox Remote Settings (the current implementation is Kinto):

- the v3 blocklist bloomfilter files are attachments in the records of https://firefox.settings.services.mozilla.com/v1/buckets/blocklists/collections/addons-bloomfilters/records
- the v2 blocklist is the full output of https://firefox.settings.services.mozilla.com/v1/buckets/blocklists/collections/addons/records
- the v1 blocklist is a (server-side) wrapper around the v2 blocklist that rewrites the JSON into XML

Note: v2/v1 are referred to as "legacy" blocklist in these docs.

AMO holds the addon blocklist records and generates the bloomfilters as needed, which are then uploaded to Remote Settings. The records are managed via the admin tools on addons-server, where updates can also be made to the legacy blocklists (*in most cases*).

1.7.2 Admin Tool

Block records aren't created and changed directly via the admin tool; instead *BlockSubmission* records are created that hold details of the submission of (potentially many) blocks that will be created, updated, or deleted. If the add-ons that the Block affects are used by a significant number of users (see *DUAL_SIGNOFF_AVERAGE_DAILY_USERS_THRESHOLD* setting - currently 100k) then the BlockSubmission must be signed off (approved) by another admin user first.

Once the submission is approved - or immediately after saving if the average daily user counts are under the threshold - a task is started to asynchronously create, update, or delete, Block records. If the Block records are marked as legacy then they will be uploaded to the legacy blocklist collection too in this task.

1.7.3 Bloomfilter Generation

Generating a bloomfilter can be quite slow, so a new one is only generated every 6 hours - or less frequently if no Block records have been changed/added/deleted in that time - via a cron job.

An ad-hoc bloomfilter can be created with the *export_blocklist* command but it isn't considered for the cron job (or *stashing*)

Bloomfilter records

A record is created on Remote Settings for each bloomfilter and the filter uploaded as an attachment. The *generation_time* property represents the point in time when all previous addon guid + versions and blocks were used to generate the bloomfilter. An add-on version/file from before this time will definitely be accounted for in the bloomfilter so we can reliably assert if it's blocked or not. An add-on version/file from after this time can't be reliably asserted - there may be false positives or false negatives.

See https://github.com/mozilla/addons-server/issues/13695 and https://github.com/mozilla/addons-server/blob/master/src/olympia/blocklist/cron.py

Bloomfilter record example

```
{
  "attachment": {
    "hash": "37ba24caec49afe6c97c424623e226e31ad052286ffa66d794eb82497dabc279",
    "size": 28561,
    "filename": "filter.bin",
    "location": "staging/addons-bloomfilters/1234567890.bin",
    "mimetype": "application/octet-stream"
  },
  "key_format": "{guid}:{version}",
  "attachment_type": "bloomfilter-base",
  "generation_time": 1587990908999,
}
```

Stashing

Because the bloomfilter files can be quite large "stash" files are also generated, which represent the changes since the previous bloomfilter generation and can be used by Firefox instead to save on bandwidth.

Multiple stashes can be applied by Firefox (in chronological order) to match the state of an up-to-date bloomfilter.

Stash record example

```
{
  "stash": {
    "blocked": [
      "{6f6b1eaa-bb69-4cdb-a24f-1014493d4290}:10.48",
      "kittens@pioneer.mozilla.com:1.2",
      "kittens@pioneer.mozilla.com:1.1",
      "{b01e0601-eddc-4306-886b-8a4fb5c38a1e}:1",
      "{232f11df-20ca-49d4-94eb-e3e63d7ae773}:1.1.2",
      "kittens@pioneer.mozilla.com:1.3",
    ],
    "unblocked": [
      "{896aff0b-d86e-4dd5-9097-5869579b4c28}:1.2",
      "{95ffc924-6ea7-4dfb-8f7b-1dd44f2159d1}:1.22.2"
    ]
  },
  "key_format": "{guid}:{version}",
  "stash_time": 1587990908999,
}
```

The blocked items represent new versions that should be blocked in addition to any matches in the bloomfilter; the unblocked items represent versions that shouldn't be blocked (even though they would match the bloomfilter). *stash_time* is a timestamp that can be relied on to order the stashes.

addons-bloomfilter collection

The collection on Remote Settings at any given point will consist of a single record with *"attachment-type": "bloomfilter-base"*, which is the base bloomfilter to compare the stash files to, and potentially subsequent records which either contain an attachment with *"attachment-type": "bloomfilter-full"*, or stash data directly in the data property. The client-side algorithm would be to:

- Get the entire collection from Remote Settings (the implementation supports diffing so only new records would be downloaded).
- Download the base bloomfilter attachment (*"attachment-type": "bloomfilter-base"*) if it hasn't already been downloaded.
- Gather the stash records and consolidate them, taking into account timestamps so later stashes override earlier stashes.

Stashing support disabled in Firefox

If stashing support is disabled in a Firefox version the stash records can be ignored and all bloomfilters considered instead. (Records with a bloomfilter attachment always have a *generation_time* field). Firefox would just download the latest attachment and use that as it's bloomfilter.

Process

The server process is:

- If the *blocklist_mlbf_submit* waffle switch is enabled, check if there have been any changes to the blocklist since the previous execution of the cron job - if not return without any action. (not blocked guids)
- Produce a list of all "guid:version" combinations of all signed webextension addons/versions in the database. (blocked guids)
- Produce a list of "guid:version" combinations that the Block records cover. Blocks have a minimum and maximum version range - 0 being the minimum, and * meaning infinity, so 0 - * would be all versions of an add-on.
- Create and verify a bloomfilter with these two lists (we use https://github.com/mozilla/filter-cascade/); save the filter file and the two lists (as JSON)
- Compare list of blocked guids from this execution to the base bloomfilter file. If there have been few changes then write those changes to a stash JSON blob
    1. Upload the stash as JSON data in record
    2. Upload the filter as an attachment to a separate record with the type *bloomfilter-full*
- If there have been many changes then:
    1. clear the collection on Remote Settings
    2. Upload the filter as an attachment to a separate record with the type *bloomfilter-base*

1.7.4 Legacy Blocklist

To populate the blocklist on AMO the legacy blocklist on Remote Settings was imported; all guids that matched addons on AMO (and that had at least one webextension version) were added; any guids that were *regular expressions* were "expanded" to individual records for each addon present in the AMO database.

If the *In legacy blocklist* checkbox is selected in AMO's admin tool the block will also be saved to the legacy blocklist. Edits to Blocks will propagate to the legacy blocklist too (apart from *regular expression based blocks*). An existing Block in the legacy blocklist can be removed (while keeping it in the current v3 blocklist) by deselecting the *In legacy blocklist* checkbox; or added to the legacy blocklist too by enabling the *In legacy blocklist* checkbox.

Regex-based Blocks

The legacy blocklist records can contain regular expressions in the *guid* field, which are interpreted by Firefox to match addon guids.

AMO's blocklist implementation, by design, does not generate or change regular expressions in legacy blocklist records. So Blocks imported from a regular expression in the legacy blocklist can be viewed and updated on AMO, and are included in the v3 bloomfilter blocklist, but changes to those records cannot be propagated back to the legacy blocklist because the regular expression would need to be amended.

A warning message is displayed and the user must manually make the changes to the legacy blocklist via the kinto admin tool.

Note: Blocks with a *legacy_id* property starting with * were imported from regular expression based Remote Setting records.

CHAPTER

TWO

ARCHIVED CONTENTS

2.1 External API (V3)

This shows you how to use the addons.mozilla.org API at `/api/v3/` which is hosted at the following URLs:

Environment	URL
Production	`https://addons.mozilla.org/api/v3/`
Staging	`https://addons.allizom.org/api/v3/`
Development	`https://addons-dev.allizom.org/api/v3/`

Production Connect to this API for all normal operation.

Staging or Development Connect to these APIs if you need to work with a scratch database or you're testing features that aren't available in production yet. Your production account is not linked to any of these APIs.

Dive into the *overview section* and the *authentication section* for an example of how to get started using the API.

2.1.1 Overview

This describes the details of the requests and responses you can expect from the addons.mozilla.org API.

Requests

All requests should be made with the header:

```
Content-type: application/json
```

Responses

Status Codes

There are some common API responses that you can expect to receive at times.

`GET /api/v3/...`

Status Codes

- 200 OK – Success.
- 201 Created – Creation successful.

- 202 Accepted – The request has been accepted for processing. This usually means one or more asyncronous tasks is being executed in the background so results aren't immediately visible.
- 204 No Content – Success (no content is returned).
- 400 Bad Request – There was a problem with the parameters sent with this request.
- 401 Unauthorized – Authentication is required or failed.
- 403 Forbidden – You are not permitted to perform this action.
- 404 Not Found – The requested resource could not be found.
- 500 Internal Server Error – An unknown error occurred.
- 503 Service Unavailable – The site is in maintenance mode at this current time and the operation can not be performed.

Bad Requests

When returning a `HTTP 400 Bad Request` response, the API will try to return some information about the error(s) in the body of the response, as a JSON object. The keys of that object indicate the field(s) that caused an error, and for each, a list of messages will be provided (often only one message will be present, but sometimes more). If the error is not attached to a specific field the key `non_field_errors` will be used instead.

Example:

```
{
    "username": ["This field is required."],
    "non_field_errors": ["Error not linked to a specific field."]
}
```

Unauthorized and Permission Denied

When returning `HTTP 401 Unauthorized` and `HTTP 403 Permission Denied` responses, the API will try to return some information about the error in the body of the response, as a JSON object. A `detail` property will contain a message explaining the error. In addition, in some cases, an optional `code` property will be present and will contain a constant corresponding to specific problems to help clients address the situation programmatically. The constants are as follows:

Value	Description
ERROR_INVALID_HEADER	The `Authorization` header is invalid.
ERROR_SIGNATURE_EXPIRED	The signature of the token indicates it has expired.
ERROR_DECODING_SIGNATURE	The token was impossible to decode and probably invalid.

Pagination

By default, all endpoints returning a list of results are paginated. The default number of items per page is 25 and clients can use the *page_size* query parameter to change it to any value between 1 and 50. Exceptions to those rules are possible but will be noted in the corresponding documentation for affected endpoints.

The following properties will be available in paginated responses:

- *next*: the URL for the next page in the pagination.
- *previous*: the URL for the previous page in the pagination.
- *page_size*: The number of items per page in the pagination.
- *page_count*: The number of pages available in the pagination. It may be lower than *count / page_size* for elasticsearch based paginations that go beyond our *max_result_window* configuration.
- *count*: the total number of records.
- *results*: the array containing the results for this page.

Translated Fields

Fields that can be translated by users (typically name, description) have a special behaviour. The default is to return them as an object, with languages as keys and translations as values:

```
{
    "name": {
        "en-US": "Games",
        "fr": "Jeux",
        "kn": ""
    }
}
```

However, for performance, if you pass the `lang` parameter to a `GET` request, then only the most relevant translation (the specified language or the fallback, depending on whether a translation is available in the requested language) will be returned as a string.

```
{
    "name": "Games"
}
```

This behaviour also applies to `POST`, `PATCH` and `PUT` requests: you can either submit an object containing several translations, or just a string. If only a string is supplied, it will only be used to translate the field in the current language.

Outgoing Links

If the `wrap_outgoing_links` query parameter is present, any external links returned for properties such as `support_url` or `homepage` will be wrapped through `outgoing.prod.mozaws.net`. Fields supporting some HTML, such as add-on `description`, always do this regardless of whether or not the query parameter is present.

Cross Origin

All APIs are available with Cross-Origin Resource Sharing unless otherwise specified.

2.1.2 Authentication (External)

To access the API as an external consumer, you need to include a JSON Web Token (JWT) in the `Authorization` header for every request. This header acts as a one-time token that authenticates your user account. No JWT claims are made about the actual API request you are making.

If you are building an app that lives on the AMO domain, read the *documentation for internal authentication* instead.

Access Credentials

To create JWTs, first obtain a **key** and **secret** from the API Credentials Management Page.

Note: Keep your API keys secret and *never* commit them to a public code repository or share them with anyone, including Mozilla contributors.

If someone obtains your secret they can make API requests on behalf of your user account.

Create a JWT for each request

Prior to making every API request, you need to generate a fresh JWT. The JWT will have a short expiration time and is only valid for a single request so you can't cache or reuse it. You only need to include a few standard fields; here's what the raw JSON object needs to look like before it's signed:

```
{
    "iss": "your-api-key",
    "jti": "0.47362944623455405",
    "iat": 1447273096,
    "exp": 1447273156
}
```

iss This is a standard JWT claim identifying the *issuer*. Set this to the **API key** you generated on the credentials management page. For example: `user:543210:23`.

jti This is a standard JWT claim declaring a *JWT ID*. This value needs to have a high probability of being unique across all recent requests made by your issuer ID. This value is a type of cryptographic nonce designed to prevent replay attacks.

iat This is a standard JWT claim indicating the *issued at time*. It should be a Unix epoch timestamp and **must be in UTC time**.

exp This is a standard JWT claim indicating the *expiration time*. It should be a Unix epoch timestamp in UTC time and must be **no longer than five minutes** past the issued at time.

> Changed in version 2016-10-06: We increased the expiration time from 60 seconds to five minutes to workaround support for large and slow uploads.

Note: If you're having trouble authenticating, make sure your system clock is correct and consider synchronizing it with something like tlsdate.

Take this JSON object and sign it with the **API secret** you generated on the credentials management page. You must sign the JWT using the `HMAC-SHA256` algorithm (which is typically the default). The final JWT will be a blob of base64 encoded text, something like:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↪eyJpc3MiOiJ5b3VyLWFwaS1rZXkiLCJqdGkiOiIwLjQ3MzYyOTQ0NjIzNDU1NDA1IiwiaWF0IjoxNDQ3MjczMDk2LCJleHAiOjE
↪fQGPSV85QPhbNmuu86CIgZiluKBvZKd-NmzM6vo11D
```

Note: See jwt.io debugger for more information about the token.

Here is an example of creating a JWT in NodeJS using the node-jsonwebtoken library:

```
var jwt = require('jsonwebtoken');

var issuedAt = Math.floor(Date.now() / 1000);
var payload = {
  iss: 'your-api-key',
  jti: Math.random().toString(),
  iat: issuedAt,
  exp: issuedAt + 60,
};

var secret = 'your-api-secret';  // store this securely.
var token = jwt.sign(payload, secret, {
  algorithm: 'HS256',  // HMAC-SHA256 signing algorithm
});
```

Create an Authorization header

When making each request, put your generated JSON Web Token (JWT) into an HTTP Authorization header prefixed with `JWT`, like this:

```
Authorization: JWT eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↪eyJpc3MiOiJ5b3VyLWFwaS1rZXkiLCJqdGkiOiIwLjQ3MzYyOTQ0NjIzNDU1NDA1IiwiaWF0IjoxNDQ3MjczMDk2LCJleHAiOjE
↪fQGPSV85QPhbNmuu86CIgZiluKBvZKd-NmzM6vo11DM
```

Example request

Using the *profile* as an example endpoint, here's what a JWT authenticated HTTP request would look like in curl:

```
curl "https://addons.mozilla.org/api/v3/accounts/profile/" \
     -H "Authorization: JWT eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↪eyJpc3MiOiJ5b3VyLWFwaS1rZXkiLCJqdGkiOiIwLjQ3MzYyOTQ0NjIzNDU1NDA1IiwiaWF0IjoxNDQ3MjczMDk2LCJleHAiOjE
↪fQGPSV85QPhbNmuu86CIgZiluKBvZKd-NmzM6vo11DM"
```

Find a JWT library

There are robust open source libraries for creating JWTs in all major programming languages.

2.1.3 Authentication (internal)

This documents how to use authentication in your API requests when you are working on a web application that lives on AMO domain or subdomain. If you are looking for how to authenticate with the API from an external client, using your API keys, read the *documentation for external authentication* instead.

When using this authentication mechanism, the server is responsible for creating an API Token when the user logs in, and sends it back in the response. The clients must then include that token as an `Authorization` header on requests that need authentication. The clients never generate JWTs themselves.

Fetching the token

A fresh token, valid for 30 days, is automatically generated and added to the responses of the following endpoint:

- `/api/v3/accounts/authenticate/`

The token is available in two forms:

- For the endpoint mentioned above, as a property called `token`.
- For all endpoints, as a cookie called `frontend_auth_token`. This cookie expires after 30 days and is set as `HttpOnly`.

Creating an Authorization header

When making an authenticated API request, put your generated API Token into an HTTP Authorization header prefixed with `Bearer`, like this:

```
Authorization: Bearer␣
↪eyJhdXRoX2hhc2giOiJiY2E0MTZkN2RiMGU3NjFmYTA2NDE4MjAzZWU1NTMwOTM4OGZhNzcxIiwidXNlcl9pZCI6MTIzNDV9:1
↪gNo3EWU8IfL8
```

2.1.4 Abuse Reports

The following API endpoint covers abuse reporting

Submitting an add-on abuse report

The following API endpoint allows an abuse report to be submitted for an Add-on, either listed on https://addons.mozilla.org or not. Authentication is not required, but is recommended so reports can be responded to if nessecary.

`POST /api/v3/abuse/report/addon/`

Request JSON Object

- **`addon`** (*`string`*) – The id, slug, or guid of the add-on to report for abuse (required).
- **`message`** (*`string`*) – The body/content of the abuse report (required).

Response JSON Object

- **reporter** (*object|null*) – The user who submitted the report, if authenticated.
- **reporter.id** (*int*) – The id of the user who submitted the report.
- **reporter.name** (*string*) – The name of the user who submitted the report.
- **reporter.username** (*string*) – The username of the user who submitted the report.
- **reporter.url** (*string*) – The link to the profile page for of the user who submitted the report.
- **addon** (*object*) – The add-on reported for abuse.
- **addon.guid** (*string*) – The add-on extension identifier.
- **addon.id** (*int|null*) – The add-on id on AMO. If the guid submitted didn't match a known add-on on AMO, then null.
- **addon.slug** (*string|null*) – The add-on slug. If the guid submitted didn't match a known add-on on AMO, then null.
- **message** (*string*) – The body/content of the abuse report.

Submitting a user abuse report

The following API endpoint allows an abuse report to be submitted for a user account on https://addons.mozilla.org. Authentication is not required, but is recommended so reports can be responded to if nessecary.

POST /api/v3/abuse/report/user/

Request JSON Object

- **user** (*string*) – The id or username of the user to report for abuse (required).
- **message** (*string*) – The body/content of the abuse report (required).

Response JSON Object

- **reporter** (*object|null*) – The user who submitted the report, if authenticated.
- **reporter.id** (*int*) – The id of the user who submitted the report.
- **reporter.name** (*string*) – The name of the user who submitted the report.
- **reporter.url** (*string*) – The link to the profile page for of the user who submitted the report.
- **reporter.username** (*string*) – The username of the user who submitted the report.
- **user** (*object*) – The user reported for abuse.
- **user.id** (*int*) – The id of the user reported.
- **user.name** (*string*) – The name of the user reported.
- **user.url** (*string*) – The link to the profile page for of the user reported.
- **user.username** (*string*) – The username of the user reported.
- **message** (*string*) – The body/content of the abuse report.

2.1.5 Accounts

The following API endpoints cover a users account.

Account

This endpoint returns information about a user's account, by the account id. Only *developer* accounts are publicly viewable - other user's accounts will return a 404 not found response code. Most of the information is optional and provided by the user so may be missing or inaccurate.

A developer is defined as a user who is listed as a developer or owner of one or more approved add-ons.

GET /api/v3/accounts/account/(int:user_id|string:username)/

Response JSON Object

- **average_addon_rating** (*float*) – The average rating for addons the developer has listed on the website.
- **biography** (*string|null*) – More details about the user.
- **created** (*string*) – The date when this user first logged in and created this account.
- **has_anonymous_display_name** (*boolean*) – The user has chosen neither a name nor a username.
- **has_anonymous_username** (*boolean*) – The user hasn't chosen a username.
- **homepage** (*string|null*) – The user's website.
- **id** (*int*) – The numeric user id.
- **is_addon_developer** (*boolean*) – The user has developed and listed add-ons on this website.
- **is_artist** (*boolean*) – The user has developed and listed themes on this website.
- **location** (*string|null*) – The location of the user.
- **name** (*string*) – The name chosen by the user, or the username if not set.
- **num_addons_listed** (*int*) – The number of addons the developer has listed on the website.
- **occupation** (*string|null*) – The occupation of the user.
- **picture_type** (*string|null*) – the image type (only 'image/png' is supported) if a user photo has been uploaded, or null otherwise.
- **picture_url** (*string|null*) – URL to a photo of the user, or null if no photo has been uploaded.
- **username** (*string*) – username chosen by the user, used in the account url. If not set will be a randomly generated string.

If you authenticate and access your own account by specifing your own `user_id` the following additional fields are returned. You can always access your account, regardless of whether you are a developer or not. If you have *Users:Edit* permission you will see these extra fields for all user accounts.

GET /api/v3/accounts/account/(int:user_id|string:username)/

Response JSON Object

- **deleted** (*boolean*) – Is the account deleted.
- **display_name** (*string|null*) – The name chosen by the user.
- **email** (*string*) – Email address used by the user to login and create this account.
- **last_login** (*string*) – The date of the last successful log in to the website.
- **last_login_ip** (*string*) – The IP address of the last successfull log in to the website.
- **is_verified** (*boolean*) – The user has been verified via FirefoxAccounts.
- **permissions** (*array*) – A list of the additional *permissions* this user has.
- **read_dev_agreement** (*boolean*) – The user has read, and agreed to, the developer agreement that is required to submit addons.

Status Codes

- 200 OK – account found.
- 400 Bad Request – an error occurred, check the `error` value in the JSON.
- 404 Not Found – no account with that user id.

Important:

- `Biography` can contain HTML, or other unsanitized content, and it is the responsibiliy of the client to clean and escape it appropriately before display.

Permissions can be any arbritary string in the format *app:action*. Either *app* or *action* can be the wildcard *, so **:** means the user has permission to do all actions (i.e. full admin).

The following are some commonly tested permissions; see https://github.com/mozilla/addons-server/blob/master/src/olympia/constants/permissions.py for the full list.

Value	Description
AdminTools:View	Can access the website admin interface index page. Inner pages may require other/additional permissions.
Addons:Edit	Allows viewing and editing of any add-ons details in developer tools.
Addons:Review	Can access the add-on reviewer tools to approve/reject add-on submissions.
Personas:Review	Can access the theme reviewer tools to approve/reject theme submissions.

Profile

Note: This API requires *authentication*.

This endpoint is a shortcut to your own account. It returns an *account object*

GET /api/v3/accounts/profile/

Edit

Note: This API requires *authentication* and *Users:Edit* permission to edit accounts other than your own.

This endpoint allows some of the details for an account to be updated. Any fields in the *account* (or *self*) but not listed below are not editable and will be ignored in the patch request.

PATCH /api/v3/accounts/account/(int:user_id|string:username)/

Request JSON Object

- **biography** (*string|null*) – More details about the user. No links are allowed.
- **display_name** (*string|null*) – The name chosen by the user.
- **homepage** (*string|null*) – The user's website.
- **location** (*string|null*) – The location of the user.
- **occupation** (*string|null*) – The occupation of the user.
- **username** (*string|null*) – username to be used in the account url. The username can only contain letters, numbers, underscores or hyphens. All-number usernames are prohibited as they conflict with user-ids.

Uploading a picture

To upload a picture for the profile the request must be sent as content-type *multipart/form-data* instead of JSON. Images must be either PNG or JPG; the maximum file size is 4MB. Other *editable values* can be set at the same time.

PATCH /api/v3/accounts/account/(int:user_id|string:username)/

Request:

```
curl "https://addons.mozilla.org/api/v3/accounts/account/12345/"
    -g -XPATCH --form "picture_upload=@photo.png"
    -H "Authorization: Bearer <token>"
```

Parameters

- **user-id** – The numeric user id.

Form Parameters

- **picture_upload** – The user's picture to upload.

Request Headers

- Content-Type – multipart/form-data

Deleting the picture

To delete the account profile picture call the special endpoint.

DELETE /api/v3/accounts/account/(int:user_id|string:username)/picture

Delete

Note: This API requires *authentication* and *Users:Edit* permission to delete accounts other than your own.

Note: Accounts of users who are authors of Add-ons can't be deleted. All Add-ons (and Themes) must be deleted or transfered to other users first.

This endpoint allows the account to be deleted. The reviews and ratings created by the user will not be deleted; but all the user's details are cleared.

DELETE /api/v3/accounts/account/(int:user_id|string:username)/

Notifications List

Note: This API requires *authentication* and *Users:Edit* permission to list notifications on accounts other than your own.

This endpoint allows you to list the account notifications set for the specified user. The result is an unpaginated list of the fields below. There are currently 11 notification types.

GET /api/v3/accounts/account/(int:user_id|string:username)/notifications/

Response JSON Object

- **name** (*string*) – The notification short name.
- **enabled** (*boolean*) – If the notification is enabled (defaults to True).
- **mandatory** (*boolean*) – If the notification can be set by the user.

Notifications Update

Note: This API requires *authentication* and *Users:Edit* permission to set notification preferences on accounts other than your own.

This endpoint allows account notifications to be set or updated. The request should be a dict of *name*:TruelFalse pairs. Any number of notifications can be changed; only non-mandatory notifications can be changed - attempting to set a mandatory notification will return an error.

POST /api/v3/accounts/account/(int:user_id|string:username)/notifications/

Request JSON Object

- **<name>** (*boolean*) – Is the notification enabled?

Super-creation

Note: This API requires *authentication*.

This allows you to generate a new user account and sign in as that user.

Important:

- Your API user must be in the `Accounts:SuperCreate` group to access this endpoint. Use `manage.py createsuperuser --add-to-supercreate-group` to create a superuser with proper access.
- This endpoint is not available in all *API environments*.

`POST /api/v3/accounts/super-create/`

Request:

Parameters

- **`email`** – assign the user a specific email address. A fake email will be assigned by default.
- **`username`** – assign the user a specific username. A random username will be assigned by default.
- **`fxa_id`** – assign the user a Firefox Accounts ID, like one returned in the `uuid` parameter of a profile request. This is empty by default, meaning the user's account will need to be migrated to a Firefox Account.
- **`group`** – assign the user to a permission group. Valid choices:
  - **reviewer**: can access add-on reviewer pages, formerly known as Editor Tools
  - **admin**: can access any protected page

```
curl "https://addons.mozilla.org/api/v3/accounts/super-create/" \
    -X POST -H "Authorization: JWT <jwt-token>"
```

Response:

```
{
    "username": "super-created-7ee304ce",
    "display_name": "Super Created 7ee304ce",
    "user_id": 10985,
    "email": "super-created-7ee304ce@addons.mozilla.org",
    "fxa_id": null,
    "groups": [],
    "session_cookie": {
        "encoded": "sessionid=.eJyrVopPLC3JiC8tTi2KT...",
        "name": "sessionid",
        "value": ".eJyrVopPLC3JiC8tTi2KT..."
    }
}
```

Status Codes

- 201 Created – Account created.
- 422 Unprocessable Entity – Incorrect request parameters.

The session cookie will enable you to sign in for a limited time as this new user. You can pass it to any login-protected view like this:

```
curl --cookie sessionid=... -s -D - \
    "https://addons.mozilla.org/en-US/developers/addon/submit/1" \
    -o /dev/null
```

Session

Log out of the current session. This is for use with the *internal authentication* that authenticates browser sessions.

DELETE /api/v3/accounts/session/

Request:

```
curl "https://addons.mozilla.org/api/v3/accounts/session/"
    -H "Authorization: Bearer <jwt-token>" -X DELETE
```

Response:

```
{
    "ok": true
}
```

Status Codes

- 200 OK – session logged out.
- 401 Unauthorized – authentication failed.

2.1.6 Activity

Note: These APIs are experimental and are currently being worked on. Endpoints may change without warning. The only authentication method available at the moment is *the internal one*.

Review Notes List

This endpoint allows you to list the approval/rejection review history for a version of an add-on.

GET /api/v3/addons/addon/(int:addon_id|string:addon_slug|string:addon_guid)/versions/(int: *id*) **/ revie**

Note: All add-ons require authentication and either reviewer permissions or a user account listed as a developer of the add-on.

Response JSON Object

- **count** (*int*) – The number of versions for this add-on.
- **next** (*string*) – The URL of the next page of results.

- **previous** (*string*) – The URL of the previous page of results.
- **results** (*array*) – An array of *per version review notes*.

Review Notes Detail

This endpoint allows you to fetch a single review note for a specific version of an add-on.

GET /api/v3/addons/addon/(int:addon_id|string:addon_slug|string:addon_guid)/versions/(int:id)/revie

int: *id*/

Response JSON Object

- **id** (*int*) – The id for a review note.
- **action** (*string*) – The *type of review note*.
- **action_label** (*string*) – The text label of the action.
- **user.id** (*int*) – The id of the reviewer or author who left the review note.
- **user.name** (*string*) – The name of the reviewer or author.
- **user.url** (*string*) – The link to the profile page for of the reviewer or author.
- **user.username** (*string*) – The username of the reviewer or author.
- **comments** (*string*) – The text content of the review note.
- **date** (*string*) – The date the review note was created.

Possible values for the `action` field:

Value	Description
approved	Version, or file in the version, was approved
rejected	Version, or file in the version, was rejected
review-requested	Developer requested review
more-information-requested	Reviewer requested more information from developer
super-review-requested	Add-on was referred to an admin for attention
comment	Reviewer added comment for other reviewers
review-note	Generic review comment

Incoming Mail End-point

This endpoint allows a mail server or similar to submit a json object containing single email into AMO which will be processed. The only type of email currently supported is a reply to an activity email (e.g an add-on review, or a reply to an add-on review). Any other content or invalid emails will be discarded.

POST /api/v3/activity/mail

Note: This API endpoint uses a custom authentication method. The value *SecretKey* in the submitted json must match one defined in *settings.INBOUND_EMAIL_SECRET_KEY*. The IP address of the request must match one defined in *settings.ALLOWED_CLIENTS_EMAIL_API*, if defined.

Request JSON Object

- **SecretKey** (*string*) – A value that matches *settings.INBOUND_EMAIL_SECRET_KEY*.
- **Message.TextBody** (*string*) – The plain text body of the email.
- **To** (*array*) – Array of To email addresses. All will be parsed, and the first matching the correct format used.
- **To[].EmailAddress** (*string*) – An email address in the format *reviewreply+randomuuidstring@addons.mozilla.org*.

2.1.7 Add-ons

Note: These APIs are experimental and are currently being worked on. Endpoints may change without warning. The only authentication method available at the moment is *the internal one*.

Featured

This endpoint allows you to list featured add-ons matching some parameters. Results are sorted randomly and therefore, the standard pagination parameters are not accepted. The query parameter `page_size` is allowed but only serves to customize the number of results returned, clients can not request a specific page.

GET /api/v3/addons/featured/

Query Parameters

- **app** (*string*) – **Required**. Filter by *add-on application* availability.
- **category** (*string*) – Filter by *category slug*. `app` and `type` parameters need to be set, otherwise this parameter is ignored.
- **lang** (*string*) – Request add-ons featured for this specific language to be returned alongside add-ons featured globally. Also activate translations for that query. (See *translated fields*)
- **type** (*string*) – Filter by *add-on type*.
- **page_size** (*int*) – Maximum number of results to return. Defaults to 25.

Response JSON Object

- **results** (*array*) – An array of *add-ons*.

Search

This endpoint allows you to search through public add-ons.

GET /api/v3/addons/search/

Query Parameters

- **q** (*string*) – The search query. The maximum length allowed is 100 characters.
- **app** (*string*) – Filter by *add-on application* availability.
- **appversion** (*string*) – Filter by application version compatibility. Pass the full version as a string, e.g. `46.0`. Only valid when the `app` parameter is also present.

- **author** (*string*) – Filter by exact author username. Multiple author names can be specified, separated by comma(s), in which case add-ons with at least one matching author are returned.
- **category** (*string*) – Filter by *category slug*. `app` and `type` parameters need to be set, otherwise this parameter is ignored.
- **exclude_addons** (*string*) – Exclude add-ons by `slug` or `id`. Multiple add-ons can be specified, separated by comma(s).
- **featured** (*boolean*) – Filter to only featured add-ons. Only `featured=true` is supported. If `app` is provided as a parameter then only featured collections targeted to that application are taken into account. If `lang` is provided then only featured collections targeted to that language, (and collections for all languages), are taken into account. Both `app` and `lang` can be provided to filter to addons that are featured in collections that application and for that language, (and for all languages).
- **guid** (*string*) – Filter by exact add-on guid. Multiple guids can be specified, separated by comma(s), in which case any add-ons matching any of the guids will be returned. As guids are unique there should be at most one add-on result per guid specified.
- **lang** (*string*) – Activate translations in the specific language for that query. (See *translated fields*)
- **page** (*int*) – 1-based page number. Defaults to 1.
- **page_size** (*int*) – Maximum number of results to return for the requested page. Defaults to 25.
- **platform** (*string*) – Filter by *add-on platform* availability.
- **tag** (*string*) – Filter by exact tag name. Multiple tag names can be specified, separated by comma(s), in which case add-ons containing *all* specified tags are returned.
- **type** (*string*) – Filter by *add-on type*.
- **sort** (*string*) – The sort parameter. The available parameters are documented in the *table below*.

Response JSON Object

- **count** (*int*) – The number of results for this query.
- **next** (*string*) – The URL of the next page of results.
- **previous** (*string*) – The URL of the previous page of results.
- **results** (*array*) – An array of *add-ons*. As described below, the following fields are omitted for performance reasons: `release_notes` and `license` fields on `current_version` as well as `picture_url` from `authors`.

Available sorting parameters:

Parameter	Description
created	Creation date, descending.
downloads	Number of weekly downloads, descending.
hotness	Hotness (average number of users progression), descending.
random	Random ordering. Only available when no search query is passed and when filtering to only return featured add-ons.
rating	Bayesian rating, descending.
relevance	Search query relevance, descending.
updated	Last updated date, descending.
users	Average number of daily users, descending.

The default is to sort by relevance if a search query (`q`) is present, otherwise sort by number of weekly downloads, descending.

You can combine multiple parameters by separating them with a comma. For instance, to sort search results by downloads and then by creation date, use `sort=downloads,created`. The only exception is the `random` sort parameter, which is only available alone.

Autocomplete

Similar to *add-ons search endpoint* above, this endpoint allows you to search through public add-ons. Because it's meant as a backend for autocomplete though, there are a couple key differences:

- No pagination is supported. There are no `next`, `prev` or `count` fields, and passing `page_size` or `page` has no effect, a maximum of 10 results will be returned at all times.
- Only a subset of fields are returned.

`GET /api/v3/addons/autocomplete/`

Query Parameters

- **`q`** (*`string`*) – The search query.
- **`app`** (*`string`*) – Filter by *add-on application* availability.
- **`appversion`** (*`string`*) – Filter by application version compatibility. Pass the full version as a string, e.g. `46.0`. Only valid when the `app` parameter is also present.
- **`author`** (*`string`*) – Filter by exact author username.
- **`category`** (*`string`*) – Filter by *category slug*. `app` and `type` parameters need to be set, otherwise this parameter is ignored.
- **`lang`** (*`string`*) – Activate translations in the specific language for that query. (See *translated fields*)
- **`platform`** (*`string`*) – Filter by *add-on platform* availability.
- **`tag`** (*`string`*) – Filter by exact tag name. Multiple tag names can be specified, separated by comma(s).
- **`type`** (*`string`*) – Filter by *add-on type*.
- **`sort`** (*`string`*) – The sort parameter. The available parameters are documented in the *table below*.

Response JSON Object

- **results** (*array*) – An array of *add-ons*. Only the `id`, `icon_url`, `name` and `url` fields are supported though.

Detail

This endpoint allows you to fetch a specific add-on by id, slug or guid.

Note: Non-public add-ons and add-ons with only unlisted versions require both authentication and reviewer permissions or an account listed as a developer of the add-on.

A 401 or 403 error response will be returned when clients don't meet those requirements. Those responses will contain the following properties:

- `detail`: string containing a message about the error.
- `is_disabled_by_developer`: boolean set to `true` when the add-on has been voluntarily disabled by its developer.
- `is_disabled_by_mozilla`: boolean set to `true` when the add-on has been disabled by Mozilla.

GET /api/v3/addons/addon/(int:id|string:slug|string:guid)/

Query Parameters

- **lang** (*string*) – Activate translations in the specific language for that query. (See *Translated Fields*)
- **wrap_outgoing_links** (*string*) – If this parameter is present, wrap outgoing links through `outgoing.prod.mozaws.net` (See *Outgoing Links*)

Response JSON Object

- **id** (*int*) – The add-on id on AMO.
- **authors** (*array*) – Array holding information about the authors for the add-on.
- **authors[].id** (*int*) – The id for an author.
- **authors[].name** (*string*) – The name for an author.
- **authors[].url** (*string*) – The link to the profile page for an author.
- **authors[].username** (*string*) – The username for an author.
- **authors[].picture_url** (*string*) – URL to a photo of the user, or */static/img/anon_user.png* if not set. For performance reasons this field is omitted from the search endpoint.
- **average_daily_users** (*int*) – The average number of users for the add-on (updated daily).
- **categories** (*object*) – Object holding the categories the add-on belongs to.
- **categories[app_name]** (*array*) – Array holding the *category slugs* the add-on belongs to for a given *add-on application*. (Combine with the add-on `type` to determine the name of the category).

- **contributions_url** (*string|null*) – URL to the (external) webpage where the addon's authors collect monetary contributions, if set.
- **current_version** (*object*) – Object holding the current *version* of the add-on. For performance reasons the `license` field omits the `text` property from the detail endpoint. In addition, `license` and `release_notes` are omitted entirely from the search endpoint.
- **default_locale** (*string*) – The add-on default locale for translations.
- **description** (*string|object|null*) – The add-on description (See *translated fields*).
- **developer comments** (*string|object|null*) – Additional information about the add-on provided by the developer. (See *translated fields*).
- **edit_url** (*string*) – The URL to the developer edit page for the add-on.
- **guid** (*string*) – The add-on extension identifier.
- **has_eula** (*boolean*) – The add-on has an End-User License Agreement that the user needs to agree with before installing (See *add-on EULA and privacy policy*).
- **has_privacy_policy** (*boolean*) – The add-on has a Privacy Policy (See *add-on EULA and privacy policy*).
- **homepage** (*string|object|null*) – The add-on homepage (See *translated fields*).
- **icon_url** (*string*) – The URL to icon for the add-on (including a cachebusting query string).
- **icons** (*object*) – An object holding the URLs to an add-ons icon including a cachebusting query string as values and their size as properties. Currently exposes 32 and 64 pixels wide icons.
- **is_disabled** (*boolean*) – Whether the add-on is disabled or not.
- **is_experimental** (*boolean*) – Whether the add-on has been marked by the developer as experimental or not.
- **is_featured** (*boolean*) – The add-on appears in a featured collection.
- **is_source_public** (*boolean*) – Whether the add-on source is publicly viewable or not.
- **name** (*string|object|null*) – The add-on name (See *translated fields*).
- **last_updated** (*string*) – The date of the last time the add-on was updated by its developer(s).
- **latest_unlisted_version** (*object|null*) – Object holding the latest unlisted *version* of the add-on. This field is only present if the user has unlisted reviewer permissions, or is listed as a developer of the add-on.
- **previews** (*array*) – Array holding information about the previews for the add-on.
- **previews[].id** (*int*) – The id for a preview.
- **previews[].caption** (*string|object|null*) – The caption describing a preview (See *translated fields*).
- **previews[].image_size[]** (*int*) – width, height dimensions of of the preview image.

- **previews[].image_url** (*string*) – The URL (including a cachebusting query string) to the preview image.
- **previews[].thumbnail_size[]** (*int*) – width, height dimensions of of the preview image thumbnail.
- **previews[].thumbnail_url** (*string*) – The URL (including a cachebusting query string) to the preview image thumbnail.
- **public_stats** (*boolean*) – Boolean indicating whether the add-on stats are public or not.
- **ratings** (*object*) – Object holding ratings summary information about the add-on.
- **ratings.count** (*int*) – The total number of user ratings for the add-on.
- **ratings.text_count** (*int*) – The number of user ratings with review text for the add-on.
- **ratings_url** (*string*) – The URL to the user ratings list page for the add-on.
- **ratings.average** (*float*) – The average user rating for the add-on.
- **ratings.bayesian_average** (*float*) – The bayesian average user rating for the add-on.
- **requires_payment** (*boolean*) – Does the add-on require payment, non-free services or software, or additional hardware.
- **review_url** (*string*) – The URL to the reviewer review page for the add-on.
- **slug** (*string*) – The add-on slug.
- **status** (*string*) – The *add-on status*.
- **summary** (*string|object|null*) – The add-on summary (See *translated fields*).
- **support_email** (*string|object|null*) – The add-on support email (See *translated fields*).
- **support_url** (*string|object|null*) – The add-on support URL (See *translated fields*).
- **tags** (*array*) – List containing the text of the tags set on the add-on.
- **theme_data** (*object*) – Object holding lightweight theme (Persona) data. Only present for themes (Persona).
- **type** (*string*) – The *add-on type*.
- **url** (*string*) – The (absolute) add-on detail URL.
- **weekly_downloads** (*int*) – The number of downloads for the add-on in the last week. Not present for lightweight themes.

Possible values for the `status` field / parameter:

Value	Description
lite	Preliminarily Reviewed
public	Fully Reviewed
deleted	Deleted
pending	Pending approval (Valid for themes only)
disabled	Disabled by Mozilla
rejected	Rejected (Valid for themes only)
nominated	Awaiting Full Review
incomplete	Incomplete
unreviewed	Awaiting Preliminary Review
lite-nominated	Preliminarily Reviewed and Awaiting Full Review
review-pending	Flagged for further review (Valid for themes only)

Possible values for the keys in the `compatibility` field, as well as the `app` parameter in the search API:

Value	Description
android	Firefox for Android
firefox	Firefox
seamonkey	SeaMonkey
thunderbird	Thunderbird

Note: For possible version values per application, see valid application versions.

Possible values for the `current_version.files[].platform` field:

Value	Description
all	All
mac	Mac
linux	Linux
android	Android
windows	Windows

Possible values for the `type` field / parameter:

Note: For backwards-compatibility reasons, the value for Theme is `persona`. `theme` refers to a Complete Theme.

Value	Description
theme	Complete Theme
search	Search Engine
persona	Theme
language	Language Pack (Application)
extension	Extension
dictionary	Dictionary

Add-on and Version Submission

See *Uploading a version*.

Versions List

This endpoint allows you to list all versions belonging to a specific add-on.

GET /api/v3/addons/addon/(int:addon_id|string:addon_slug|string:addon_guid)/versions/

Note: Non-public add-ons and add-ons with only unlisted versions require both:

- authentication
- reviewer permissions or an account listed as a developer of the add-on

---

Query Parameters

- **filter** (*string*) – The *filter* to apply.
- **lang** (*string*) – Activate translations in the specific language for that query. (See *translated fields*)
- **page** (*int*) – 1-based page number. Defaults to 1.
- **page_size** (*int*) – Maximum number of results to return for the requested page. Defaults to 25.

Response JSON Object

- **count** (*int*) – The number of versions for this add-on.
- **next** (*string*) – The URL of the next page of results.
- **previous** (*string*) – The URL of the previous page of results.
- **results** (*array*) – An array of *versions*.

By default, the version list API will only return public versions (excluding versions that have incomplete, disabled, deleted, rejected or flagged for further review files) - you can change that with the `filter` query parameter, which may require authentication and specific permissions depending on the value:

Value	Description
all_without_unlisted	Show all listed versions attached to this add-on. Requires either reviewer permissions or a user account listed as a developer of the add-on.
all_with_unlisted	Show all versions (including unlisted) attached to this add-on. Requires either reviewer permissions or a user account listed as a developer of the add-on.
all_with_deleted	Show all versions attached to this add-on, including deleted ones. Requires admin permissions.

Version Detail

This endpoint allows you to fetch a single version belonging to a specific add-on.

GET /api/v3/addons/addon/(int:addon_id|string:addon_slug|string:addon_guid)/versions/(int:id)/

Query Parameters

- **lang** (*string*) – Activate translations in the specific language for that query. (See *translated fields*)

Response JSON Object

- **id** (*int*) – The version id.
- **channel** (*string*) – The version channel, which determines its visibility on the site. Can be either `unlisted` or `listed`.
- **compatibility** (*object*) – Object detailing which *applications* the version is compatible with. The exact min/max version numbers in the object correspond to valid application versions. Example:

```
{
  "compatibility": {
    "android": {
      "min": "38.0a1",
      "max": "43.0"
    },
    "firefox": {
      "min": "38.0a1",
      "max": "43.0"
    }
  }
}
```

- **compatibility[app_name].max** (*object*) – Maximum version of the corresponding app the version is compatible with. Should only be enforced by clients if `is_strict_compatibility_enabled` is `true`.
- **compatibility[app_name].min** (*object*) – Minimum version of the corresponding app the version is compatible with.
- **edit_url** (*string*) – The URL to the developer edit page for the version.
- **files** (*array*) – Array holding information about the files for the version.
- **files[].id** (*int*) – The id for a file.
- **files[].created** (*string*) – The creation date for a file.
- **files[].hash** (*string*) – The hash for a file.
- **files[].platform** (*string*) – The *platform* for a file.
- **files[].id** – The size for a file, in bytes.
- **files[].is_mozilla_signed_extension** (*boolean*) – Whether the file was signed with a Mozilla internal certificate or not.
- **files[].is_restart_required** (*boolean*) – Whether the file requires a browser restart to work once installed or not.

- **files[].is_webextension** (*boolean*) – Whether the file is a WebExtension or not.
- **files[].status** (*int*) – The *status* for a file.
- **files[].url** (*string*) – The (absolute) URL to download a file. Clients using this API can append an optional `src` query parameter to the url which would indicate the source of the request (See *download sources*).
- **files[].permissions[]** (*array*) – Array of the webextension permissions for this File, as strings. Empty for non-webextensions.
- **license** (*object*) – Object holding information about the license for the version. For performance reasons this field is omitted from add-on search endpoint.
- **license.name** (*string|object|null*) – The name of the license (See *translated fields*).
- **license.text** (*string|object|null*) – The text of the license (See *translated fields*). For performance reasons this field is omitted from add-on detail endpoint.
- **license.url** (*string|null*) – The URL of the full text of license.
- **release_notes** (*string|object|null*) – The release notes for this version (See *translated fields*). For performance reasons this field is omitted from add-on search endpoint.
- **reviewed** (*string*) – The date the version was reviewed at.
- **is_strict_compatibility_enabled** (*boolean*) – Whether or not this version has strictCompatibility. set.
- **version** (*string*) – The version number string for the version.

Add-on EULA and Privacy Policy

This endpoint allows you to fetch an add-on EULA and privacy policy.

GET /api/v3/addons/addon/(int:id|string:slug|string:guid)/eula_policy/

Note: Non-public add-ons and add-ons with only unlisted versions require both:

- authentication
- reviewer permissions or an account listed as a developer of the add-on

---

Response JSON Object

- **eula** (*string|object|null*) – The text of the EULA, if present (See *translated fields*).
- **privacy_policy** (*string|object|null*) – The text of the Privacy Policy, if present (See *translated fields*).

Language Tools

This endpoint allows you to list all public language tools add-ons available on AMO.

GET /api/v3/addons/language-tools/

Note: Because this endpoint is intended to be used to feed a page that displays all available language tools in a single page, it is not paginated as normal, and instead will return all results without obeying regular pagination parameters. The ordering is left undefined, it's up to the clients to re-order results as needed before displaying the add-ons to the end-users.

In addition, the results can be cached for up to 24 hours, based on the full URL used in the request.

Query Parameters

- **app** (*string*) – Mandatory. Filter by *add-on application* availability.
- **appversion** (*string*) – Filter by application version compatibility. Pass the full version as a string, e.g. `46.0`. Only valid when both the `app` and `type` parameters are also present, and only makes sense for Language Packs, since Dictionaries are always compatible with every application version.
- **lang** (*string*) – Activate translations in the specific language for that query. (See *translated fields*)
- **type** (*string*) – Mandatory when `appversion` is present. Filter by *add-on type*. The default is to return both Language Packs or Dictionaries.

Response JSON Object

- **results** (*array*) – An array of language tools.
- **results[].id** (*int*) – The add-on id on AMO.
- **results[].current_compatible_version** (*object*) – Object holding the latest publicly available *version* of the add-on compatible with the `appversion` parameter used. Only present when `appversion` is passed and valid. For performance reasons, only the following version properties are returned on the object: `id`, `files`, `reviewed`, and `version`.
- **results[].default_locale** (*string*) – The add-on default locale for translations.
- **results[].name** (*string|object|null*) – The add-on name (See *translated fields*).
- **results[].guid** (*string*) – The add-on extension identifier.
- **results[].locale_disambiguation** (*string|null*) – Free text field allowing clients to distinguish between multiple dictionaries in the same locale but different spellings. Only present when using the Language Tools endpoint.
- **results[].slug** (*string*) – The add-on slug.
- **results[].target_locale** (*string*) – For dictionaries and language packs, the locale the add-on is meant for. Only present when using the Language Tools endpoint.
- **results[].type** (*string*) – The *add-on type*.
- **results[].url** (*string*) – The (absolute) add-on detail URL.

Replacement Add-ons

This endpoint returns a list of suggested replacements for legacy add-ons that are unsupported in Firefox 57. Added to support the TAAR recommendation service.

GET /api/v3/addons/replacement-addon/

Query Parameters

- **page** (*int*) – 1-based page number. Defaults to 1.
- **page_size** (*int*) – Maximum number of results to return for the requested page. Defaults to 25.

Response JSON Object

- **count** (*int*) – The total number of replacements.
- **next** (*string*) – The URL of the next page of results.
- **previous** (*string*) – The URL of the previous page of results.
- **results** (*array*) – An array of replacements matches.
- **results[].guid** (*string*) – The extension identifier of the legacy add-on.
- **results[].replacement[]** (*string*) – An array of guids for the replacements add-ons. If there is a direct replacement this will be a list of one add-on guid. The list can be empty if all the replacement add-ons are invalid (e.g. not publicly available on AMO). The list will also be empty if the replacement is to a url that is not an addon or collection.

Compat Override

This endpoint allows compatibility overrides specified by AMO admins to be searched. Compatibilty overrides are used within Firefox i(and other toolkit applications e.g. Thunderbird) to change compatibility of installed add-ons where they have stopped working correctly in new release of Firefox, etc.

GET /api/v3/addons/compat-override/

Query Parameters

- **guid** (*string*) – Filter by exact add-on guid. Multiple guids can be specified, separated by comma(s), in which case any add-ons matching any of the guids will be returned. As guids are unique there should be at most one add-on result per guid specified.
- **page** (*int*) – 1-based page number. Defaults to 1.
- **page_size** (*int*) – Maximum number of results to return for the requested page. Defaults to 25.

Response JSON Object

- **count** (*int*) – The number of results for this query.
- **next** (*string*) – The URL of the next page of results.
- **previous** (*string*) – The URL of the previous page of results.
- **results** (*array*) – An array of compat overrides.
- **results[].addon_id** (*int|null*) – The add-on identifier on AMO, if specified.
- **results[].addon_guid** (*string*) – The add-on extension identifier.

- **results[].name** (*string*) – A description entered by AMO admins to describe the override.
- **results[].version_ranges** (*array*) – An array of affected versions of the add-on.
- **results[].version_ranges[].addon_min_version** (*string*) – minimum version of the add-on to be disabled.
- **results[].version_ranges[].addon_max_version** (*string*) – maximum version of the add-on to be disabled.
- **results[].version_ranges[].applications** (*array*) – An array of affected applications for this range of versions.
- **results[].version_ranges[].applications[].name** (*string*) – Application name (e.g. Firefox).
- **results[].version_ranges[].applications[].id** (*int*) – Application id on AMO.
- **results[].version_ranges[].applications[].min_version** (*string*) – minimum version of the application to be disabled in.
- **results[].version_ranges[].applications[].max_version** (*string*) – maximum version of the application to be disabled in.
- **results[].version_ranges[].applications[].guid** (*string*) – Application guid.

Recommendations

This endpoint provides recommendations of other addons to install, fetched from the recommendation service. Four recommendations are fetched, but only valid, publicly available addons are shown (so max 4 will be returned, and possibly less).

GET /api/v3/addons/recommendations/

Query Parameters

- **guid** (*string*) – Fetch recommendations for this add-on guid.
- **lang** (*string*) – Activate translations in the specific language for that query. (See *translated fields*)
- **recommended** (*boolean*) – Fetch recommendations from the recommendation service, or return a curated fallback list instead.

Response JSON Object

- **outcome** (*string*) – Outcome of the response returned. Will be either: `recommended` - responses from recommendation service; `recommended_fallback` - service timed out or returned empty results so we returned fallback; `curated` - `recommended=False` was requested so fallback returned.
- **fallback_reason** (*string|null*) – if `outcome` was `recommended_fallback` then the reason why. Will be either: `timeout` or `no_results`.
- **count** (*int*) – The number of results for this query.
- **next** (*string*) – The URL of the next page of results.
- **previous** (*string*) – The URL of the previous page of results.

- **results** (*array*) – An array of *add-ons*. The following fields are omitted for performance reasons: `release_notes` and `license` fields on `current_version` and `current_beta_version`, as well as `picture_url` from `authors`.

2.1.8 Categories

Note: These APIs are experimental and are currently being worked on. Endpoints may change without warning.

Category List

Categories are defined by a name, a slug, a type and an application. Slugs are only guaranteed to be unique for a given `app` and `type` combination, and can therefore be re-used for different categories.

This endpoint is not paginated.

GET /api/v3/addons/categories/

Response JSON Object

- **id** (*int*) – The category id.
- **name** (*string*) – The category name. Returns the already translated string.
- **slug** (*string*) – The category slug. See *csv table* for more possible values.
- **application** (*string*) – Application, see *add-on application* for more details.
- **misc** (*boolean*) – Whether or not the category is miscellaneous.
- **type** (*string*) – Category type, see *add-on type* for more details.
- **weight** (*int*) – Category weight used in sort ordering.
- **description** (*string*) – The category description. Returns the already translated string.

Current categories

Name	Slug	Type	Application
Alerts & Updates	alerts-updates	extension	firefox
Appearance	appearance	extension	firefox
Bookmarks	bookmarks	extension	firefox
Download Management	download-management	extension	firefox
Feeds, News & Blogging	feeds-news-blogging	extension	firefox
Games & Entertainment	games-entertainment	extension	firefox
Language Support	language-support	extension	firefox
Photos, Music & Videos	photos-music-videos	extension	firefox
Privacy & Security	privacy-security	extension	firefox
Search Tools	search-tools	extension	firefox
Shopping	shopping	extension	firefox
Social & Communication	social-communication	extension	firefox
Tabs	tabs	extension	firefox
Web Development	web-development	extension	firefox

continues on next page

Table 1 – continued from previous page

Name	Slug	Type	Application
Other	other	extension	firefox
Animals	animals	theme	firefox
Compact	compact	theme	firefox
Large	large	theme	firefox
Miscellaneous	miscellaneous	theme	firefox
Modern	modern	theme	firefox
Nature	nature	theme	firefox
OS Integration	os-integration	theme	firefox
Retro	retro	theme	firefox
Sports	sports	theme	firefox
General	general	dictionary	firefox
Bookmarks	bookmarks	search	firefox
Business	business	search	firefox
Dictionaries & Encyclopedias	dictionaries-encyclopedias	search	firefox
General	general	search	firefox
Kids	kids	search	firefox
Multiple Search	multiple-search	search	firefox
Music	music	search	firefox
News & Blogs	news-blogs	search	firefox
Photos & Images	photos-images	search	firefox
Shopping & E-Commerce	shopping-e-commerce	search	firefox
Social & People	social-people	search	firefox
Sports	sports	search	firefox
Travel	travel	search	firefox
Video	video	search	firefox
General	general	language	firefox
Abstract	abstract	persona	firefox
Causes	causes	persona	firefox
Fashion	fashion	persona	firefox
Film and TV	film-and-tv	persona	firefox
Firefox	firefox	persona	firefox
Foxkeh	foxkeh	persona	firefox
Holiday	holiday	persona	firefox
Music	music	persona	firefox
Nature	nature	persona	firefox
Other	other	persona	firefox
Scenery	scenery	persona	firefox
Seasonal	seasonal	persona	firefox
Solid	solid	persona	firefox
Sports	sports	persona	firefox
Websites	websites	persona	firefox
Appearance and Customization	appearance	extension	thunderbird
Calendar and Date/Time	calendar	extension	thunderbird
Chat and IM	chat	extension	thunderbird
Contacts	contacts	extension	thunderbird
Folders and Filters	folders-and-filters	extension	thunderbird
Import/Export	importexport	extension	thunderbird
Language Support	language-support	extension	thunderbird
Message Composition	composition	extension	thunderbird

continues on next page

Table 1 – continued from previous page

Name	Slug	Type	Application
Message and News Reading	message-and-news-reading	extension	thunderbird
Miscellaneous	miscellaneous	extension	thunderbird
Privacy and Security	privacy-and-security	extension	thunderbird
Tags	tags	extension	thunderbird
Compact	compact	theme	thunderbird
Miscellaneous	miscellaneous	theme	thunderbird
Modern	modern	theme	thunderbird
Nature	nature	theme	thunderbird
General	general	dictionary	thunderbird
General	general	language	thunderbird
Bookmarks	bookmarks	extension	seamonkey
Downloading and File Management	downloading-and-file-management	extension	seamonkey
Interface Customizations	interface-customizations	extension	seamonkey
Language Support and Translation	language-support-and-translation	extension	seamonkey
Miscellaneous	miscellaneous	extension	seamonkey
Photos and Media	photos-and-media	extension	seamonkey
Privacy and Security	privacy-and-security	extension	seamonkey
RSS, News and Blogging	rss-news-and-blogging	extension	seamonkey
Search Tools	search-tools	extension	seamonkey
Site-specific	site-specific	extension	seamonkey
Web and Developer Tools	web-and-developer-tools	extension	seamonkey
Miscellaneous	miscellaneous	theme	seamonkey
General	general	dictionary	seamonkey
General	general	language	seamonkey
Device Features & Location	device-features-location	extension	android
Experimental	experimental	extension	android
Feeds, News, & Blogging	feeds-news-blogging	extension	android
Performance	performance	extension	android
Photos & Media	photos-media	extension	android
Security & Privacy	security-privacy	extension	android
Shopping	shopping	extension	android
Social Networking	social-networking	extension	android
Sports & Games	sports-games	extension	android
User Interface	user-interface	extension	android

2.1.9 Collections

The following API endpoints cover user created collections.

List

Note: This API requires *authentication* and *Collections:Edit* permission to list collections other than your own.

This endpoint allows you to list all collections authored by the specified user. The results are sorted by the most recently updated collection first.

GET /api/v3/accounts/account/(int:user_id|string:username)/collections/

Response JSON Object

- **count** (*int*) – The number of results for this query.
- **next** (*string*) – The URL of the next page of results.
- **previous** (*string*) – The URL of the previous page of results.
- **results** (*array*) – An array of *collections*.

Detail

This endpoint allows you to fetch a single collection by its `slug`. It returns any `public` collection by the specified user. You can access a non-`public` collection only if it was authored by you, the authenticated user. If your account has the *Collections:Edit* permission then you can access any collection.

GET /api/v3/accounts/account/(int:user_id|string:username)/collections/(string:collection_slug)/

Response JSON Object

- **id** (*int*) – The id for the collection.
- **addon_count** (*int*) – The number of add-ons in this collection.
- **author.id** (*int*) – The id of the author (creator) of the collection.
- **author.name** (*string*) – The name of the author.
- **author.url** (*string*) – The link to the profile page for of the author.
- **author.username** (*string*) – The username of the author.
- **default_locale** (*string*) – The default locale of the description and name fields. (See *translated fields*).
- **description** (*string|object|null*) – The description the author added to the collection. (See *translated fields*).
- **modified** (*string*) – The date the collection was last updated.
- **name** (*string|object*) – The name of the collection. (See *translated fields*).
- **public** (*boolean*) – Whether the collection is *listed* - publicly viewable.
- **slug** (*string*) – The name used in the URL.
- **url** (*string*) – The (absolute) collection detail URL.
- **uuid** (*string*) – A unique identifier for this collection; primarily used to count addon installations that come via this collection.

If the `with_addons` parameter is passed then *addons in the collection* are returned along with the detail. Add-ons returned are limited to the first 25 in the collection, in the default sort (popularity, descending). Filtering is as per *collection addon list endpoint* - i.e. defaults to only including public add-ons. Additional add-ons can be returned from the *Collection Add-on list endpoint*.

GET /api/v3/accounts/account/(int:user_id|string:username)/collections/(string:collection_slug)/?with_addons

Query Parameters

- **filter** (*string*) – The *filter* to apply.

Response JSON Object

- **id** (*int*) – The id for the collection.
- **addon_count** (*int*) – The number of add-ons in this collection.
- **addons** (*array*) – An array of *addons with notes*.

... rest as *collection detail response*

Create

Note: This API requires *authentication*.

This endpoint allows a collection to be created under your account. Any fields in the *collection* but not listed below are not settable and will be ignored in the request.

POST /api/v3/accounts/account/(int:user_id|string:username)/collections/

Request JSON Object

- **default_locale** (*string|null*) – The default locale of the description and name fields. Defaults to *en-US*. (See *translated fields*).
- **description** (*string|object|null*) – The description the author added to the collection. (See *translated fields*).
- **name** (*string|object*) – The name of the collection. (required) (See *translated fields*).
- **public** (*boolean*) – Whether the collection is *listed* - publicly viewable. Defaults to *True*.
- **slug** (*string*) – The name used in the URL (required).

Edit

Note: This API requires *authentication* and *Collections:Edit* permission to edit collections other than your own.

This endpoint allows some of the details for a collection to be updated. Any fields in the *collection* but not listed below are not editable and will be ignored in the patch request.

PATCH /api/v3/accounts/account/(int:user_id|string:username)/collections/(string: *collection_slug*) **/**

Request JSON Object

- **default_locale** (*string*) – The default locale of the description and name fields. (See *translated fields*).

- **description** (*string|object|null*) – The description the author added to the collection. (See *translated fields*).
- **name** (*string|object*) – The name of the collection. (See *translated fields*).
- **public** (*boolean*) – Whether the collection is *listed* - publicly viewable.
- **slug** (*string*) – The name used in the URL.

Delete

Note: This API requires *authentication* and *Collections:Edit* permission to delete collections other than your own.

This endpoint allows the collection to be deleted.

DELETE /api/v3/accounts/account/(int:user_id|string:username)/collections/(string:*collection_slug***)/**

Collection Add-ons List

This endpoint lists the add-ons in a collection, together with collector's notes.

GET /api/v3/accounts/account/(int:user_id|string:username)/collections/(string:*collection_slug***)/addons/**

Query Parameters

- **filter** (*string*) – The *filter* to apply.
- **sort** (*string*) – The sort parameter. The available parameters are documented in the *table below*.

Response JSON Object

- **count** (*int*) – The number of results for this query.
- **next** (*string*) – The URL of the next page of results.
- **previous** (*string*) – The URL of the previous page of results.
- **results** (*array*) – An array of *items* in this collection.

Available sorting parameters:

Parameter	Description
added	Date the add-on was added to the collection, ascending.
popularity	Number of total weekly downloads of the add-on, ascending.
name	Add-on name, ascending.

All sort parameters can be reversed, e.g. '-added' for descending dates. The default sorting is by popularity, descending ('-popularity').

By default, the collection addon list API will only return public add-ons (excluding add-ons that have no approved listed versions, are disabled or deleted) - you can change that with the `filter` query parameter:

Value	Description
all	Show all add-ons in the collection, including those that have non-public statuses. This still excludes deleted add-ons.
all_with_deleted	Show all add-ons in the collection, including deleted add-ons too.

Collection Add-ons Detail

This endpoint gets details of a single add-on in a collection, together with collector's notes.

GET /api/v3/accounts/account/(int:user_id|string:username)/collections/(string: *collection_slug*) **/addons/(int:addon_id|stri**

Response JSON Object

- **addon** (*object*) – The *add-on* for this item.
- **notes** (*string|object|null*) – The collectors notes for this item. (See *translated fields*).
- **downloads** (*int*) – The downloads that occured via this collection.

Collection Add-ons Create

Note: This API requires *authentication* and *Collections:Edit* permission to edit collections other than your own.

This endpoint allows a single add-on to be added to a collection, optionally with collector's notes.

POST /api/v3/accounts/account/(int:user_id|string:username)/collections/(string: *collection_slug*) **/addons/**

Request JSON Object

- **addon** (*string*) – The add-on id or slug to be added (required).
- **notes** (*string|object|null*) – The collectors notes for this item. (See *translated fields*).

Collection Add-ons Edit

Note: This API requires *authentication* and *Collections:Edit* permission to edit collections other than your own.

This endpoint allows the collector's notes for single add-on to be updated.

PATCH /api/v3/accounts/account/(int:user_id|string:username)/collections/(string:*collection_slug***)/addons/(int:addon_id|str**

Request JSON Object

- **notes** (*string|object|null*) – The collectors notes for this item. (See *translated fields*).

Collection Add-ons Delete

Note: This API requires *authentication* and *Collections:Edit* permission to edit collections other than your own.

This endpoint allows a single add-on to be removed from a collection.

DELETE /api/v3/accounts/account/(int:user_id|string:username)/collections/(string:*collection_slug***)/addons/(int:addon_id|st**

2.1.10 Discovery

Note: These APIs are experimental and are currently being worked on. Endpoints may change without warning.

Discovery Content

This endpoint allows you to fetch content for the new Discovery Pane in Firefox (about:addons).

GET /api/v3/discovery/

Query Parameters

- **lang** (*string*) – Activate translations in the specific language for that query. (See *translated fields*)
- **edition** (*string*) – Return content for a specific edition of Firefox. Currently only `china` is supported.

Response JSON Object

- **count** (*int*) – The number of results for this query.
- **results** (*array*) – The array containing the results for this query.
- **results[].heading** (*string*) – The heading for this item. May contain some HTML tags.
- **results[].description** (*string|null*) – The description for this item, if any. May contain some HTML tags.
- **results[].is_recommendation** (*boolean*) – If this item was from the recommendation service, rather than static curated content.
- **results[].addon** (*object*) – The *add-on* for this item. Only a subset of fields are present: `id`, `current_version` (with only the `compatibility` and `files` fields present), `guid`, `icon_url`, `name`, `slug`, `theme_data`, `type` and `url`.

Discovery Recommendations

If a telemetry client id is passed as a parameter to the discovery pane api endpoint then static curated content is amended with recommendations from the recommendation service. The same number of results will be returned as a standard discovery response and only extensions (not themes) are recommeded. Only valid, publicly available addons are shown.

E.g. a standard discovery pane will display 7 items, 4 extensions and 3 themes. Up to 4 extensions will be replaced with recommendations; the 3 themes will not be replaced. The API will still return a total of 7 items.

Note:

Specifying an `edition` parameter disables recommendations - the `telemetry-client-id` is ignored and standard discovery response returned.

GET /api/v3/discovery/?telemetry-client-id=12345678-90ab-cdef-1234-567890abcdef

Query Parameters

- **telemetry-client-id** (*string*) – The telemetry client ID to be passed to the TAAR service.
- **lang** (*string*) – In addition to activating translations (see *Discovery Content*), this will be passed as *locale* to TAAR.
- **platform** (*string*) – The platform identifier to be passed to TAAR.
- **branch** (*string*) – Additional parameter passed along to TAAR.
- **study** (*string*) – Additional parameter passed along to TAAR.

2.1.11 Download Sources

When requesting an add-on file URL, clients have the option to indicate what is the source of the request. This will then be used to group by downloads by sources in the add-on statistics page.

To indicate the source, add the `src` query parameter to the download URL. The following values are recognized:

Name	Description
api	Add-ons Manager
discovery-promo	Add-ons Manager Promo
discovery-featured	Add-ons Manager Featured
discovery-learnmore	Add-ons Manager Learn More
ss	Search Suggestions
search	Search Results
homepagepromo	Homepage Promo
hp-btn-promo	Homepage Promo
hp-dl-promo	Homepage Promo
hp-hc-featured	Homepage Featured
hp-dl-featured	Homepage Featured
hp-hc-upandcoming	Homepage Up and Coming
hp-dl-upandcoming	Homepage Up and Coming
hp-dl-mostpopular	Homepage Most Popular
dp-btn-primary	Detail Page
dp-btn-version	Detail Page (bottom)
addondetail	Detail Page
addon-detail-version	Detail Page (bottom)
dp-btn-devchannel	Detail Page (Development Channel)
oftenusedwith	Often Used With
dp-hc-oftenusedwith	Often Used With
dp-dl-oftenusedwith	Often Used With
dp-hc-othersby	Others By Author
dp-dl-othersby	Others By Author
dp-hc-dependencies	Dependencies
dp-dl-dependencies	Dependencies
dp-hc-upsell	Upsell
dp-dl-upsell	Upsell
developers	Meet the Developer
userprofile	User Profile
version-history	Version History
sharingapi	Sharing
category	Category Pages
collection	Collections
cb-hc-featured	Category Landing Featured Carousel
cb-dl-featured	Category Landing Featured Carousel
cb-hc-toprated	Category Landing Top Rated
cb-dl-toprated	Category Landing Top Rated
cb-hc-mostpopular	Category Landing Most Popular
cb-dl-mostpopular	Category Landing Most Popular
cb-hc-recentlyadded	Category Landing Recently Added
cb-dl-recentlyadded	Category Landing Recently Added
cb-btn-featured	Browse Listing Featured Sort
cb-dl-featured	Browse Listing Featured Sort

continues on next page

Table 2 – continued from previous page

Name	Description
cb-btn-users	Browse Listing Users Sort
cb-dl-users	Browse Listing Users Sort
cb-btn-rating	Browse Listing Rating Sort
cb-dl-rating	Browse Listing Rating Sort
cb-btn-created	Browse Listing Created Sort
cb-dl-created	Browse Listing Created Sort
cb-btn-name	Browse Listing Name Sort
cb-dl-name	Browse Listing Name Sort
cb-btn-popular	Browse Listing Popular Sort
cb-dl-popular	Browse Listing Popular Sort
cb-btn-updated	Browse Listing Updated Sort
cb-dl-updated	Browse Listing Updated Sort
cb-btn-hotness	Browse Listing Up and Coming Sort
cb-dl-hotness	Browse Listing Up and Coming Sort
find-replacement	Find replacement service for obsolete add-ons

2.1.12 Reviews

Note: These APIs are experimental and are currently being worked on. Endpoints may change without warning. The only authentication method available at the moment is *the internal one*.

List reviews

This endpoint allows you to fetch reviews for a given add-on or user. Either `addon` or `user` query parameters are required, and they can be combined together.

When `addon`, `user` and `version` are passed on the same request, `page_size` will automatically be set to `1`, since an user can only post one review per version of a given add-on. This can be useful to find out if a user has already posted a review for the current version of an add-on.

`GET /api/v3/reviews/review/`

Query Parameters

- **`addon`** (*`string`*) – The *add-on* id, slug, or guid to fetch reviews from. When passed, the reviews shown will always be the latest posted by each user on this particular add-on (which means there should only be one review per user in the results), unless the `version` parameter is also passed.
- **`filter`** (*`string`*) – The *filter(s)* to apply.
- **`user`** (*`string`*) – The user id to fetch reviews from.
- **`show_grouped_ratings`** (*`boolean`*) – Whether or not to show ratings aggregates for this add-on in the response (Use "true"/"1" as truthy values, "0"/"false" as falsy ones).
- **`version`** (*`string`*) – The version id to fetch reviews from.
- **`page`** (*`int`*) – 1-based page number. Defaults to 1.
- **`page_size`** (*`int`*) – Maximum number of results to return for the requested page. Defaults to 25.

Response JSON Object

- **count** (*int*) – The number of results for this query.
- **next** (*string*) – The URL of the next page of results.
- **previous** (*string*) – The URL of the previous page of results.
- **results** (*array*) – An array of *reviews*.
- **grouped_ratings** (*object*) – Only present if `show_grouped_ratings` query parameter is present. An object with 5 key-value pairs, the keys representing each possible rating (Though a number, it has to be converted to a string because of the JSON formatting) and the values being the number of times the corresponding rating has been posted for this add-on, e.g. `{"1": 4, "2": 8, "3": 15, "4": 16: "5": 23}`.

By default, the review list API will only return not-deleted reviews, and include reviews without text. You can change that with the `filter` query parameter. You can filter by multiple values, e.g. `filter=with_deleted,without_empty_body,with_yours`

Value	Description
with_deleted	Returns deleted reviews too. This requires the Addons:Edit permission.
without_empty_body	Excludes reviews that only contain a rating, and no textual content.
with_yours	Used in combination *without_empty_body* to include your own reviews, even if they have no text.

Detail

This endpoint allows you to fetch a review by its id.

GET /api/v3/reviews/review/(int: *id*)/

Response JSON Object

- **id** (*int*) – The review id.
- **addon** (*object*) – An object included for convenience that contains only two properties: `id` and `slug`, corresponding to the add-on id and slug.
- **body** (*string|null*) – The text of the review.
- **is_latest** (*boolean*) – Boolean indicating whether the review is the latest posted by the user on the same add-on.
- **previous_count** (*int*) – The number of reviews posted by the user on the same add-on before this one.
- **rating** (*int*) – The rating the user gave as part of the review.
- **reply** (*object|null*) – The review object containing the developer reply to this review, if any (The fields `rating`, `reply` and `version` are omitted).
- **title** (*string|null*) – The title of the review.
- **version.id** (*int*) – The add-on version id the review applies to.
- **version.version** (*string*) – The add-on version string the review applies to.
- **user** (*object*) – Object holding information about the user who posted the review.

- **user.id** (*string*) – The user id.
- **user.name** (*string*) – The user name.
- **user.url** (*string*) – The user profile URL.
- **user.username** (*string*) – The user username.

Post

This endpoint allows you to post a new review for a given add-on and version. If successful a *review object* is returned.

> **Note:** Requires authentication.

POST /api/v3/reviews/review/

Request JSON Object

- **addon** (*string*) – The add-on id the review applies to (required).
- **body** (*string|null*) – The text of the review.
- **title** (*string|null*) – The title of the review.
- **rating** (*int*) – The rating the user wants to give as part of the review (required).
- **version** (*int*) – The add-on version id the review applies to (required).

Edit

This endpoint allows you to edit an existing review by its id. If successful a *review object* is returned.

> **Note:** Requires authentication and Addons:Edit permissions or the user account that posted the review.
>
> Only body, title and rating are allowed for modification.

PATCH /api/v3/reviews/review/(int: *id*)/

Request JSON Object

- **body** (*string|null*) – The text of the review.
- **title** (*string|null*) – The title of the review.
- **rating** (*int*) – The rating the user wants to give as part of the review.

Delete

This endpoint allows you to delete an existing review by its id.

> **Note:** Requires authentication and Addons:Edit permission or the user account that posted the review. Even with the right permission, users can not delete a review from somebody else if it was posted on an add-on they are listed as a developer of.

DELETE /api/v3/reviews/review/(int: *id*)/

Reply

This endpoint allows you to reply to an existing user review. If successful a *review reply object* is returned.

> **Note:** Requires authentication and either Addons:Edit permission or a user account listed as a developer of the add-on.

POST /api/v3/reviews/review/(int: *id*)/reply/

Request JSON Object

- **body** (*string*) – The text of the reply (required).
- **title** (*string|null*) – The title of the reply.

Flag

This endpoint allows you to flag an existing user review, to let a moderator know that something may be wrong with it.

> **Note:** Requires authentication and a user account different from the one that posted the review.

POST /api/v3/reviews/review/(int: *id*)/flag/

Request JSON Object

- **flag** (*string*) – A *constant* describing the reason behind the flagging.
- **note** (*string|null*) – A note to explain further the reason behind the flagging. This field is required if the flag is `review_flag_reason_other`, and passing it will automatically change the flag to that value.

Response JSON Object

- **object** – If successful, an object with a `msg` property containing a success message. If not, an object indicating which fields contain errors.

Available constants for the `flag` property:

Constant	Description
review_flag_reason_spam	Spam or otherwise non-review content
review_flag_reason_language	Inappropriate language/dialog
review_flag_reason_bug_support	Misplaced bug report or support request
review_flag_reason_other	Other (please specify)

2.1.13 Reviewers

Note: These APIs are experimental and are currently being worked on. Endpoints may change without warning. The only authentication method available at the moment is *the internal one*.

Subscribe

This endpoint allows you to subscribe the current user to the notification sent when a new listed version is submitted on a particular add-on.

Note: Requires authentication and the current user to have any reviewer-related permission.

Unsubscribe

This endpoint allows you to unsubscribe the current user to the notification sent when a new listed version is submitted on a particular add-on.

Note: Requires authentication and the current user to have any reviewer-related permission.

Disable

This endpoint allows you to disable the public listing for an add-on.

Note:

Requires authentication and the current user to have `Reviews:Admin` permission.

Enable

This endpoint allows you to re-enable the public listing for an add-on. If the add-on can't be public because it does not have public versions, it will instead be changed to awaiting review or incomplete depending on the status of its versions.

Note: Requires authentication and the current user to have `Reviews:Admin` permission.

Flags

This endpoint allows you to manipulate various reviewer-specific flags on an add-on.

> **Note:** Requires authentication and the current user to have `Reviews:Admin` permission.

2.1.14 Signing

Note: This API requires *authentication*.

The following API endpoints help you get your add-on signed by Mozilla so it can be installed into Firefox without error. See extension signing for more details about Firefox's signing policy.

Client Libraries

The following libraries will make it easier to use the signing API:

- sign-addon, for general programattic use in NodeJS
- web-ext sign, for developing Web Extensions

If you are using `curl` to interact with the API you should be sure to pass the `-g` flag to skip "URL globbing" which won't interact well with add-on Ids that have {} characters in them.

Uploading a version

You can upload a new version for signing by issuing a `PUT` request and including the contents of your add-on in the `upload` parameter as multi-part formdata. This will create a pending version on the add-on and will prevent future submissions to this version unless validation or review fails.

If the upload succeeded then it will be submitted for validation and you will be able to check its status.

PUT /api/v3/addons/(string: *guid*)/versions/string: *version*/ Request:

```
curl "https://addons.mozilla.org/api/v3/addons/@my-addon/versions/1.0/"
    -g -XPUT --form "upload=@build/my-addon.xpi"
    -H "Authorization: JWT <jwt-token>"
```

Parameters

- **guid** – The add-on extension identifier.
- **version** – The version of the add-on.

Form Parameters

- **upload** – The add-on file being uploaded.
- **channel** – (optional) The channel this version should be uploaded to, which determines its visibility on the site. It can be either `unlisted` or `listed`. See the note below.

Request Headers

- Content-Type – multipart/form-data

Note: `channel` is only valid for new versions on existing add-ons. If the add-on is new then the version will be created as `unlisted`. If the parameter isn't supplied then the channel of the most recent version (submitted either via this API or the website) will be assumed. For example, if you submit a version as `listed` then the next version will be `listed` if you don't specify the channel.

Response:

The response body will be the same as the *Checking the status of your upload* response.

Status Codes

- 201 Created – new add-on and version created.
- 202 Accepted – new version created.
- 400 Bad Request – an error occurred, check the *error* value in the JSON.
- 401 Unauthorized – authentication failed.
- 403 Forbidden – you do not own this add-on.
- 409 Conflict – version already exists.

Uploading without an ID

Note: This is only valid for WebExtensions. All other add-on types require an add-on ID and have to use the regular endpoint to *upload a version*.

`POST /api/v3/addons/`

Request:

```
curl "https://addons.mozilla.org/api/v3/addons/"
    -g -XPOST -F "upload=@build/my-addon.xpi" -F "version=1.0"
    -H "Authorization: JWT <jwt-token>"
```

Form Parameters

- **`upload`** – The add-on file being uploaded.
- **`version`** – The version of the add-on.

Request Headers

- Content-Type – multipart/form-data

Response:

The response body will be the same as the *Checking the status of your upload* response.

Status Codes

- 201 Created – new add-on and version created.
- 202 Accepted – new version created.
- 400 Bad Request – an error occurred, check the *error* value in the JSON.
- 401 Unauthorized – authentication failed.

- 403 Forbidden – you do not own this add-on.
- 409 Conflict – version already exists.

Creating an add-on

If this is the first time that your add-on's UUID has been seen then the add-on will be created as an unlisted add-on when the version is uploaded.

Checking the status of your upload

You can check the status of your upload by issuing a `GET` request. There are a few things that will happen once a version is uploaded and the status of those events is included in the response.

Once validation is completed (whether it passes or fails) then the `processed` property will be `true`. You can check if validation passed using the `valid` property and check the results with `validation_results`.

If validation passed then your add-on will be submitted for review. In the case of unlisted add-ons this will happen automatically. If your add-on is listed then it will be reviewed by a human and that will take a bit longer. You can check the `automated_signing` property to see if signing will happen automatically or after a manual review. Once review is complete then the `reviewed` property will be set and you can check the results with the `passed_review` property.

GET /api/v3/addons/(string: *guid*)/versions/ string: *version*/[uploads/(string:upload-pk)/] Request:

```
curl "https://addons.mozilla.org/api/v3/addons/@my-addon/versions/1.0/"
    -g -H "Authorization: JWT <jwt-token>"
```

Parameters

- **guid** – The add-on extension identifier.
- **version** – the version of the add-on.
- **upload-pk** – (optional) the pk for a specific upload.

Response:

```
{
    "guid": "420854ee-7a85-42b9-822f-8e03dc5f6de9",
    "active": true,
    "automated_signing": true,
    "files": [
        {
            "download_url": "https://addons.mozilla.org/api/v3/downloads/file/100/
↪example-id.0-fx+an.xpi?src=api",
            "hash":
↪"sha256:1bb945266bf370170a656350d9b640cbcaf70e671cf753c410e604219cdd9267",
            "signed": true
        }
    ],
    "passed_review": true,
    "pk": "f68abbb3b1624c098fe979a409fe3ce9",
    "processed": true,
    "reviewed": true,
    "url": "https://addons.mozilla.org/api/v3/addons/@example-id.0/uploads/
↪f68abbb3b1624c098fe979a409fe3ce9/",
```

(continues on next page)

(continued from previous page)

```
    "valid": true,
    "validation_results": {},
    "validation_url": "https://addons.mozilla.org/en-US/developers/upload/
↪f68abbb3b1624c098fe979a409fe3ce9",
    "version": "1.0"
}
```

Response JSON Object

- **guid** – The GUID of the addon.
- **active** – version is active.
- **automated_signing** – If true, the version will be signed automatically. If false it will end up in the manual review queue when valid.
- **files[].download_url** – URL to *download the add-on file*.
- **files[].hash** – Hash of the file contents, prefixed by the hashing algorithm used. Example: `sha256:1bb945266bf3701....` In the case of signed files, the hash will be that of the final signed file, not the original unsigned file.
- **files[].signed** – if the file is signed.
- **passed_review** – if the version has passed review.
- **pk** – the pk for this upload.
- **processed** – if the version has been processed by the validator.
- **reviewed** – if the version has been reviewed.
- **url** – URL to check the status of this upload.
- **valid** – if the version passed validation.
- **validation_results** – the validation results (removed from the example for brevity).
- **validation_url** – a URL to the validation results in HTML format.
- **version** – the version.

Status Codes

- 200 OK – request successful.
- 401 Unauthorized – authentication failed.
- 403 Forbidden – you do not own this add-on.
- 404 Not Found – add-on or version not found.

Downloading signed files

When checking on your *request to sign a version*, a successful response will give you an API URL to download the signed files. This endpoint returns the actual file data for download.

GET /api/v3/file/[int:file_id]/[string:base_filename]

Request:

```
curl "https://addons.mozilla.org/api/v3/file/123/some-addon.xpi?src=api"
    -g -H "Authorization: JWT <jwt-token>"
```

Parameters

- **file_id** – the primary key of the add-on file.
- **base_filename** – the base filename. This is just a convenience for clients so that they write meaningful file names to disk.

Response:

There are two possible responses:

- Binary data containing the file
- A header that redirects you to a mirror URL for the file. In this case, the initial response will include a `SHA-256` hash of the file in the header `X-Target-Digest`. Clients should check that the final downloaded file matches this hash.

Status Codes

- 200 OK – request successful.
- 302 Found – file resides at a mirror URL
- 401 Unauthorized – authentication failed.
- 404 Not Found – file does not exist or requester does not have access to it.

/api

/https: