
FreeField Documentation

Release latest

Marius Lindvall

Jun 05, 2019

1	Installation guide	3
1.1	Prerequisites	3
1.2	Downloading FreeField	3
1.3	Setting up the web server	4
1.4	Installation wizard	5
1.5	Post-installation steps	9
2	Authentication	11
2.1	Discord authentication	11
2.2	Telegram authentication	14
2.3	Reddit authentication	21
2.4	Facebook authentication	24
2.5	LINE authentication	28
2.6	GroupMe authentication	31
3	User accounts	35
3.1	User account details	35
3.2	Manual user approval	37
4	Permissions and groups	39
4.1	Default groups	39
4.2	Group settings	41
4.3	Adding and removing groups	42
4.4	Default group for new members	43
4.5	Managing permissions	43
5	Map settings	45
5.1	Map providers	45
6	Pokéstops	47
6.1	Adding a new Pokéstop	47
6.2	Moving a Pokéstop	48
6.3	Removing Pokéstops	49
6.4	Clearing field research	51
6.5	Renaming Pokéstops	51
6.6	Managing Pokéstops	52
6.7	Exporting and importing Pokéstops	53

7	Geofencing	55
7.1	Defining a boundary	55
7.2	Limiting Pokéstop submission	57
7.3	Applying geofences to webhooks	58
8	Webhooks	59
8.1	Webhook basics	59
8.2	Common webhook targets	60
8.3	Webhook properties	66
8.4	Payload	67
8.5	Task-based filtering	72
9	Site appearance	73
9.1	Site name	73
9.2	Message of the Day	74
9.3	Search indexing	75
9.4	Group labels and colors	75
9.5	PWA theming	75
9.6	Favicon	75
9.7	Title bar theme color	75
9.8	Site theme color	76
9.9	UI and map themes	76
9.10	Map markers	77
10	Progressive Web Apps (PWA)	79
10.1	How it works	79
10.2	Configuration	80
10.3	Installation preview	83
11	Installing updates	87
11.1	Update branches	88
11.2	Installing an update	89
11.3	Troubleshooting	91
12	Reporting field research	93
12.1	Submitting a report	93
12.2	Reporting unlisted research objectives	96
13	Filtering Pokéstops	99
14	Searching for Pokéstops	103
15	Frequently asked questions	105
15.1	Why are some markers hidden from the map?	105
16	API Reference	107
16.1	Authentication	107
16.2	Response format	108
16.3	Object reference	108
16.4	Method reference	111
17	Adding research data	123
17.1	Objectives	123
17.2	Common objectives	125
17.3	Rewards	125
17.4	Research parameters	127

17.5	Icons and categories	132
18	Code style	135
18.1	Non-compliance	135
18.2	Indentation	136
18.3	Spacing	136
18.4	Code blocks	136
18.5	Wrapping	138
18.6	Naming	139
18.7	Miscellaneous	139
18.8	Commenting	139
19	Configuration	143
19.1	Internationalization	144
19.2	Domains and sections	144
19.3	Options and data types	145
20	Internationalization	153
20.1	String assignment	153
20.2	Usage	154
20.3	Localization file structure	155
21	Icon sets	157
21.1	Location	157
21.2	File structure	157
21.3	The pack.ini metadata file	158
21.4	List of icon IDs	162
22	Security in FreeField	165
22.1	Cross-site scripting (XSS)	165
22.2	Cross-site request forgery (CSRF)	167
22.3	Securing authentication	167
22.4	Permissions	168
23	Translating FreeField	171
23.1	Getting started	171
23.2	Files to translate	171
23.3	Questions	171

This is the official documentation for FreeField. If you are setting up FreeField and are looking for installation help, please see “Installation guide” below.

CHAPTER 1

Installation guide

Welcome to the FreeField installation guide! This guide will guide you through installing and setting up FreeField for production use.

1.1 Prerequisites

FreeField requires a configured database backend with an account that allows creating, altering and deleting tables, and inserting, updating and deleting data, in the database that will hold FreeField data. All tables created by FreeField will use a table prefix you decide, to prevent it from interfering with tables created by other software using the same database. It is recommended that you use MySQL with FreeField.

A web server with PHP 5.6 or higher is also required. FreeField comes pre-configured for Apache. Setting up FreeField on another web server, such as nginx, may require completing additional setup steps. More information about configuring your webserver is explained in the *Setting up the web server* section below.

Tip: Setting up a web server with PHP is not in scope of this installation guide. However, if you are new to server hosting, there are many good guides you can follow, such as [this one by DigitalOcean](#). You should also follow [this guide](#) from section 4 and on to learn how to control Apache and how to set up virtual hosts (which you will need for HTTPS, which FreeField strongly recommends). Finally, [this guide helps you set up HTTPS](#) for your installation, which is recommended to get the most out of your FreeField installation.

FreeField uses PDO to connect to the database backend. Please ensure that you have enabled PDO with your preferred database backend, e.g. `pdo_mysql` for MySQL, in your `php.ini` file.

1.2 Downloading FreeField

Please download the latest release from [GitHub](#). Extract the files from the downloaded archive to the directory on the server that you wish to install FreeField to.

1.2.1 Rolling release

Some users may prefer installing FreeField as a rolling release directly from git. This is possible by running `git clone https://github.com/bilde2910/FreeField.git`, although installation from point releases (i.e. published packaged releases) are generally recommended for most users. The following table outlines the differences between the two installation methods.

Feature	Point releases (packages)	Rolling release (git clone)
Availability of new features	Within anything from a few hours (alpha) to days/weeks (stable)	Immediately upon push to FreeField repository
Stability	Varies (alpha through stable channels)	Extremely unstable (bleeding edge)
Installing new updates	From web interface	Requires shell access
Specifying update to install	Click to select desired version	Specify target hash using <code>git checkout</code>
Stability of update process	Historically unstable, see e.g. commit d76af3a	Extremely stable
Can switch release plans?	Yes, can switch to rolling release; see e.g. this post	Yes, can switch to point releases by deleting the <code>.git</code> directory in FreeField
Handling breaking changes	Breaking changes listed in changelog visible before installing update	Breaking changes listed on Releases page on GitHub only up to latest point release
Dependency and prerequisite checks	Checked automatically before updates are applied; update halted if necessary	Checked only on initial setup, not for updates

1.3 Setting up the web server

1.3.1 Apache

FreeField is pre-configured for Apache and does not require additional configuration. Therefore, it is generally recommended that you install FreeField under Apache.

1.3.2 nginx

If you are setting up FreeField for nginx, it is **critical** that you restrict access to some directories that should not be publicly visible.

includes You **must** deny access for everyone to this directory. It contains files that are included from elsewhere, and accessing these scripts directly can be dangerous.

docs You are *recommended* to restrict access to this directory, as it only contains documentation in reStructuredText format. This is already readily available from <https://freefield.readthedocs.io/> in a more user-friendly format.

Please check that you are not able to access these directories from the browser before you continue setting up FreeField.

Danger: If you do not deny all direct access to `/includes` (via e.g. `deny all`), any user would be able to break your entire FreeField installation beyond repair, and might even compromise the security of your server. Files in this directory contains code that is only supposed to be executed in controlled circumstances. The behavior of this code in circumstances of direct execution has not been tested.

1.3.3 Other web servers

No documentation currently exists for other web servers. If you have set up FreeField on a web server that is not listed above, please share your experience on GitHub so we can add it to this documentation.

1.4 Installation wizard

Once you have downloaded and extracted the FreeField archive, you may navigate to the installation path using your web browser of choice. You will automatically be redirected to the installation wizard.

1.4.1 Stage 1: Environment checks

This stage of the installation checks that your installation environment is suitable for FreeField. Below are a list of performed checks, and steps you may perform to correct any issues with the environment.

Encrypted connection (HTTPS) Web browsers will restrict access to geolocation and service workers, among other things, if HTTPS is not enabled on the installation site. A lack of HTTPS on your site will result in users not being able to tap on the “my location” button on the map to locate and track their own location on the map. It will also not be possible to enable Progressive Web App functionality if HTTPS is disabled, as this depends on service workers, which do not work without HTTPS for security reasons.

Tip: If your hosting provider already offers HTTPS by default, you can try to simply load your site over HTTPS instead by changing your browser URL. Otherwise, you may have to enable HTTPS yourself. If you are running your own server, and you do not have HTTPS set up, you need to enable and configure HTTPS in your HTTP daemon’s configuration file, and allow connections to TCP port 443 (or whatever port you are running HTTPS over) through your firewall.

Tip: If you need a TLS certificate, you could use a service such as Let’s Encrypt to get one for free. For information on how to set up Let’s Encrypt, please see Let’s Encrypt’s [Getting Started guide](#).

Installation directory writable In order for FreeField to perform updates, it is highly recommended that you allow writing to FreeField’s installation directory. FreeField will still function without this permission, but you will not be able to install updates.

Hint: To allow writing, either change the owner of the installation directory to the user used by the HTTP daemon using e.g. `chown -R http:http .`, or change the file permission to allow global writes, i.e. `chmod -R a+w .`, in the installation directory.

Userdata directory writable FreeField stores its configuration files and some user-submitted data in the `/includes/userdata` folder. This folder must be writable by the HTTP daemon. If it is not writable, make it writable, either by changing the owner of the file, or by allowing global writes, as detailed in the above section.

cURL extension loaded cURL is used to download updates to FreeField, as well as performing user authentication. FreeField will not work without cURL. If this check fails, ensure that the PHP cURL extension is available on your system, and that it is enabled in `php.ini`.

gd extension loaded If FreeField is configured to require approval of newly registered users, the user approval requirements notice page displayed to the newly registered users can be configured to display QR codes that, if

scanned by an administrator, allows quickly approving the user. An approval link will be required in any case that the user can forward to an admin through some messaging service/private message somewhere.

Hint: QR codes and manual approval is explained in greater detail in [Manual user approval](#).

openssl extension loaded Cryptographic functions are used for various purposes in FreeField, and these functions are provided by OpenSSL. FreeField uses encryption for session cookies and sensitive data in the configuration files, as well as `openssl_random_pseudo_bytes()` for generating CSRF state tokens, session tokens and cryptographic keys. FreeField will not function without this extension. Ensure that it is installed and enabled in `php.ini`.

PharData available PharData is used to extract updates after they have been downloaded. FreeField will still function even if PharData for some reason isn't present, but updates will not be possible to install.

You should ensure that as many as possible of the above checks pass, as failing checks may limit the functionality of FreeField or completely prevent it from working - in the latter case, the installation wizard will not allow you to proceed with the installation. You should make the desired changes now, as some configuration defaults vary depending on the state of the checks. Apply the changes, restart the HTTP daemon for the changes to take effect, and then reload the installation wizard to ensure that the changes have been applied and that the checks are now passing.

1.4.2 Stage 2: Write the configuration file

This stage simply writes a configuration file with default values applied to the userdata directory. It also generates cryptographic keys for session data and sensitive configuration file entries. This step is automatic. The output from this step should be the following three checks:

- Copied file options from template files
- Secure storage encryption keys generated
- Configuration file written

If any of those entries are missing, along with the *Continue setup* button, then something has gone very wrong, and you should check your web server error logs.

1.4.3 Stage 3: Database setup

In this stage, you need to set up a connection from FreeField to your database backend. Choose your database provider from the list of available providers and enter the required connection details.

Hostname The hostname of the database server.

Hint: This is typically “localhost”, “127.0.0.1” or “::1” if MySQL is running on the same host as the web server. If you are using shared web hosting, please check your hosting provider’s settings panel for the hostname, as shared hosting providers often have dedicated SQL servers.

Port The port that your database runs on. In most cases, you can leave this to the default – 1 to let PDO use the default port for your given database type.

Username The username used to access the database server.

Password The password used to access the database server.

Database The database that you wish to store FreeField data in.

Table prefix All FreeField tables are prefixed with this string to separate it from other tables in the database. You have to specify a string here. The default prefix `ffield_` works in most cases, though if you are running multiple instances of FreeField in the same database, you must select a different table prefix for each instance, so the instances do not interfere with each other.

Caution: Only MySQL has been tested and is known to be stable with FreeField. Providers marked “experimental” have not been tested and may be unstable, not work at all, or spontaneously break in the future. Use these at your own risk.

Note: If you cannot find your database provider in the list, then you have most likely not enabled the PDO extension for your database backend in `php.ini`. For example, if you want to use MySQL, you must ensure that `extension=pdo_mysql` is defined and not commented out in `php.ini`. If you have enabled the extension, and the option still does not show up in the selection box, then FreeField may not support your database backend. If you wish for your database backend to be supported, you may create an issue for it on GitHub, but remember to search for existing related issues first, as others may have requested it before you.

Note: If you use SQLite, please fill in the path to the SQLite database in the “Database” field, and fill in dummy values in all other fields.

When you are ready, FreeField will connect to the tables and set up the necessary database table structure. If everything went according to plan, the following five entries should all be checked with green check marks:

Database details are valid If this fails, one or more provided settings may be empty or contain invalid characters. FreeField will not attempt to connect to the database if the database settings are invalid.

Configuration file updated If this fails, then FreeField was not able to write the configuration file in the userdata directory. The userdata directory must be permanently writable in order for FreeField to function.

Connected to database If this fails, FreeField was not able to establish a connection to the database. Please read the accompanying error message for more details, or consult the troubleshooting section below for help resolving common mistakes.

Created database structure If this fails, FreeField was able to establish a connection to the database, but could not run the SQL queries necessary to set up the FreeField tables. Please read the accompanying error message for more details, or consult the troubleshooting section below for help resolving some common mistakes.

Stage 3 registered complete This step saves the progress of the installation wizard to the configuration file. If this step fails, something is seriously wrong with your server, as it means the configuration file became unwritable somewhere during the database connection process. This should never happen under any circumstances.

Troubleshooting

SQLSTATE[HY000] [1044] The authentication credentials were correct, but the database could not be connected to. Check that you did not mistype the name of the database, that the database actually exists, and that the given user has permission to access and modify it.

SQLSTATE[HY000] [1045] The provided database credentials were incorrect. Double-check the username and password you defined.

SQLSTATE[42S01] You have already set up FreeField before with these details. You can install this FreeField instance side-by-side with the other instance in the same database by changing the table prefix to some other value than the default.

1.4.4 Stage 4: Authentication setup

In this stage, you will be setting up authentication on FreeField. You have to set up at least one authentication provider and demonstrate that you are able to sign in to it in order to proceed to the next step. Please consult the [Authentication](#) docs for help setting up authentication with your preferred authentication provider. Once you are done setting up authentication, you will be prompted to sign in using one of the providers you set up.

All of the following checks must pass in order to continue to the next step:

Provided authentication details are valid If this fails, then there is an invalid value in your authentication setup. Please ensure that you have correctly inserted the required values for your authentication provider according to the [Authentication](#) docs.

Configuration file written If this fails, then FreeField failed to update the configuration file with the authentication provider settings you provided. Ensure that the userdata folder remains permanently writable.

At least one authentication provider is enabled If this fails then you have either not enabled any of the authentication providers on the previous page using the “Enable” checkboxes, or you have enabled one or more, but there is missing information for all of them (e.g. you have enabled an authentication provider, but not provided required details, such as a client ID and/or secret). Ensure that all fields are filled in, and the “Enable” checkbox ticked off, for at least one authentication provider, then try again.

Prepared authentication challenge If this fails, then something is seriously wrong with your server. It would indicate that within milliseconds of the configuration file being written above, someone or something prevented the configuration file from being written to again. This should never fail under any reasonable circumstances.

When you have configured an authentication provider, and all checks pass, you can proceed to sign in using the authentication provider you set up.

1.4.5 Stage 5: Verify authentication setup

You are automatically redirected to this stage when you click *Continue setup* in stage 4, and the authentication challenge is part of this step. Sign in using any available authentication provider.

Hint: If you for some reason cannot sign in using a provider, you can at any time click on *Reconfigure* to return to stage 4 and attempt to set up the authentication providers again. You may want to consult the [Authentication](#) docs to ensure that authentication is set up properly.

When you have signed in, you should return to the installation wizard, and all of the following checks must pass:

Authentication successful If you can see this check, then you have already successfully authenticated. This check cannot fail.

Registered account as site administrator in database This is handled by the FreeField authentication module, not the setup wizard. If you can see this check, then you have already been added to the database. This check cannot fail.

Configuration file updated If this step fails, the userdata folder (or the configuration file within) is no longer writable. The userdata folder and all contents must remain permanently writable for FreeField to function.

1.4.6 Stage 6: Map setup

In this step, you have to set up map settings to use with FreeField. You have to choose a map provider and set it up, along with map defaults. Please consult the [Map settings](#) docs for more information on how to configure map providers.

In addition to selecting a map provider, you have to specify the default starting coordinates for FreeField. The coordinates you choose are the ones that the map will be centered on when you first launch FreeField.

Tip: It is a very good idea to pick the coordinates of a centrally located and/or easily recognizable location in the town/city you are setting up FreeField for. The default 0, 0 location is **not a good location** to center the map.

When you are done with stage 6, FreeField will write the map provider settings to the configuration file. The following checks should pass:

Provided map settings are valid If this fails, there is an error in the settings you entered. Ensure that the map provider details are set up as described in the [Map settings](#) docs, and that the defaults map location you have selected are valid coordinates.

Configuration file updated If this fails, then FreeField was unable to save the settings you just entered. The most likely cause for this is that the configuration file is not writable. The userdata directory and its contents must remain permanently writable in order for FreeField to function properly.

1.5 Post-installation steps

If all these checks passed, you have successfully completed the installation wizard and set up FreeField for use. Before you grant others access to the map, you should set up additional settings such as [Geofencing](#), [Permissions and groups](#), [Site appearance](#), and then add the [Pokéstops](#) in your area to the map.

Warning: By default, FreeField allows submission of Pokéstops to the map anywhere in the world. To prevent your map from being leached by users elsewhere, it is strongly recommended that you set up a geofence after installation that restricts the area in which Pokéstops and field research can be submitted. For more information on how to do this, refer to [Geofencing](#).

FreeField uses third party OAuth2 providers for authenticating users. This allows flexibility with letting users log in to FreeField using an existing account on the same service that your community uses for communication, and also eliminates inherent problems associated with storing emails and passwords in a database.

You are required to set up at least one authentication provider in FreeField. However, you are free to set up several providers if you want to offer greater flexibility for users, or if your community uses several social media platforms. Instructions for setting up each supported authentication provider is provided on the pages below.

2.1 Discord authentication

In order to set up Discord authentication, you need to register an application on Discord. This can be done from the Discord website.

2.1.1 Registering an application

1. Go to <https://discordapp.com/developers/applications/> and log in with your Discord account.
2. Click on “Create an application”.
3. Give your application a name and description.

Tip: You should choose a name that reflects the community you have set up FreeField for. A good idea is to use the name of your community, or a location-specific name such as “New York FreeField.” You may optionally upload an icon for your application, which will be displayed when users attempt to authenticate.

4. Take note of the “Client ID” and “Client Secret” assigned to your application:

APP ICON

NAME

FreeField

CLIENT ID

495624147641303056

Copy

CLIENT SECRET

Click to reveal

Copy Regenerate

DESCRIPTION (MAXIMUM 400 CHARACTERS)

Grants access to the FreeField map.

- On the same page, look for “OAuth2” in the sidebar menu. Click on it to navigate to the authentication options.
- In the “Redirects” section, add a new redirect and paste the redirect URL for FreeField’s implementation of OAuth2 with Discord. This URL is `auth/oa2/discord.php`, relative to your installation path. E.g. if you have installed FreeField to `https://example.com/freefield/`, the redirect URL would be `https://example.com/freefield/auth/oa2/discord.php`.

REDIRECTS

You must specify at least one URI for authentication to work. If you pass a URI in an OAuth request, it must exactly match one of the URIs you enter here.

`https://staging.varden.info/auth/oa2/discord.php` ×

Add Another

Careful — you have unsaved changes!

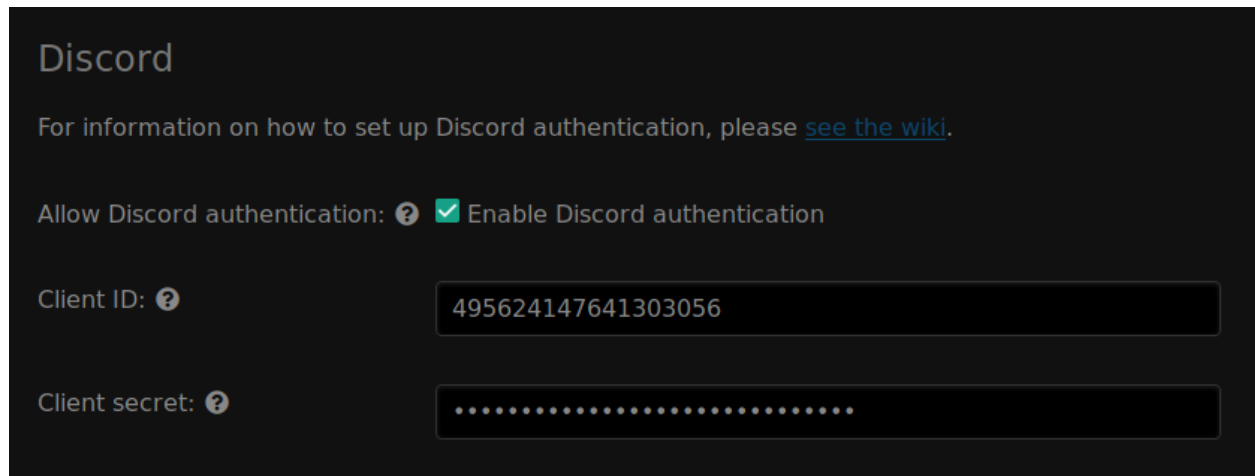
Reset Save Changes

- Ensure that you click the “Save changes” button at the bottom of the page (see the above screenshot). It is recommended that you reload the page after saving changes to ensure that the redirect URL was added.

2.1.2 Enabling Discord authentication in FreeField

After you have registered an application on Discord, you can configure FreeField to use Discord for authentication.

- In the FreeField administration pages, navigate to the “Authentication” menu.
- In the Discord section, check the box next to “Enable Discord authentication” and paste the client ID and secret you got from Discord in the relevant fields.

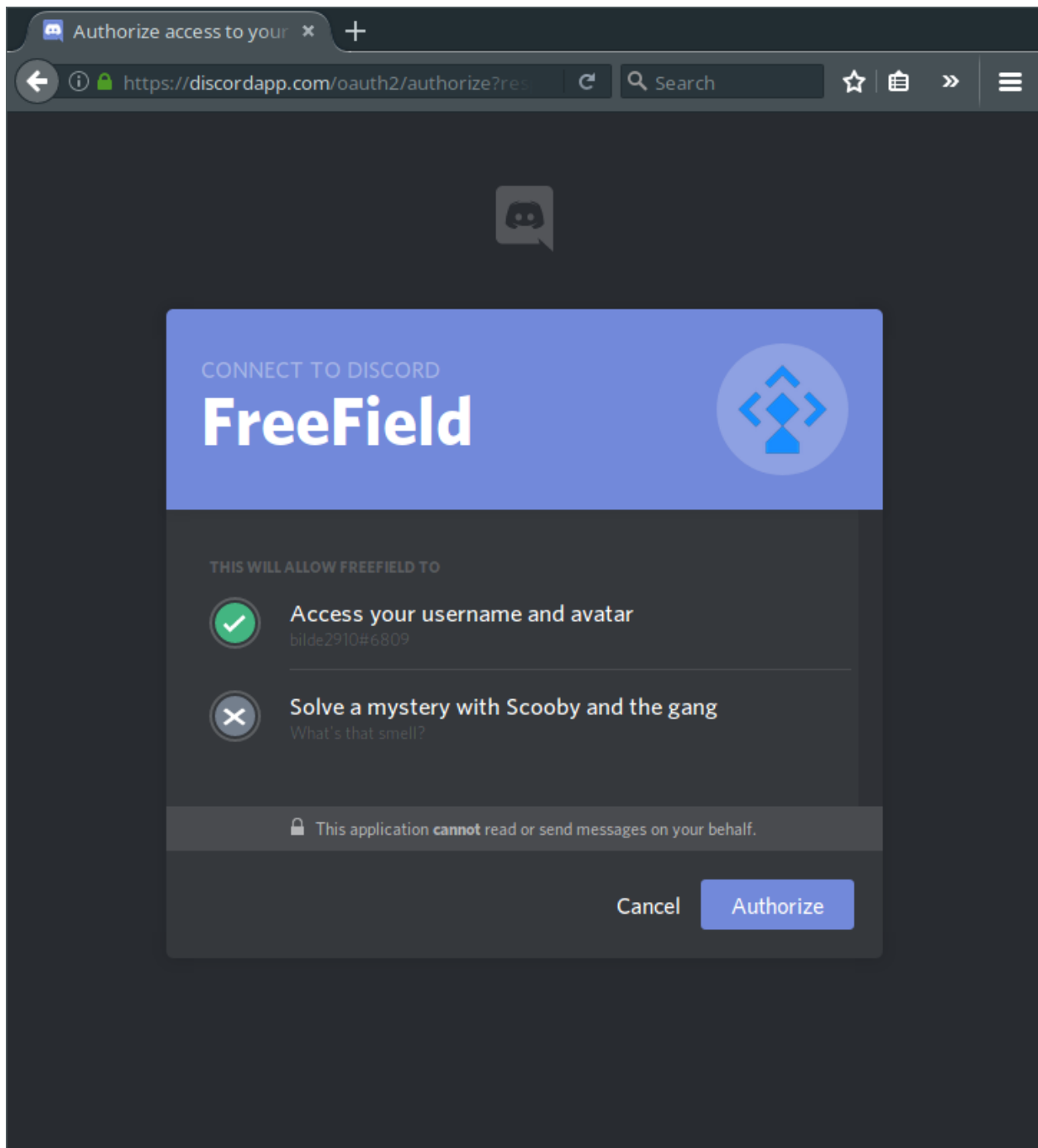


The screenshot shows a dark-themed interface for Discord authentication settings. At the top, the word "Discord" is displayed in a large, light-colored font. Below it, a line of text reads: "For information on how to set up Discord authentication, please [see the wiki](#)." Further down, there is a toggle switch labeled "Allow Discord authentication:" followed by a question mark icon and a green checkmark icon, with the text "Enable Discord authentication" to its right. Below this, there are two input fields. The first is labeled "Client ID:" with a question mark icon, and its value is "495624147641303056". The second is labeled "Client secret:" with a question mark icon, and its value is represented by a series of dots, indicating it is a masked secret.

3. Save the setting using “Save settings” at the bottom of the page.

2.1.3 Authentication preview

When users authenticate with FreeField through Discord, they will see an authentication prompt similar to this:

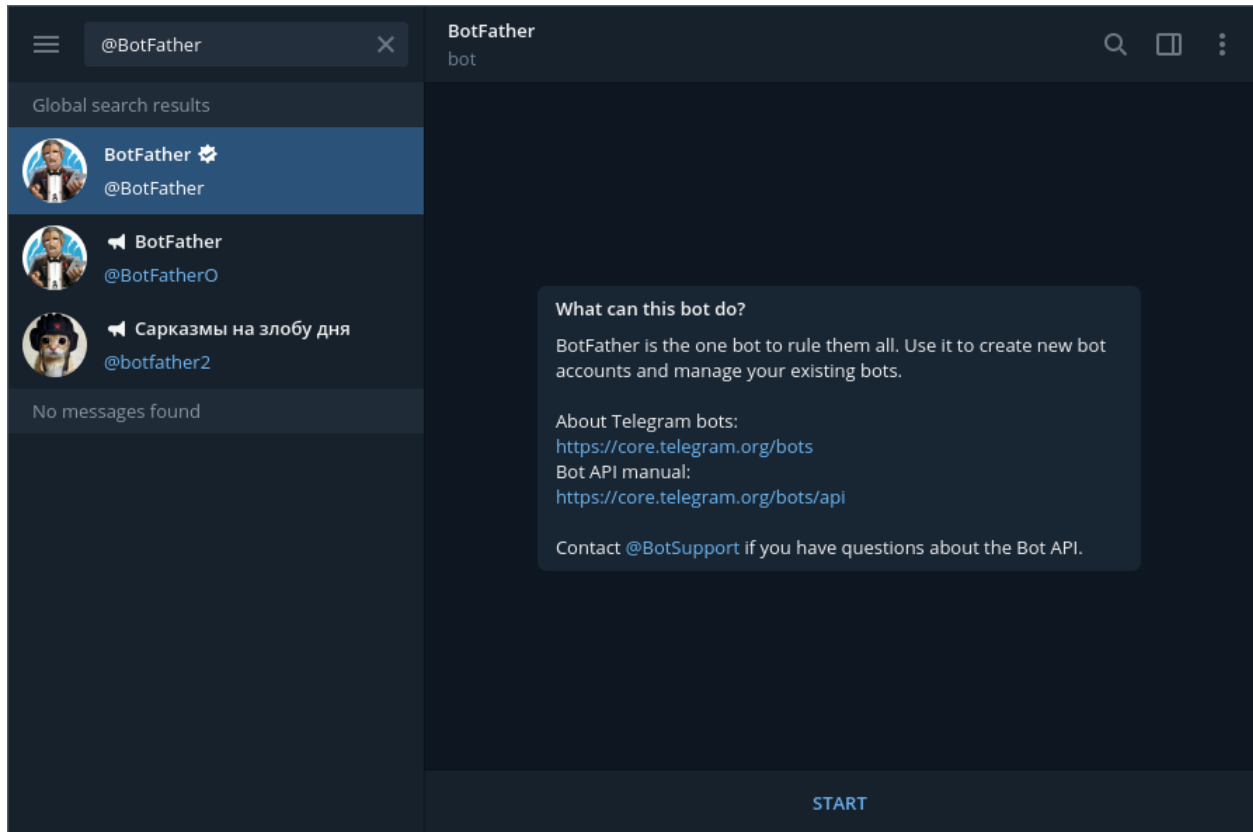


2.2 Telegram authentication

In order to set up Telegram authentication, you need to register a bot on Telegram. This can be done using @BotFather.

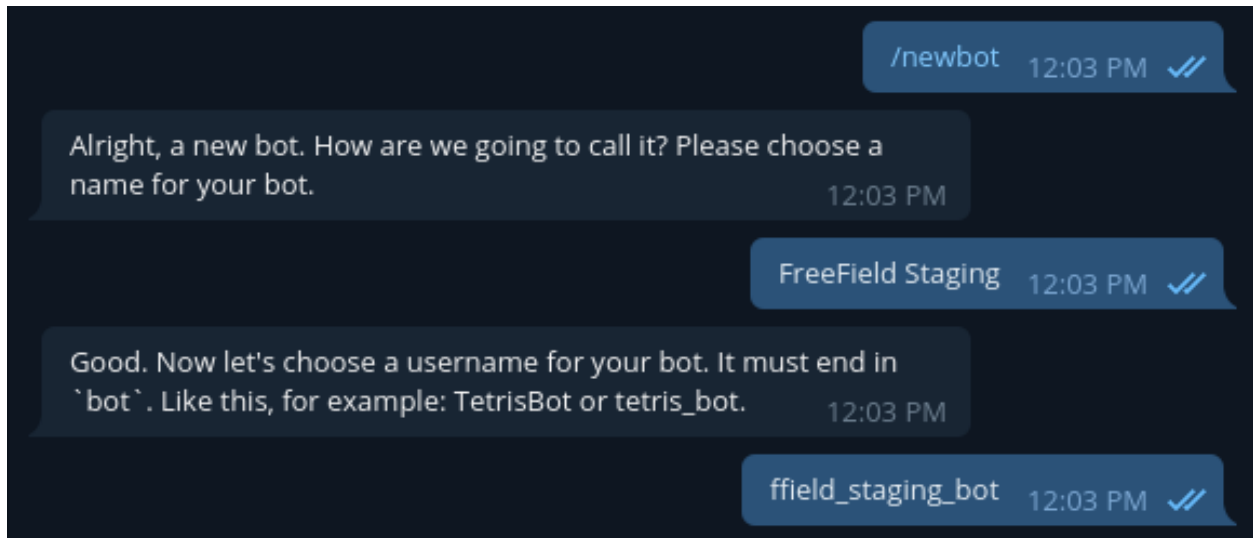
2.2.1 Registering a bot

1. Search for the user @BotFather on Telegram and open a conversation with this bot, or [click here](#) to open a conversation directly. Click on the *Start* button at the bottom of the Telegram interface.



2. Issue the `/newbot` command in chat.
3. @BotFather will ask for a display name and username of your Telegram bot. Enter a display name and username. The display name can be anything, though the username must end with “bot”.

Tip: You should choose a name that reflects the community you have set up FreeField for. A good idea is to use the name of your community, or a location-specific name such as “New York FreeField.” You may optionally upload an icon for your application, which will be displayed when users attempt to authenticate.

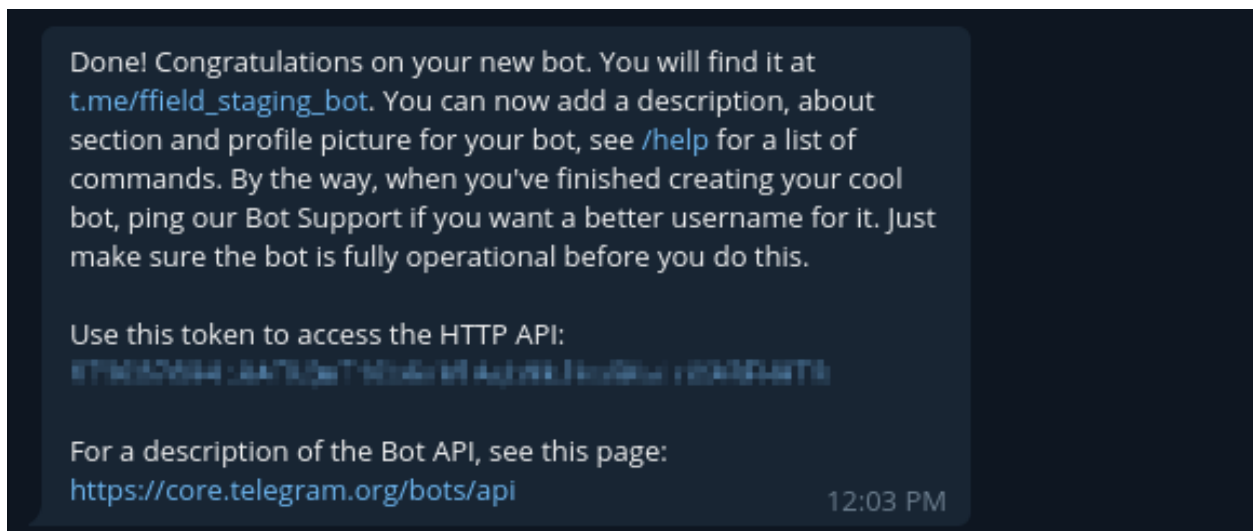


4. You should now be assigned a bot token for your bot by @BotFather. Store this token safely.

Danger: Your bot token is very sensitive information. Anyone who obtains a copy of this token will be able to impersonate Telegram and can authenticate themselves as any Telegram user on FreeField. If you or a high level administrator uses Telegram to authenticate with FreeField, it is particularly important that this token is never, ever shared with anyone else, not even with other administrators. In such a case, users who have the bot token would be able to successfully impersonate an administrator and either elevate their own privileges to a higher level, or use the administrator account directly to disrupt, destroy or even seize total, exclusive control over FreeField.

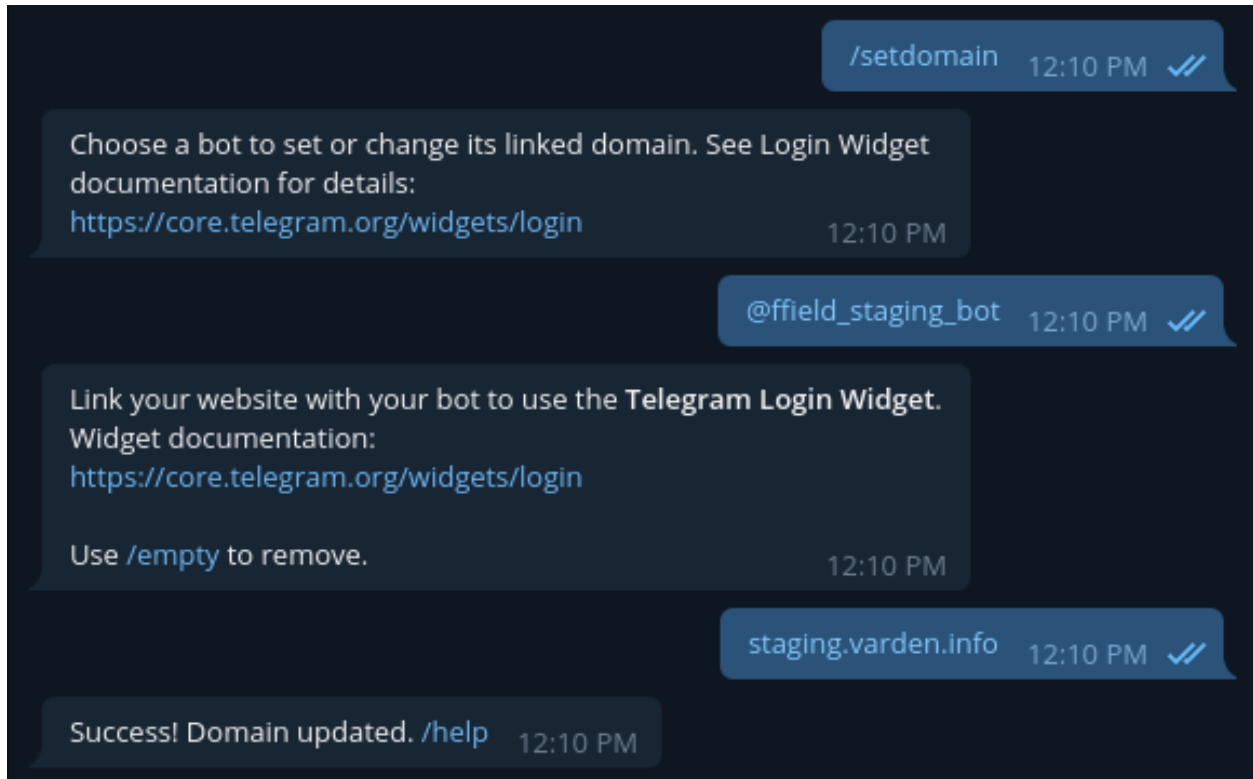
If you ever suspect that the bot token has been inadvertently shared with any other person, **immediately** shut down all access to FreeField and revoke and reissue a new bot token through @BotFather.

There is no technical way to fix this attack vector.



5. Issue the `/setdomain` command.
6. If you have several Telegram bots, you may be prompted to choose one of them to configure the domain for. Select the bot that you just created.

7. Enter the host name of your FreeField site. E.g. if you have installed FreeField to `https://example.com/freefield/`, enter `example.com`. If the domain was updated successfully, you should see the message “Success! Domain updated.”

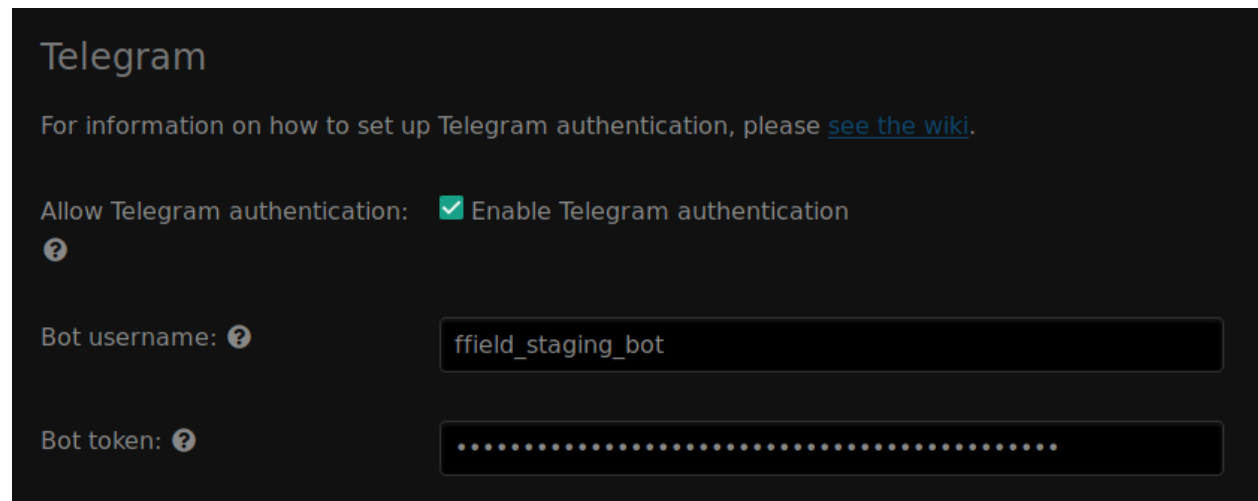


Caution: By default, your bot can be added to any Telegram group. It is a good idea to disable group joining, as this is not necessary to authenticate users. To do so, issue the `/setjoingroups` command to `@BotFather` and follow the provided instructions.

2.2.2 Enabling Telegram authentication in FreeField

After you have registered an bot on Telegram, you can configure FreeField to use Telegram for authentication.

1. In the FreeField administration pages, navigate to the “Authentication” menu.
2. In the Telegram section, check the box next to “Enable Telegram authentication” and enter the username you assigned to your bot, as well as your assigned bot token, in the relevant fields.

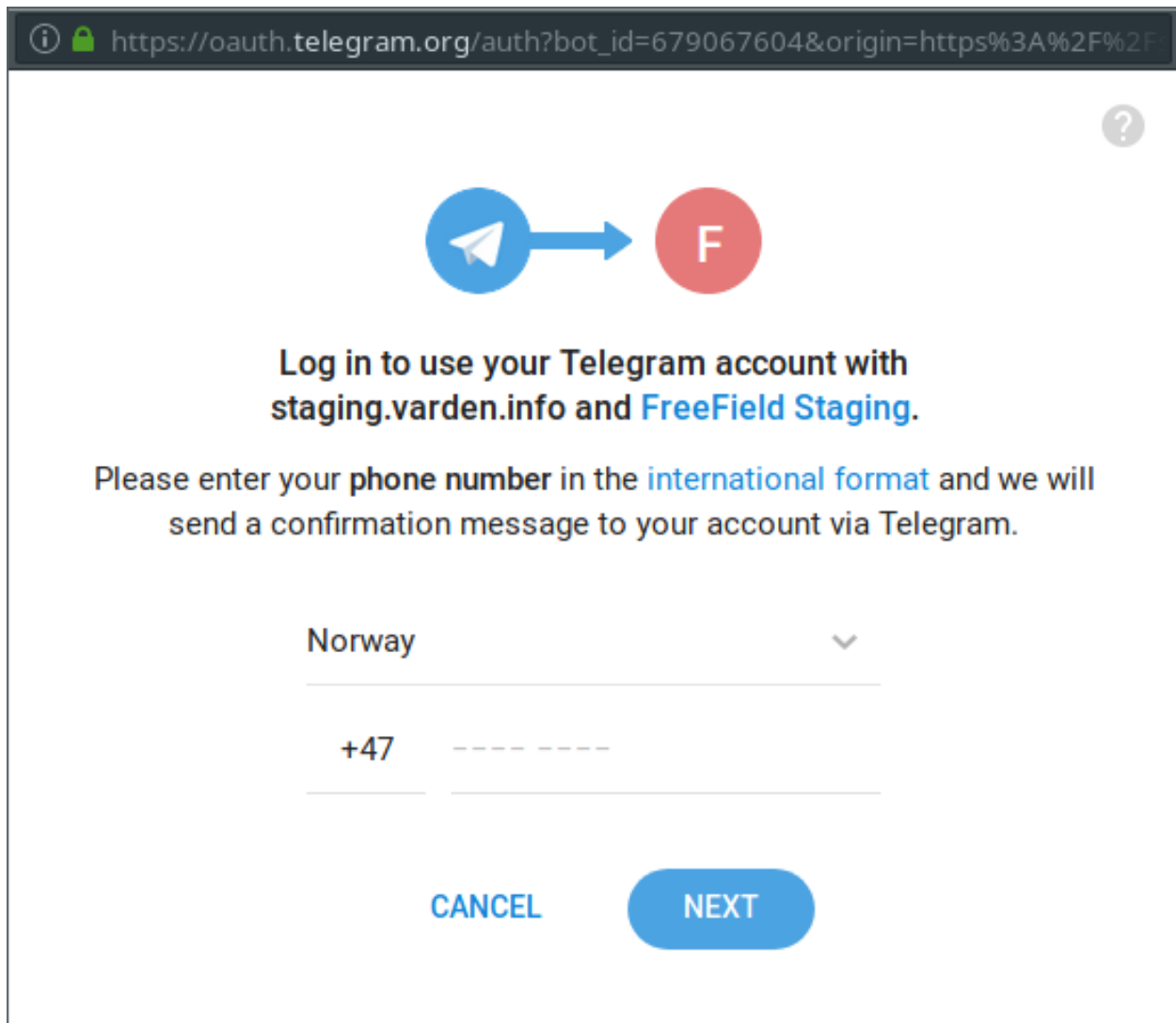


The screenshot shows a dark-themed interface for Telegram authentication settings. At the top, the word "Telegram" is displayed in a large, light-colored font. Below it, a line of text reads: "For information on how to set up Telegram authentication, please [see the wiki](#)." Further down, there is a label "Allow Telegram authentication:" followed by a green checkmark icon and the text "Enable Telegram authentication". Below this is a small question mark icon. The next section has a label "Bot username:" followed by a question mark icon and a text input field containing the value "ffield_staging_bot". The final section has a label "Bot token:" followed by a question mark icon and a text input field filled with a series of dots, indicating a masked token.

3. Save the setting using “Save settings” at the bottom of the page.

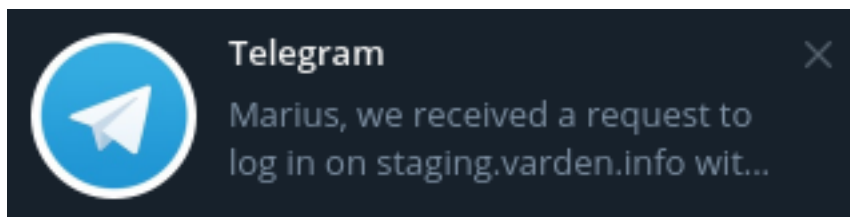
2.2.3 Authentication preview

When users authenticate with FreeField through Telegram, they will see an authentication prompt in Telegram similar to this:

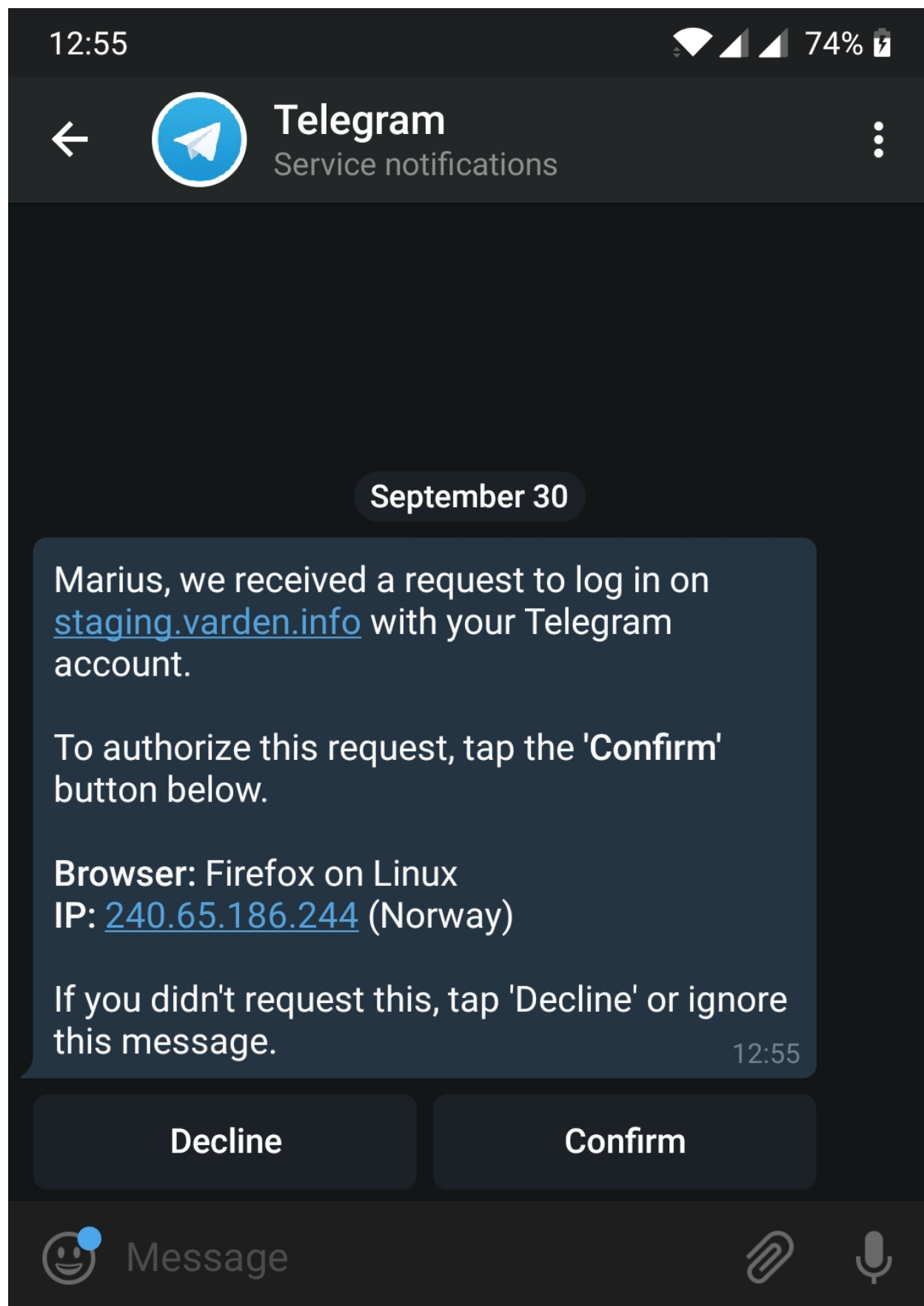


The image shows a web browser window with the URL `https://oauth.telegram.org/auth?bot_id=679067604&origin=https%3A%2F%2F`. The page features a Telegram logo (blue circle with a white paper plane) and a red circle with a white 'F' connected by a blue arrow. Below this, the text reads: "Log in to use your Telegram account with staging.varden.info and FreeField Staging." followed by "Please enter your phone number in the international format and we will send a confirmation message to your account via Telegram." There is a dropdown menu showing "Norway" with a downward arrow. Below the dropdown is a text input field with the placeholder "+47" and a dashed line. At the bottom, there are two buttons: "CANCEL" and "NEXT".

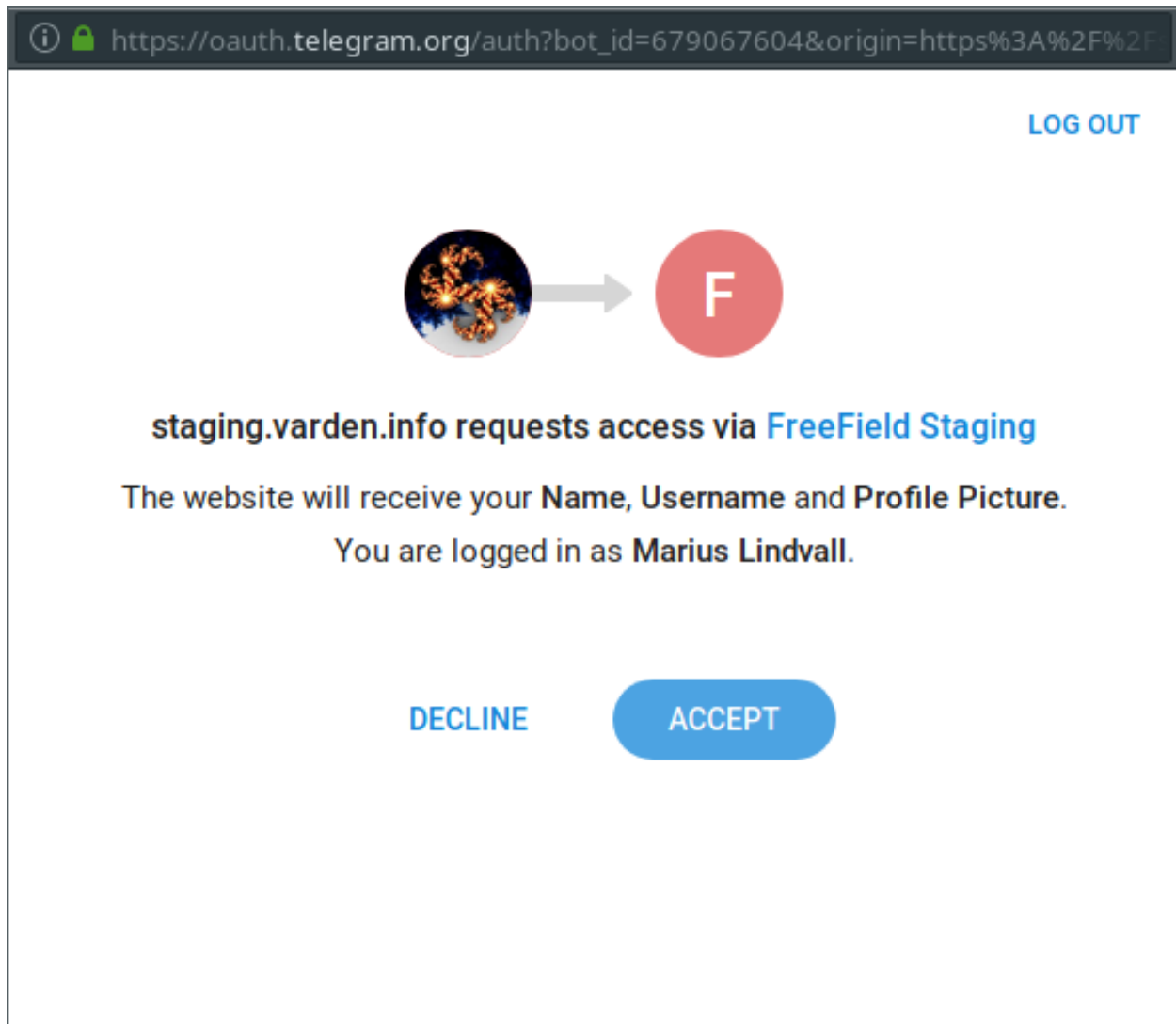
Users would then have to enter the phone number associated with their Telegram account. After clicking “Next”, they will get a notification to approve the authentication request:



Upon opening the notification, they will be prompted to confirm the sign-in attempt to their Telegram account:



Finally, upon confirmation, they will be prompted to grant FreeField access to their account:



When they click Accept on this prompt, Telegram will authorize the sign-in attempt to FreeField so that FreeField can authenticate the user.

2.3 Reddit authentication

In order to set up Reddit authentication, you need to register an app on Reddit. This can be done from the Reddit website.

2.3.1 Registering an app

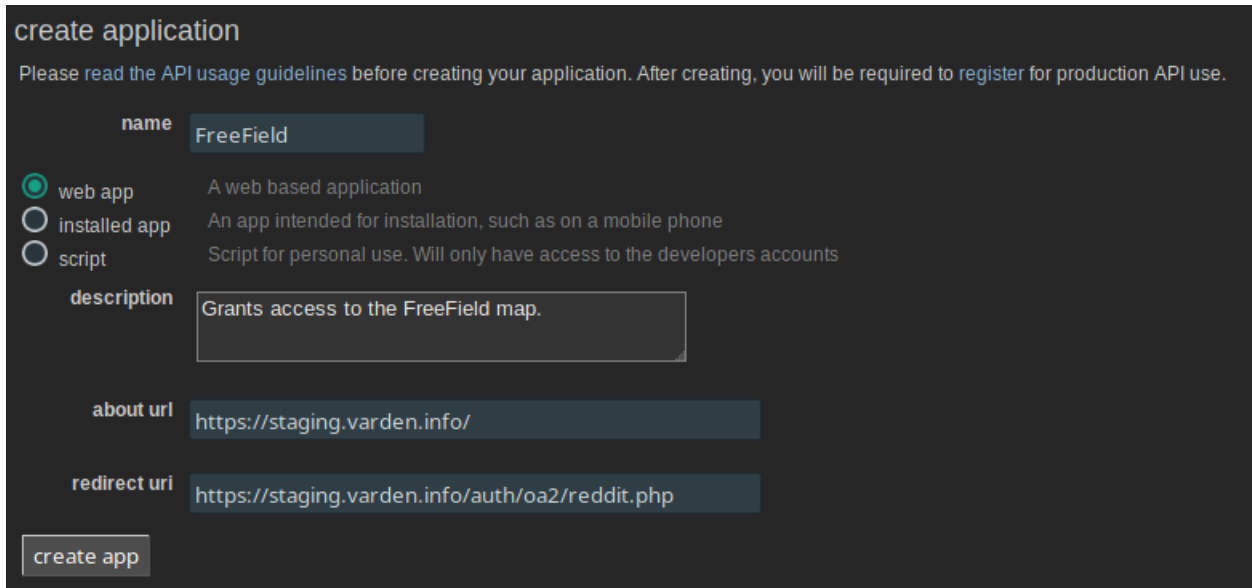
Please note that Reddit may require you to agree to API usage guidelines before you create your application. This is detailed on the applications creation page as well as on the [Reddit wiki](#).

1. Log in on Reddit, then go to <https://www.reddit.com/prefs/apps>.
2. Scroll to the bottom of the page and click on “are you a developer? create an app...”

3. Select “web app” as your app type.
4. Give your application a name and description.

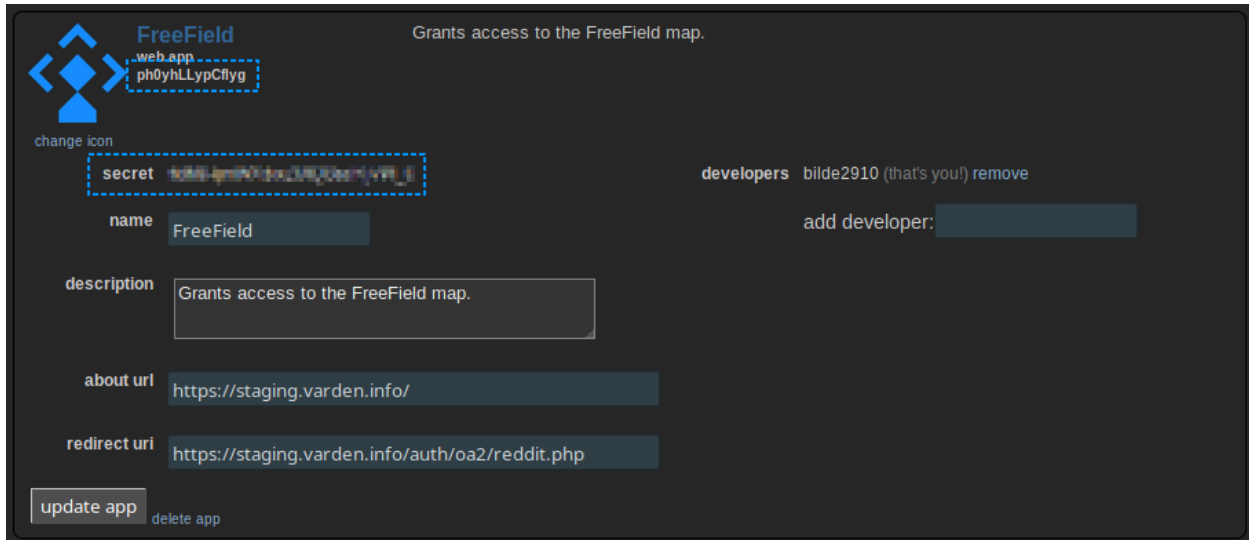
Tip: You should choose a name that reflects the community you have set up FreeField for. A good idea is to use the name of your community, or a location-specific name such as “New York FreeField.” You may optionally upload an icon for your application, which will be displayed when users attempt to authenticate.

5. In the “about url” field, enter the base URL of your FreeField installation, e.g. `https://example.com/freefield/`.
6. In the “redirect uri” field, paste the redirect URL for FreeField’s implementation of OAuth2 with Reddit. This URL is `auth/oa2/reddit.php`, relative to your installation path. E.g. if you have installed FreeField to `https://example.com/freefield/`, the redirect URL would be `https://example.com/freefield/auth/oa2/reddit.php`.
7. Click on “create app”.



The screenshot shows a web form titled "create application" on a dark background. Below the title is a note: "Please read the API usage guidelines before creating your application. After creating, you will be required to register for production API use." The form contains several fields: "name" with the value "FreeField"; a radio button selection for app type with "web app" selected (others are "installed app" and "script"); a "description" text area with the value "Grants access to the FreeField map."; an "about url" text field with the value "https://staging.varden.info/"; and a "redirect uri" text field with the value "https://staging.varden.info/auth/oa2/reddit.php". At the bottom left is a "create app" button.

8. Take note of the ID and secret assigned to your application.
9. You may optionally upload an icon for your application, which will be displayed when users attempt to authenticate.

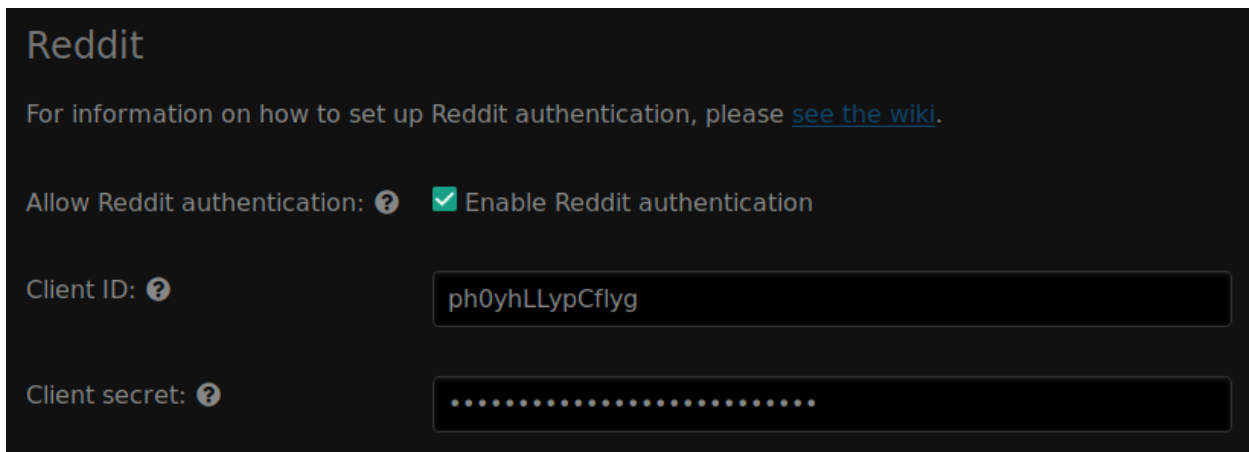


The image shows a web interface for configuring a FreeField app. At the top left is the FreeField logo, a blue diamond shape with four arrows pointing outwards. To its right is the text "FreeField". Below the logo is a "change icon" link. The main heading is "Grants access to the FreeField map." Below this is a "secret" field with a value "ph0yhLLypCflyg" and a "web.app" label. To the right of the secret field is a "developers" section showing "bilde2910 (that's you!)" with a "remove" link. Below the secret field is a "name" field with the value "FreeField" and an "add developer:" link. Below the name field is a "description" field with the value "Grants access to the FreeField map." Below the description field is an "about url" field with the value "https://staging.varden.info/". Below the about url field is a "redirect url" field with the value "https://staging.varden.info/auth/oa2/reddit.php". At the bottom left are "update app" and "delete app" buttons.

2.3.2 Enabling Reddit authentication in FreeField

After you have registered an app on Reddit, you can configure FreeField to use Reddit for authentication.

1. In the FreeField administration pages, navigate to the “Authentication” menu.
2. In the Reddit section, check the box next to “Enable Reddit authentication” and paste the client ID and secret you got from Reddit in the relevant fields.

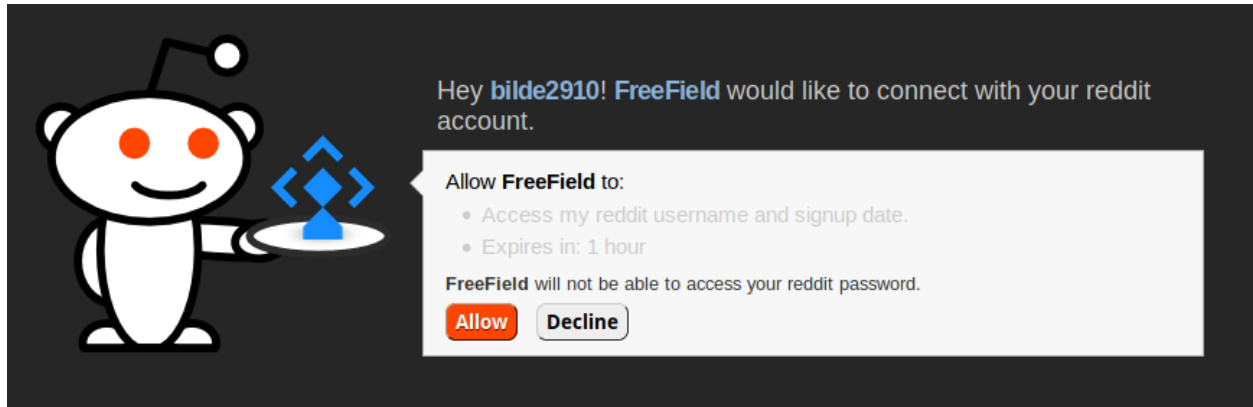


The image shows a web interface for configuring Reddit authentication. The heading is "Reddit". Below the heading is a link "For information on how to set up Reddit authentication, please [see the wiki](#)." Below this is a section "Allow Reddit authentication:" with a question mark icon and a checked checkbox "Enable Reddit authentication". Below this is a "Client ID:" field with a question mark icon and a text input field containing "ph0yhLLypCflyg". Below the Client ID field is a "Client secret:" field with a question mark icon and a password input field with 16 dots.

3. Save the setting using “Save settings” at the bottom of the page.

2.3.3 Authentication preview

When users authenticate with FreeField through Reddit, they will see an authentication prompt similar to this:



2.4 Facebook authentication

In order to set up Discord authentication, you need to register an application on Discord. This can be done from the Discord website.

2.4.1 Registering an application

1. Go to <https://developers.facebook.com/> and log in with your Facebook account.
2. Under the “My Apps” menu, click “Create New App.”
3. Give your application a name and provide your contact email address.

Tip: You should choose a name that reflects the community you have set up FreeField for. A good idea is to use the name of your community, or a location-specific name such as “New York FreeField.” You may optionally upload an icon for your application, which will be displayed when users attempt to authenticate.

4. Click on *Create App ID*. If you are prompted to provide human verification, do so and click *Confirm*.

changes in app status including when the app is made public or deleted.

Create a New App ID

Get started integrating Facebook into your app or website

Display Name

FreeField Staging

Contact Email

youremail@example.com

By proceeding, you agree to the [Facebook Platform Policies](#)





Cancel Create App ID

5. When prompted to select a Scenario, select “Integrate Facebook Login” and click *Confirm*.

Select a Scenario

Select one of the following scenarios to get product-specific help content as you build your app. If you already have your project mapped out and are ready to build, feel free to skip this step.

Examples

<input type="checkbox"/>	 Implement Marketing API Get programmatic access to the Facebooks ads platform to automate ads management, create data-based audiences and more.	<ul style="list-style-type: none"> Target audiences strategically by automatically creating different ads permutations Manage and optimize ads in real time with rules-based ads management
<input type="checkbox"/>	 Get Started with the Ads Insights API Get programmatic access to Facebooks Ads Insights.	<ul style="list-style-type: none"> Provides a single, consistent interface to retrieve ad statistics
<input checked="" type="checkbox"/>	 Integrate Facebook Login A secure, fast and convenient way for people to create accounts and log into your app across multiple platforms.	<ul style="list-style-type: none"> Create accounts without having to set a password Personalize peoples' in-app experiences
<input type="checkbox"/>	 Get Started with the Pages API With the Pages API people can update and manage Facebook Pages from your page-related app. People can publish content to Facebook or Messenger with a Page's identity.	<ul style="list-style-type: none"> Make a Pages management tool for customers or for your company Build apps so content creators and editors can easily publish as a Page

Skip Confirm

6. Take note of the “App ID” and “App Secret” assigned to your application:

App ID	App Secret
<input type="text" value="568036347048129"/>	<input type="password" value="••••••••"/> Show
Display Name	Namespace
<input type="text" value="FreeField Staging"/>	<input type="text"/>

- On the same page, look for “Facebook Login” in the sidebar menu. Click on it, then click on the “Settings” sub-option to navigate to the authentication options.
- In the “Valid OAuth Redirect URIs” field, paste the redirect URL for FreeField’s implementation of OAuth2 with Facebook. This URL is `auth/oa2/facebook.php`, relative to your installation path. E.g. if you have installed FreeField to `https://example.com/freefield/`, the redirect URL would be `https://example.com/freefield/auth/oa2/facebook.php`.
 - If your FreeField installation does not support HTTPS, make sure to disable “Enforce HTTPS” on this page as well.

Client OAuth Settings

☒ Yes
 Client OAuth Login
Enables the standard OAuth client token flow. Secure your application and prevent abuse by locking down which token redirect URIs are allowed with the options below. Disable globally if not used. [?]

☒ Yes
 Web OAuth Login
Enables web-based Client OAuth Login. [?]

☐ No
 Force Web OAuth Reauthentication
When on, prompts people to enter their Facebook password in order to log in on the web. [?]

☒ Yes
 Use Strict Mode for Redirect URIs
Only allow redirects that use the Facebook SDK or that exactly match the Valid OAuth Redirect URIs. Strongly recommended. [?]

☐ No
 Embedded Browser OAuth Login
Enable webview Redirect URIs for Client OAuth Login. [?]

☐ No
 Enforce HTTPS
Enforce the use of HTTPS for Redirect URIs and the JavaScript SDK. Strongly recommended. [?]

Valid OAuth Redirect URIs

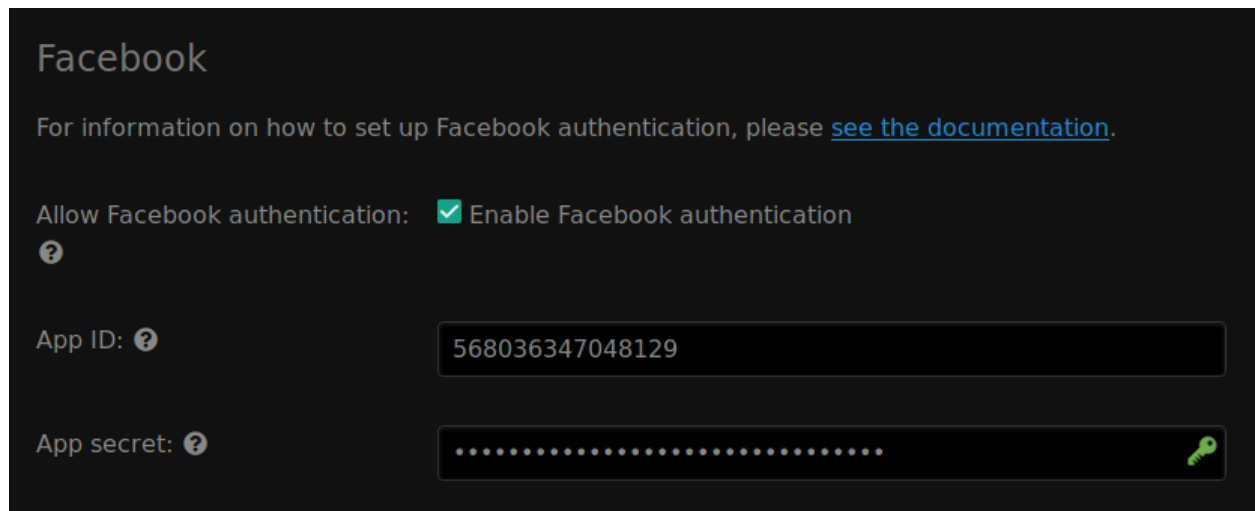
☐ No
 Login from Devices
Enables the OAuth client login flow for devices like a smart TV [?]

- Click “Save Changes” to save the settings.

2.4.2 Enabling Facebook authentication in FreeField

After you have registered an application on Facebook, you can configure FreeField to use Facebook for authentication.

- In the FreeField administration pages, navigate to the “Authentication” menu.
- In the Facebook section, check the box next to “Enable Facebook authentication” and paste the app ID and secret you got from Facebook in the relevant fields.

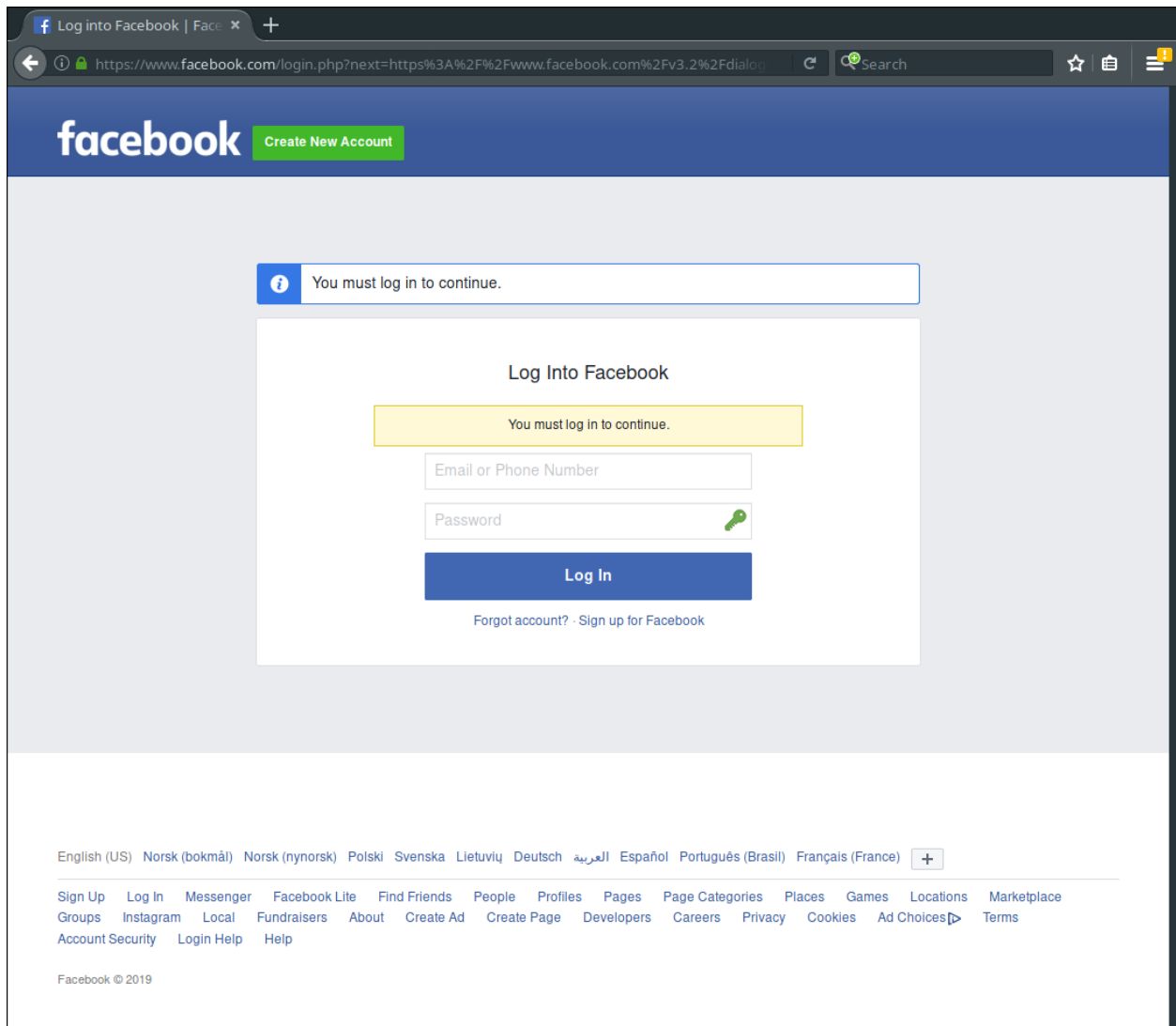


The screenshot shows a dark-themed settings panel for Facebook authentication. At the top, the word "Facebook" is displayed in a large, light-colored font. Below it, a line of text provides a link to documentation: "For information on how to set up Facebook authentication, please [see the documentation](#)." The main section is titled "Allow Facebook authentication:" followed by a checked checkbox and the text "Enable Facebook authentication". A small question mark icon is located below this text. Below the checkbox, there are two input fields. The first is labeled "App ID:" with a question mark icon, and it contains the text "568036347048129". The second is labeled "App secret:" with a question mark icon, and it contains a series of dots, indicating a masked password. A small key icon is visible at the end of the App secret field.

3. Save the setting using *Save settings* at the bottom of the page.

2.4.3 Authentication preview

When users authenticate with FreeField through Facebook, they will see an authentication prompt similar to this:



2.5 LINE authentication

In order to set up LINE authentication, you need to register an application on the LINE Developer Console. This can be done from the LINE Developers website.

2.5.1 Registering an application

1. Go to <https://developers.line.biz/console/> and log in with your LINE account.
2. Click on *Create new provider*.
3. Enter a provider name (your own name, or that of your organization).
4. Create a new channel under your provider, and select LINE Login as the channel type.
5. Give your application a name and description.

Tip: You should choose a name that reflects the community you have set up FreeField for. A good idea is to use the name of your community, or a location-specific name such as “New York FreeField.” You may optionally upload an icon for your application, which will be displayed when users attempt to authenticate.

6. Select “Use WEB” as the app type. Enter your email address and create the channel.
7. In your application’s settings, take note of the “Channel ID” and “Channel secret” assigned to your application:

Language (Other)

Channel ID ?

1629133960

Channel secret ?

App type ?

WEB

8. On the same page, look for and go to the “App settings” tab.
9. Under “Callback URL”, click *Edit* and paste the redirect URL for FreeField’s implementation of OAuth2 with LINE. This URL is `auth/oa2/line.php`, relative to your installation path. E.g. if you have installed FreeField to `https://example.com/freefield/`, the redirect URL would be `https://example.com/freefield/auth/oa2/line.php`.

Redirect settings

Set the URL for where the user is redirected after logging in.

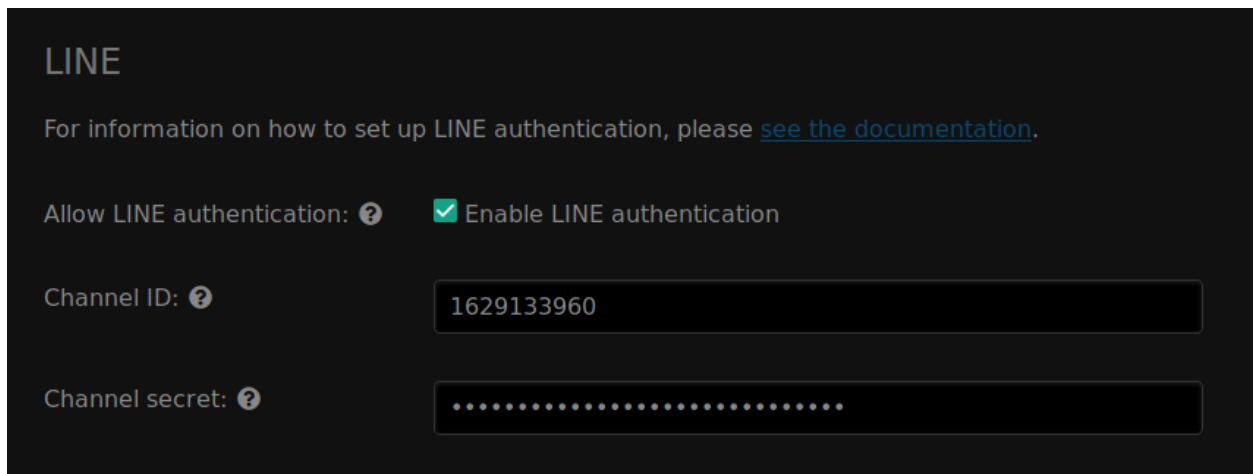
Callback URL

`https://staging.varden.info/auth/oa2/line.php`

2.5.2 Enabling LINE authentication in FreeField

After you have registered an application on LINE, you can configure FreeField to use LINE for authentication.

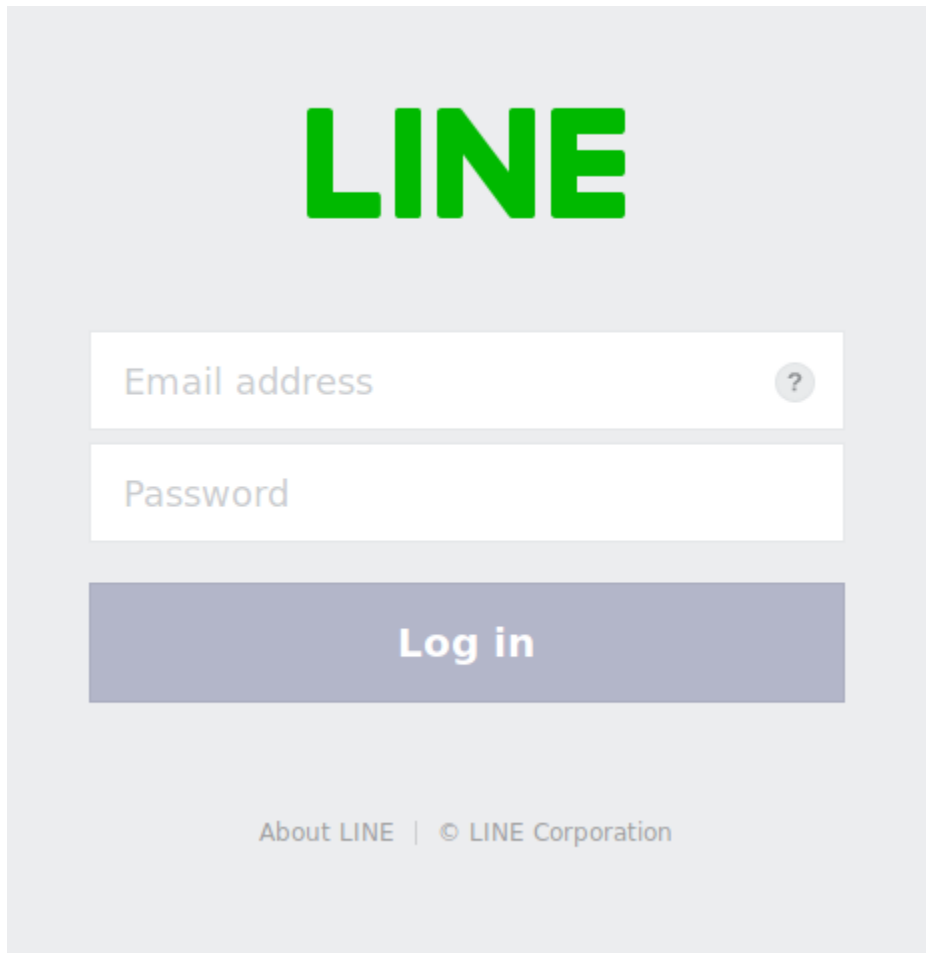
1. In the FreeField administration pages, navigate to the “Authentication” menu.
2. In the LINE section, check the box next to “Enable LINE authentication” and paste the channel ID and secret you got from LINE in the relevant fields.



3. Save the setting using “Save settings” at the bottom of the page.

2.5.3 Authentication preview

When users authenticate with FreeField through Discord, they will see an authentication prompt similar to this:



2.6 GroupMe authentication

In order to set up GroupMe authentication, you need to register an application on GroupMe. This can be done from the GroupMe website.

2.6.1 Registering an application

1. Go to <https://dev.groupme.com/applications/new> and log in with your GroupMe account.
2. Give your application a name.

Tip: You should choose a name that reflects the community you have set up FreeField for. A good idea is to use the name of your community, or a location-specific name such as “New York FreeField.” You may optionally upload an icon for your application, which will be displayed when users attempt to authenticate.

3. In the “Callback URL” box, paste the redirect URL for FreeField’s implementation of OAuth2 with GroupMe. This URL is `auth/oa2/groupme.php`, relative to your installation path. E.g. if you have installed FreeField to `https://example.com/freefield/`, the redirect URL would be `https://example.com/freefield/auth/oa2/groupme.php`.

Create Application

Application Name

Callback URL

Callback URL must be https, localhost, or a deep link.

Developer Name

Developer Email

Developer Phone Number

Developer Company

Developer Address

☐ I agree to abide by the [Terms of Use](#) and the [Brand Standards](#)

4. Fill in the rest of the form with your own contact details. Click “Save” when you are done.
5. Take note of the client ID assigned to your application. The client ID can be found at the end of the redirect URL now displayed on the page.

FreeField Staging

Details **Settings** Delete

Settings

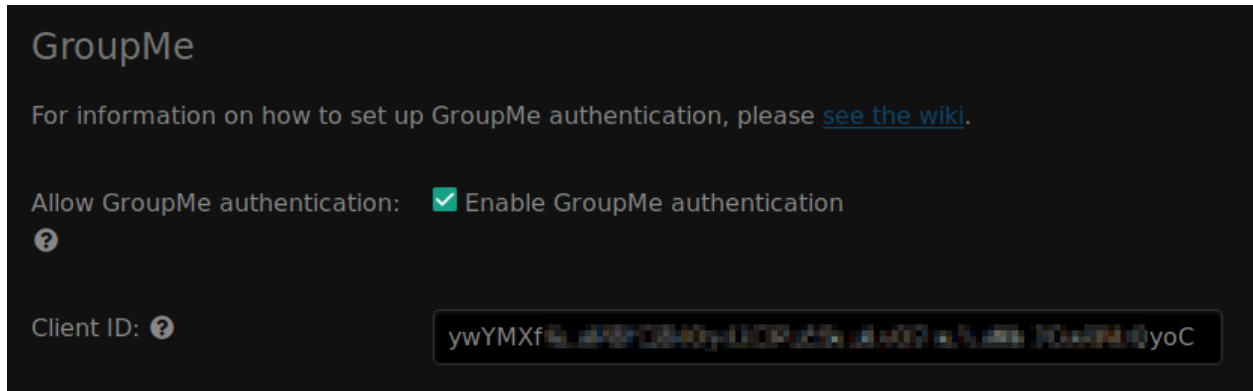
Redirect URL	https://oauth.groupme.com/oauth/authorize?client_id=ywYMXf...yoC
Callback URL	https://staging.varden.info/auth/oa2/groupme.php

Your Access Token

2.6.2 Enabling GroupMe authentication in FreeField

After you have registered an application on GroupMe, you can configure FreeField to use GroupMe for authentication.

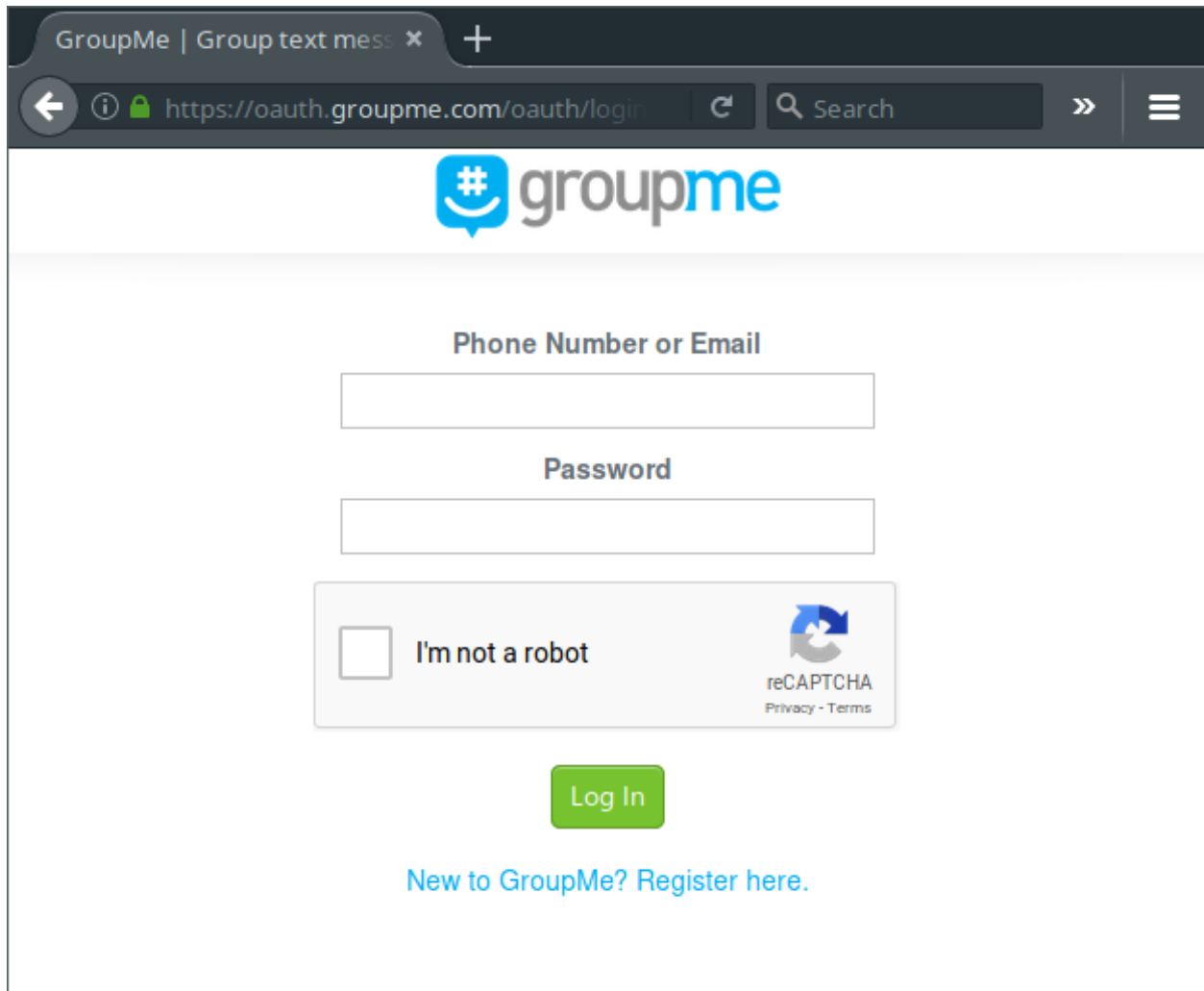
1. In the FreeField administration pages, navigate to the “Authentication” menu.
2. In the GroupMe section, check the box next to “Enable GroupMe authentication” and paste the client ID you got from GroupMe in the relevant field.

A screenshot of the 'GroupMe' settings page in a dark-themed interface. At the top, the title 'GroupMe' is displayed. Below it, a message says 'For information on how to set up GroupMe authentication, please [see the wiki](#).' The main section is titled 'Allow GroupMe authentication:' and features a green checkmark icon followed by the text 'Enable GroupMe authentication'. A small question mark icon is located below this text. At the bottom, there is a label 'Client ID: ?' and a text input field containing a long alphanumeric string: 'ywYMXf...yoC'.

3. Save the setting using “Save settings” at the bottom of the page.

2.6.3 Authentication preview

When users authenticate with FreeField through GroupMe, they will see an authentication prompt similar to this:



The image shows a web browser window with the GroupMe login page. The browser's address bar displays the URL `https://oauth.groupme.com/oauth/login`. The page features the GroupMe logo at the top, followed by a form with two input fields: "Phone Number or Email" and "Password". Below these fields is a reCAPTCHA section with a checkbox labeled "I'm not a robot" and a reCAPTCHA logo. A green "Log In" button is positioned below the reCAPTCHA. At the bottom of the form, there is a link that says "New to GroupMe? Register here."

When they log in, they will be redirected back to FreeField and be signed in automatically.

FreeField uses third party authentication providers to allow users to sign in. For information on how to set up different authentication providers, please see [Authentication](#). When any user registers on FreeField, the server checks if the user has logged into that particular FreeField instance previously, and if not, a new user record is created in the database. You can view and manage all registered users in the “Users” section of the administration pages.

3.1 User account details

The “Users” section on the administration pages contains tools needed to manage the accounts of all users registered on your FreeField instance. The page contains a table that lists all registered users, with the following fields:

3.1.1 Provider identity

Since users register by signing in with a third party authentication provider, FreeField will attempt to obtain a human-readable identity representing each user on the service they are registering through. For example, when signing in with Discord, FreeField stores the username and discriminator of the user and uses that to construct a provider identity string for the user (e.g. ExampleUser#0000). As another example, authentication with Reddit will store the account name of that user on Reddit (e.g. /u/ExampleUser). The purpose of this field is to make it easy for you as an administrator to understand who is behind each account on your instance of FreeField, in case a user misbehaves or otherwise causes problems on the map or elsewhere in your community. This allows you to contact the user in question should it be necessary.

Note: Keep in mind that on several provider platforms, the user can change their provider identity. While each FreeField user will always be internally linked to the correct user on the user’s authentication provider by means of a unique ID (such as an account number), the human-readable identity displayed in this field may not necessarily represent the correct user, or even an existing user, if they changed their provider identity since they last authenticated on FreeField. Every time a user authenticates (i.e. create a new session by signing in anew), the provider identity in the FreeField database is updated.

The provider identity will only be updated if the user actually signs in to FreeField, and you cannot force a user to authenticate. This means that there is no guaranteed way to ensure that the provider identity stored in the database

is always up to date. While you can take measures that restrict users' access to FreeField if they do not regularly authenticate, or invalidate their sessions, you should consider any ethical and privacy implications of doing so.

3.1.2 Provider

This field simply indicates which authentication platform the user has chosen to sign in on FreeField. If you have only enabled one authentication provider, only that provider will be listed here, but if you are using several providers, this field can help you differentiate between accounts that are using the same username, but on different services.

Note: If you want to disable an authentication provider for whatever reason, it is a good idea to check the user list and use this field to identify any users who may currently be using that authentication provider. If you disable a provider, none of those users will be able to sign in any longer, and would have to resort to other enabled authentication providers instead.

An existing user signing in on a new authentication provider will count as a new user registration, and you may have to reconfigure their permissions.

3.1.3 Nickname

Each user on FreeField is automatically assigned a nickname when they register, which reflects the name or identity of the account the user used to authenticate. The nickname appears in webhooks when users submit research to Pokéstops, and is generally considered public information. Users can change their own nicknames if they have been given the “Change own nickname” permission (see [Permissions and groups](#)). You can also change a user's nickname by editing it in the relevant box on the “Users” section on the administration pages.

3.1.4 Group

Each user on FreeField is assigned to a group. The group functions as a permission tier, and changing the group that a user is a member of will result in that user being granted or revoked permissions on FreeField.

Hint: The concept of groups is explained in greater detail on the [Permissions and groups](#) page.

3.1.5 Actions

The “Users” section on the administration pages allows administrators to perform various actions on individual registered users, such as deleting their account. Actions can be performed on several accounts at once through selecting an action for several users in the list, which will then be applied all at once when clicking on *Save settings*. The available actions for registered users are as follows:

Delete account This action will, if selected, delete the user's account from the FreeField users database.

Note: This will not delete references to the user in other locations, such as the Pokéstops database - if the user has submitted a Pokéstop on the map, or was the last person to report research for some Pokéstop, then the account ID of the user will still be stored in the Pokéstops database, although the list in the Pokéstops section on the administration pages will display “<DeletedUser>” rather than the username of the deleted user.

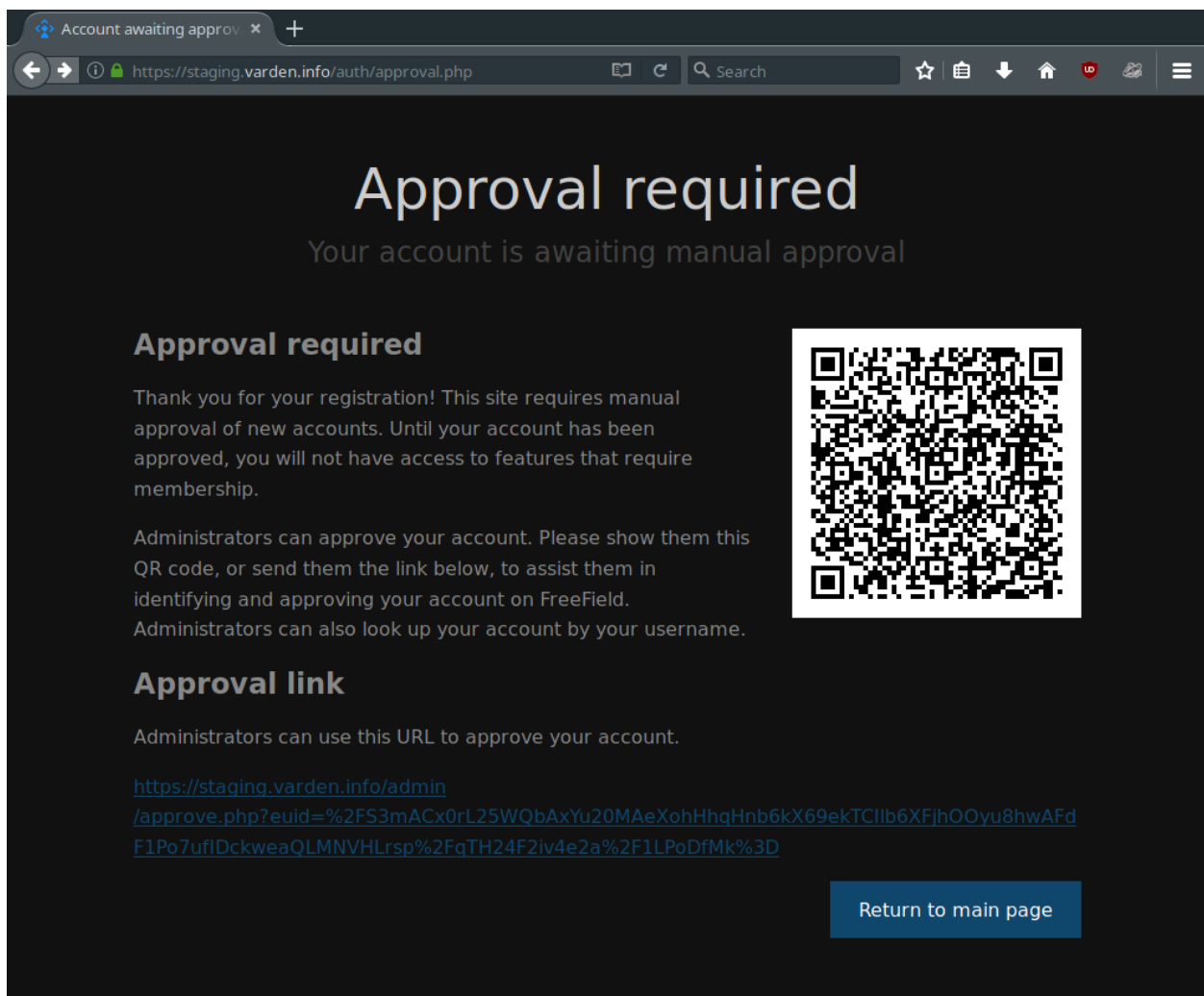
Important: A user whose account has been deleted will still be able to sign up again later if they attempt to log in. This will also re-link any references to the user in other places, such as in the Pokéstops database.

Log out everywhere If you believe that a user has had their accounts compromised, or for some other reason want to force a user to sign out, then you can use this action to reset their access token in the users database. This will immediately and permanently invalidate all of the user's active sessions, forcing them to log back in the next time they visit FreeField.

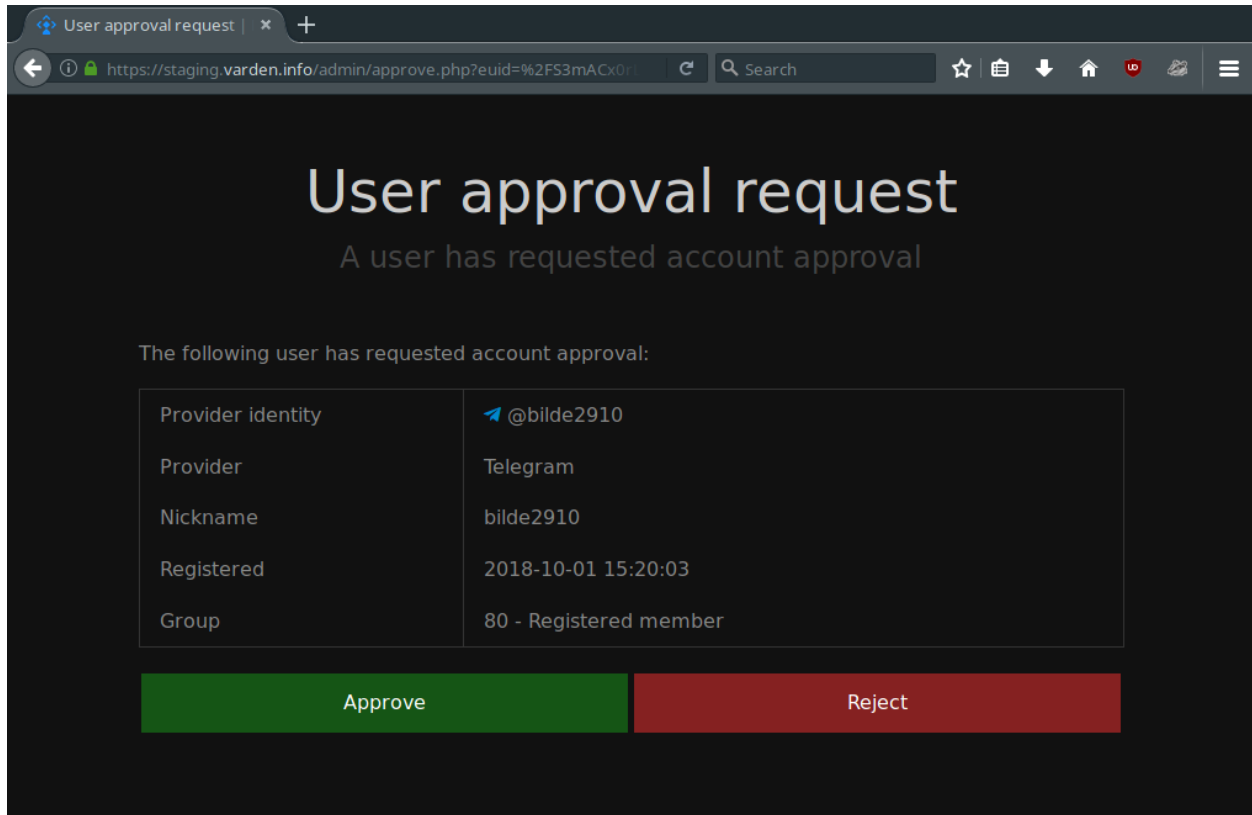
3.2 Manual user approval

FreeField has a manual approval setting that allows administrators to require that all newly registered users are manually approved by an administrator before they can use their accounts. Until a user's account has been approved, the user will be given the same permissions as users who are browsing anonymously, i.e. haven't signed in. To enable manual approval, check the "Require manual approval" box under "User creation" on the "Security" section of the administration pages.

When manual user approval is enabled, newly registered users will see a screen after signing up that looks something like this:



The page will contain a URL that the newly registered user can forward to a site administrator. When an administrator clicks on the link, they will see a screen similar to the following:



This way, administrators can easily approve users by link rather than having to look up the user manually in the user list on the administration pages.

3.2.1 Approval by QR code

If you have installed and enabled the `gd` extension in `php.ini`, FreeField will by default also generate a QR code that can be scanned by an administrator using a mobile device. This makes it easy to approve users in scenarios where the administrators wish to meet people in person to approve their accounts. The QR code will point to the same approval screen as the link, meaning that if administrators meet the user in person, there is no need for the user to exchange their link with an administrator on a third party messaging service.

If you wish to enable or disable this functionality, you can check or uncheck the “Present approval QR codes” box underneath the “Require manual approval” checkbox.

Permissions and groups

FreeField uses a permissions tier system to assign users various permissions. Each permission tier is called a group, and each group is assigned a permission level, which is an integer between 0 and 250 describing the access level of users within that role to functionality in FreeField.

Tip: Each group can additionally be assigned a color code that is used to distinguish users of a particular group in the user list. This is described in greater detail in the [Group settings](#) section of this page.

4.1 Default groups

By default, FreeField comes with seven groups:

Group name	Permission level	Color
Site host	250	Green
Administrator	200	Red
Moderator	160	Purple
Pokéstop submitter	120	Turquoise
Registered member	80	< none >
Read-only member	40	< none >
Anonymous visitor	0	< none >

Members of each group are automatically granted the permissions of all groups below their permission level in addition to the permissions specific to their own group. The default groups are set up in such a way that they should be safe for production use if set up properly according to the intentions below.

Caution: If a permission that is reserved to a high-level group is desired for a lower level group, then that group should be granted that specific permission rather than be merged with a higher permission level group. Granting

users access to permissions they do not need can be dangerous, and the security and availability of your site may be put at risk if you do so.

4.1.1 Site host

The site host is the group with highest permissions in FreeField. As the name implies, this group should be reserved only for the person who is hosting the FreeField site. Site hosts have access to settings that can be very dangerous to change, or can break FreeField if changed, such as database connection settings, authentication provider setup and site updates.

Danger: If your community has several administrators, it is strongly recommended that only the administrator responsible for hosting FreeField has this role, and that the others are assigned to the Administrator group instead. Failure to do so presents a risk that your entire FreeField installation can be hijacked by a rogue administrator, in the worst case warranting a complete reinstall of FreeField. This is explained in greater detail under *The “Manage own group” permission* at the end of this document.

4.1.2 Administrator

Administrators have the second highest level of access to FreeField, and have access to most functionality needed to run and manage FreeField at a co-administrative level. Administrators have access to functionality such as site appearance settings, permissions management, security settings, *Geofencing* and webhooks. Only trusted community leaders should be assigned to this group, as malicious users with this level of access will be able to do significant damage to FreeField.

4.1.3 Moderator

Moderators have the third highest level of access to FreeField. The purpose of this group is to allow privileged members of a community to manage other members’ access - i.e. they can approve and reject users awaiting approval (see *Manual user approval*), delete the accounts of registered users, and modify and delete any Pokéstops.

They can also change the group memberships of other users, but only for users who are below their own level of access.

4.1.4 Pokéstop submitter

Pokéstop submitters have the same privileges as regular members, with the notable exception that they are able to submit new Pokéstops to the map. This group should be assigned to members who wish to contribute to the completeness of the map.

4.1.5 Registered member

This is the default group for new members. The only privilege these members have over non-registered users is that they are able to submit field research tasks. They can also overwrite field research tasks submitted by others, but this permission can be revoked if deemed necessary by administrators.

4.1.6 Read-only member

Users can be assigned to this group if they have caused trouble or for some other reason have to be denied the permission to submit field research tasks. By default, this group is functionally the same as the Anonymous group in terms of permissions and access, but it has been included in case administrators would want to restrict viewing the map to members with accounts only. This way, administrators would be able to grant access to the map for a user, without also having to grant them access to submitting field research, if the map is set up to generally require an account to view the map in the first place.

4.1.7 Anonymous visitor

This group is the default group that all visitors, no matter where they are, are automatically assigned to. Users who have not registered for FreeField are treated as if they were members of this group. If you set any permissions to this level of access, anyone who visits your FreeField instance can perform the functions associated with that permission without having to sign in. The permission level of this group is always 0, and cannot be changed.

The only permission which is set to this group by default is “View map,” to allow anyone to view the map and any submitted research without having to authenticate.

4.2 Group settings

You can add, remove and manage groups as you wish on the “Groups” section of the administration interface. The following options are available to change for each group:

4.2.1 Group name

Each group has a name that is used to refer to that group elsewhere in FreeField. This name can be just a plain string, such as “Moderator,” or it can be an internationalization token which is automatically translated into the language of the user who is browsing FreeField.

All of the default groups use internationalization tokens to ensure that the names are readable in all supported languages, and not limited to one single language for everyone. An internationalization token takes the form `{i18n:token_id}` where the token ID is a string representing the key for a particular localizable string in the localization files. The latest localization files can be [found on GitHub](#). The token IDs used by groups in FreeField all start with `group.level..`

Note: If you want to use a custom name for a group, you should replace the entire internationalization token with the string that you wish to use. You should not add the string to your local copy of the localization files, as these are overwritten every time FreeField is updated - make the required changes on the administration pages instead.

4.2.2 Permission level

Each group is assigned a permission level that dictates which permissions the group has. Each group is granted all permissions at and below their permission level automatically.

Two groups cannot share the same permission level.

Caution: It is strongly recommended that you do not change the permission levels assigned to the default groups. This is because updates to FreeField that add new permissions will use the default permission levels as a reference when they are populated with defaults on your FreeField installation. E.g. if a new permission is added that is only meant to be accessible to administrators by default, the permission will be set at level 200 regardless of what value you may have chosen for the local Administrators group.

4.2.3 Color

Each group can also be assigned a color. This color is displayed in other places on the administration pages, as well as in the users list, to more easily distinguish those groups from others. A group can also be assigned the default color.

To assign a color to a group, select a color from the color input box in the row that corresponds to your group. If you wish to use the default color, uncheck the checkbox next to the color box. The default color is #888888 (r=136, g=136, b=136) when using the dark color theme, and #777777 (r=119, g=119, b=119) when using the light theme.

4.2.4 Actions

The “Groups” section on the administration pages allows administrators to perform actions on groups. Actions can be performed on several groups at once through selecting an action for several groups in the list, which will then be applied all at once when clicking on *Save settings*. The available actions for groups are as follows:

Delete group This action will, if selected, delete the group from the groups database. There are several considerations you should consider when deleting groups. See *Adding and removing groups* for more information.

4.3 Adding and removing groups

In addition to the default groups that are pre-installed on FreeField, it is possible to add additional groups for more granular control over individual permissions. When adding a new group, you have to enter a name for the group, a permission level, and an optional color to represent it.

The permission level should be chosen so that it falls between two other groups in FreeField. For example, if you wish to add a new group between the “Registered member” group (level 80) and “Pokéstop submitter” group (level 120), you could assign the new group permission level 100. Note that it is not possible for two groups to share the same permission level.

You can also delete groups by selecting the “Delete group” action for the group in the groups list. There are several considerations you should consider when deleting a group:

- Users who are in the group when it is deleted will automatically be reassigned to an “Unknown” group with a permission level corresponding to the level of the deleted group.
- Permissions which are set to the group that is being deleted will automatically change to be granted to the aforementioned “Unknown” group. This ensures that the permissions of any members in the group remain unchanged.
- Users can be moved from the “Unknown” group to any other group, but cannot be moved from another group to the “Unknown” group.
- Similarly, permissions which are set to the permission level of the “Unknown” group can be changed to another permission level, but cannot be changed back again.
- If a new group is created with the same permission level as a previously deleted group, then all members who are currently in the “Unknown” group corresponding to the permission level of that group are automatically moved to the new group.

- This also applies to permissions - any permissions which are explicitly granted to any “Unknown” group that corresponds to the level that the new group is added at, are reconfigured to be granted to the newly added group instead.
- If you change the permission level of, or delete, a default group, then any future updates to FreeField that add additional permissions to that default group will result in those new permissions automatically being assigned to an “Unknown” group that corresponds to the default level of that group. You may want to change the group assignment of those permissions after such an update has completed.

4.4 Default group for new members

The default group for new members is “Registered member.” This can be changed on the “Permissions” section of the administration pages.

Hint: If you wish to manually approve new members before granting them access to FreeField, then this is not the setting you should change. Instead, look into [Manual user approval](#).

4.5 Managing permissions

You can find a list of all configurable permissions in FreeField on the “Permissions” section of the administration pages. If you set a permission to a particular level, then all users who are assigned to a group with a permission level at or above the level of the selected group are granted the permission in question.

Users who have access to change permissions (i.e. users who have been granted “Manage permissions”) are only able to change permissions whose currently assigned group has a permission level lower than the one they themselves are a member of. This means that Administrators, for example, cannot change permissions which are currently granted to Administrators or the Site host. Neither can they restrict a permission that they *can* change to a group with a permission level that is the same as or higher than that of their own group. This means that Administrator users cannot change the assigned group of a permission that is currently granted to Pokéstop submitters, to Administrators or the Site host. They can, however, change the permission to any group ranging from Anonymous visitor through Moderator, as these are all below the permission level of the Administrator user who is making those changes.

4.5.1 The “Manage own group” permission

There is one permission in FreeField that warrants extra attention in the documentation - the “Manage own group” permission, which by default is only granted to the Site host.

The default behavior of FreeField when it comes to users sharing a group, is that users can only make changes to other users, groups and permissions that are *below* the current level of their own group. This means that members within a group cannot change each others’ details, they cannot restrict access to a permission to their own group, and they cannot assign or revoke access for members to their own group. In practice, this means that moderators cannot appoint other moderators, and administrators cannot appoint other administrators - they would have to consult with a user of a higher level group to make those changes on their own behalf.

This is a security measure. If e.g. administrators were able to manage their own group, then nothing would stop one administrator from demoting all other administrators to a lower rank, taking practically full control over FreeField and leaving the Site host to clean up the mess. Furthermore, restricting access for users to manage their own group and their group’s members reduces the attack surface for malicious users who try to seize control of an administrator account for e.g. escalating their own account to administrator level to only one account (the Site host) rather than the entire administration team.

This unfortunately has a significant practical implication - several settings in FreeField are restricted to being changeable by the Site host only by default, meaning that if the Site host could not change settings at their own level, they would not be able to change the settings despite being super-administrators on the site, a permission level whose intention is to be able to manage literally every setting in FreeField.

To remedy this, the “Manage own group” setting exists. Groups who have this permission will bypass the group self-management restrictions, so that they *can* make changes at their own permission level. This setting essentially raises the permission level of the groups who have the permission granted by one. This is also why the “Manage own group” setting should always remain at the Site host level and should never be granted to other users.

Members of groups with this permission granted will still not be able to change permissions or group/membership settings for any groups *above* their current permission level, even though they can make changes *at or below* their own level.

This permission is also the reason that there should only be one Site host. If you as the Site host assign another user to the Site host group, that user would have full rights to revoke your own Site host group membership, seizing full and unrestricted access to the entire FreeField installation, and eliminating your own ability to take back control. The only way to recover from such a breach would be to access the users table in the database and change the malicious user’s permission level directly. If the user manages to switch the database connection settings to another database provider first, then recovering would be even harder, likely warranting directly modifying the FreeField config.json file or even completely reinstalling FreeField.

5.1 Map providers

FreeField uses basemap tiles from third party providers. Currently, Mapbox and Thunderforest is supported.

You are required to set up one map provider in FreeField, otherwise the map will not load. Instructions for setting up each supported map provider is provided on the pages below.

5.1.1 Mapbox

Mapbox is a commercial basemap provider that provides map tiles rendered from OpenStreetMap data. Mapbox operates with several pricing tiers and an additional monthly fee for commercial use. Please see Mapbox' [pricing page](#) for more information.

In order to set up Mapbox as the basemap provider for FreeField, you have to register an account on Mapbox and obtain an access token.

Attention: You are fully responsible for ensuring you comply with the Mapbox terms of service if you use Mapbox as your map provider. You are also fully responsible for ensuring that you stay within your usage quota, and for covering any costs incurred by your usage of Mapbox' services through FreeField. You are also fully responsible for determining whether or not your usage of Mapbox' services qualify as commercial use (for example, whether or not you are restricting access to your application), and for covering the cost of a commercial use license if required by Mapbox.

Obtaining an access token

To obtain an access token, go to [this page](#). If you do not have a Mapbox account, create one and follow the steps on the Mapbox website to get an access token.

Using Mapbox with FreeField

1. In the FreeField administration pages, navigate to the “Map settings” menu. Look for the “Map provider” section on this page.
2. In the “Map source” menu, select “Mapbox (mapbox-gl.js)”.
3. Enter your access token in the “Mapbox access token” field.
4. Save the settings using *Save settings* at the bottom of the page.

5.1.2 Thunderforest

Thunderforest is a basemap provider that provides map tiles rendered from OpenStreetMap data. The Thunderforest Platform is the set of systems that power OpenCycleMap, and is developed by Andy Allan. Thunderforest operates with several pricing tiers and an additional monthly fee for commercial use. Please see Thunderforest’s [pricing page](#) for more information.

In order to set up Thunderforest as the basemap provider for FreeField, you have to register an account on Thunderforest and obtain an API key.

Attention: You are fully responsible for ensuring you comply with the Thunderforest Terms and Conditions if you use Thunderforest as your map provider. You are also fully responsible for ensuring that you stay within your usage quota, and for covering any costs incurred by your usage of Thunderforest’s services through FreeField. You are also fully responsible for determining whether or not your usage of Thunderforest’s services is permitted within the service plan you have chosen for your account with Thunderforest, and for covering the cost of higher service plans if required by Thunderforest.

Obtaining an API key

To obtain an API key, go to [this page](#) and sign in. If you do not have a Thunderforest account, choose a service plan create an account [here](#), verify your account, then go to the dashboard to find your API key.

Using Thunderforest with FreeField

1. In the FreeField administration pages, navigate to the “Map settings” menu. Look for the “Map provider” section on this page.
2. In the “Map source” menu, select “Thunderforest (leaflet.js)”.
3. Enter your API key in the “Thunderforest API key” field.
4. Save the settings using *Save settings* at the bottom of the page.

CHAPTER 6

Pokéstops

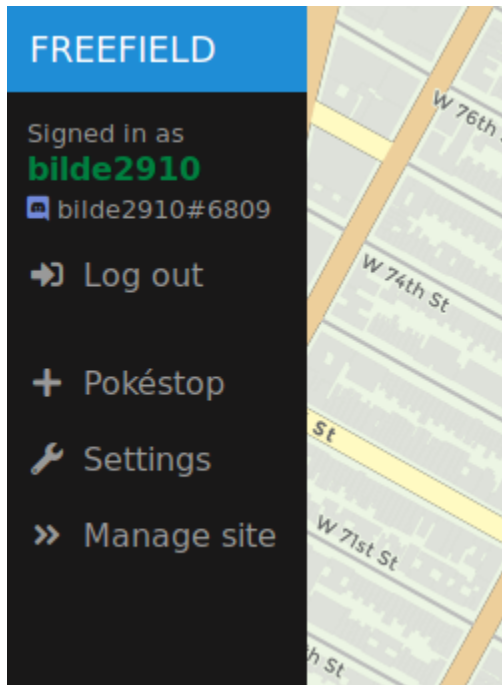
For FreeField to function according to its intended purpose, a list of Pokéstop locations need to be added to the map. This page will describe the processes of adding and managing Pokéstops.

Note: This page only covers administrative management of Pokéstops. For information on submitting research, please see [Reporting field research](#).

6.1 Adding a new Pokéstop

Any user who has the “Submit Pokéstops” permission (by default, the Pokéstop submitter group and above) can submit Pokéstops directly from the map view.

1. On the main page of FreeField, click on + *Pokéstop* in the sidebar menu.

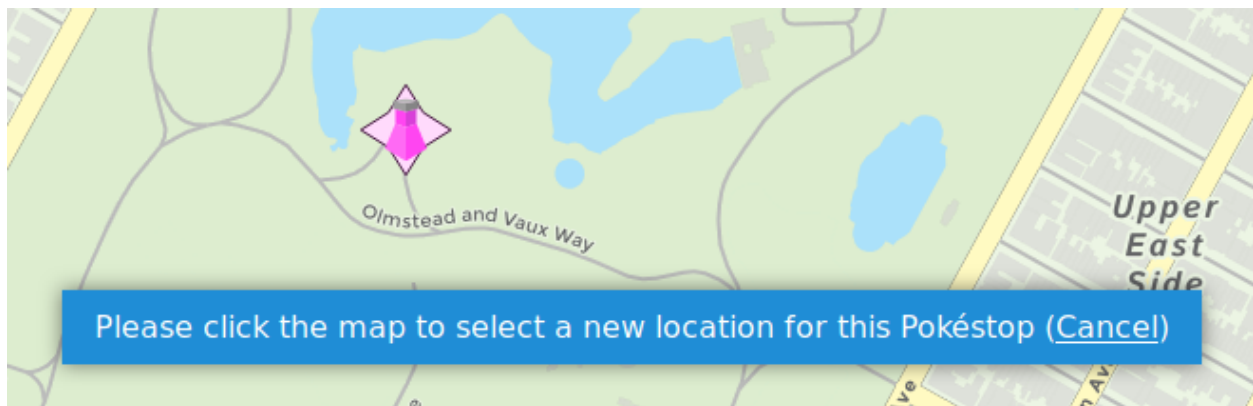


2. Click on the map at the position that the Pokéstop should be located.
3. A popup will appear, prompting you for the name of the Pokéstop. Write a name, then click on *Add Pokéstop*.

6.2 Moving a Pokéstop

If a Pokéstop is wrongly located, you can move it to another location directly from the map view. This requires the “Manage Pokéstops” permission.

1. Click on the Pokéstop on the map that is wrongly located.
2. Click the “Move” button (indicated by a four arrows icon).
3. Click on the map at the location that the Pokéstop should be moved to.

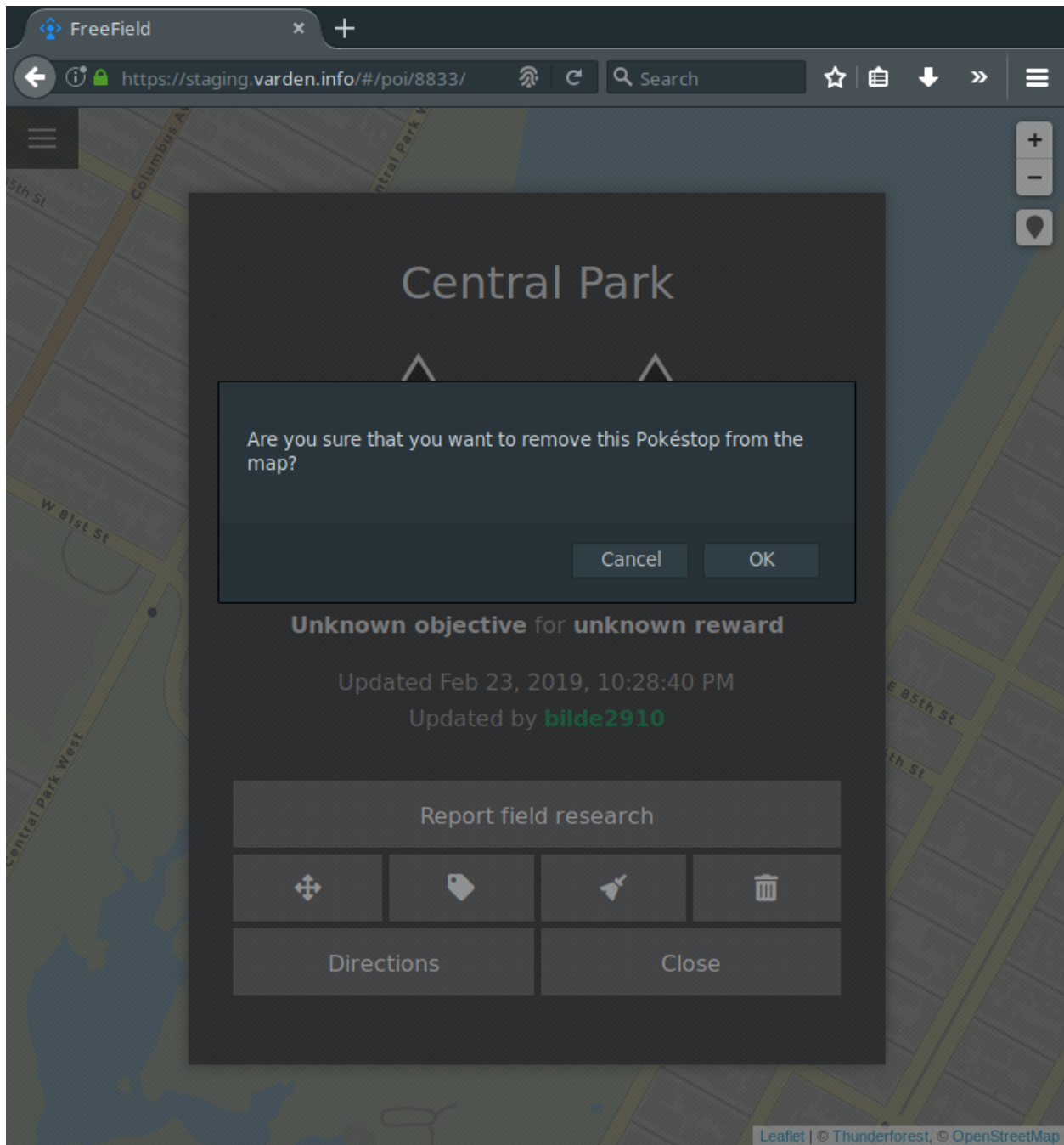


6.3 Removing Pokéstops

Pokéstops can be removed in two ways - through the administration pages, or directly from the map view.

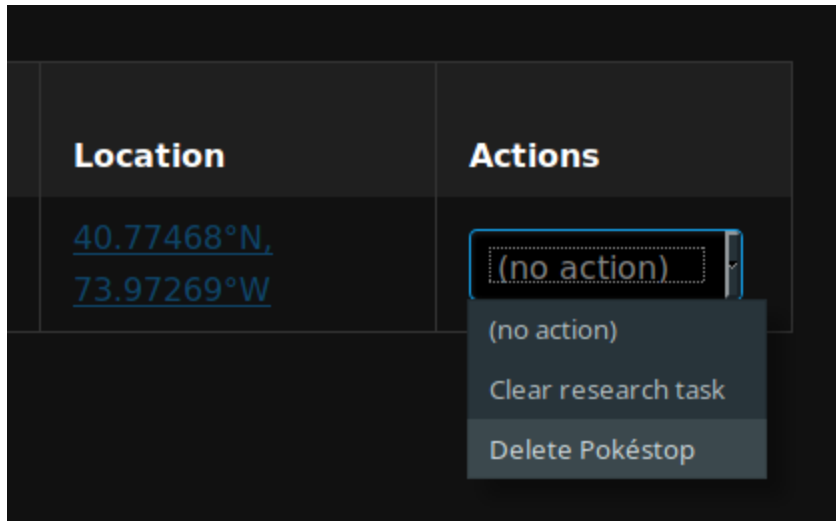
6.3.1 Removing from the map view

1. Click on the Pokéstop on the map that should be removed.
2. Click the “Delete” button (indicated by a trash bin icon).
3. In the popup that appears, click *OK* to confirm removal of the Pokéstop.



6.3.2 Removing from the administration pages

1. Navigate to the “Pokéstops” section on the administration pages.
2. In the list of Pokéstops, look for the Pokéstop(s) that you want to remove.
3. In the “Actions” column for the Pokéstop(s), select “Delete Pokéstop.”
4. Click on *Save settings* at the bottom of the page.



6.4 Clearing field research

To clear the currently active field research on a Pokéstop and reset it to the “unknown” research task, do either of the following:

6.4.1 Clearing from the map view

1. Click on the Pokéstop on the map that you want to clear the research task for.
2. Click the “Clear research” button (indicated by a brush icon).
3. In the popup that appears, click *OK* to confirm that you wish to clear the currently active research from the Pokéstop.

6.4.2 Clearing from the administration pages

1. Navigate to the “Pokéstops” section on the administration pages.
2. In the list of Pokéstops, look for the Pokéstop(s) whose active research task you want to clear.
3. In the “Actions” column for the Pokéstop(s), select “Clear research task.”
4. Click on *Save settings* at the bottom of the page.

6.5 Renaming Pokéstops

You can rename a Pokéstop either from the map or from the administration pages:

6.5.1 Renaming from the map view

1. Click on the Pokéstop on the map that you want to rename.
2. Click the “Rename” button (indicated by a tag icon).

3. In the popup that appears, enter a new name for the Pokéstop and click *OK* to submit the name change.

6.5.2 Renaming from the administration pages

1. Navigate to the “Pokéstops” section on the administration pages.
2. In the list of Pokéstops, look for the Pokéstop(s) you want to rename.
3. In the “Name” column for the Pokéstop(s), enter the new name for the Pokéstop(s).
4. Click on *Save settings* at the bottom of the page.

6.6 Managing Pokéstops

FreeField has several functions for managing submitted Pokéstops. You can, for example, change the name of Pokéstops, see who submitted each Pokéstop, and perform actions on several Pokéstops at a time. You can find the Pokéstop management interface in the “Pokéstops” section of the administration, where every Pokéstop in the FreeField database will be listed. The table contains the following fields:

Name The name of the Pokéstop. You can change the name to something else here. Names are not unique; several Pokéstops can have the same names without causing conflicts with each other.

Created Shows the date and time at which the Pokéstop was added to the Pokéstops database in FreeField.

Created by Shows the nickname and identity of the user who submitted the Pokéstop to FreeField.

Current research Shows a brief summary of the currently active research task on this Pokéstop. The icons in this field can be clicked to open the details dialog popup for the Pokéstop on the FreeField map, where new research can be reported.

Last updated Shows the last time an update was made to a Pokéstop. Updates include research submission, clearing its currently active research, as well as moving the Pokéstop to another location.

Updated by Shows the nickname and identity of the user who made the aforementioned update to the Pokéstop.

Location Shows the coordinates of each Pokéstop. This field is a link that can be clicked to pan the FreeField map to the location of the Pokéstop.

Actions Allows performing actions on the given Pokéstop, such as deleting it, or clearing its field research.

6.6.1 Batch processing

It is possible to clear research from, and delete, Pokéstops in batches. This is done under the “Batch processing” heading of the “Pokéstops” section of the administration pages.

Batch processing is performed using geofences. When you perform an action on a geofence, the action affects all Pokéstops within that area. Please see [Geofencing](#) for information on setting up geofences.

Caution: Geofences do not function as masks. If you have one geofence within another, and you perform an action on the larger geofence while keeping the smaller one at “no action,” the action will still affect Pokéstops in the smaller area, given that those Pokéstops are also within the bounds of the larger geofence.

6.7 Exporting and importing Pokéstops

You may export the list of Pokéstops stored in FreeField. The exported file can be used to re-import Pokéstops later in another FreeField instance. It serves as an easy way to take a backup of the database.

6.7.1 Exporting Pokéstops

To export all of your Pokéstops, go to the “Pokéstops” section of the administration pages, scroll to the bottom of the page, and look for the “Export Pokéstops” header. Click on the link labeled “Click here to export the Pokéstop database.” All of the Pokéstops in the database will be downloaded to your computer in CSV format.

6.7.2 Importing Pokéstops

You can import a previously exported list of Pokéstops back into FreeField.

1. Navigate to the “Pokéstops” section of the administration pages and scroll down to the “Import Pokéstops” section.
2. Next to “Choose file,” browse for the file that contains the Pokéstop database export CSV file.
3. In the “Name,” “Latitude” and “Longitude” fields, select the column of the CSV file that contains the names, latitudes and longitudes of the Pokéstops, respectively.
4. The table below the above mentioned fields will be populated with the Pokéstops from the CSV export you selected. Make sure to check that all fields in the table are correct, and to fill in missing entries (if any). Missing fields are highlighted in red. If you do not wish to import a specific Pokéstop in the table, you can select “Don’t import” in the “Import?” column of the table for the Pokéstops you wish to ignore.
5. Click on *Save settings* at the bottom of the page to import all listed Pokéstops that are marked for import.

Import Pokéstops

Choose file: ?

Browse...

FFExportPoi_FreeField_2018-10-16_09-28-27.csv

Name field: ?

Name

Latitude field: ?




Latitude

Longitude field: ?

Longitude

Preview imported Pokéstops

The selected input file contains 1 Pokéstops.

Name	Latitude	Longitude	Import?
Cherry Hill	40.774675604275 	-73.972685337067 	Import 

FreeField supports restricting the geographical scope and coverage of various functionality through geofencing.

Note: If you have just set up FreeField, it is highly recommended that you set up a geofence that defines the boundary of the region you are covering with FreeField, to prevent users from submitting Pokéstops to the map in areas that are not supposed to be covered by your map. To do this, please follow the steps in *Defining a boundary* below to create a geofence that covers your entire community area, then follow the instructions in *Limiting Pokéstop submission* to apply the fence to FreeField and restrict usage of FreeField to the area within the defined geofence.

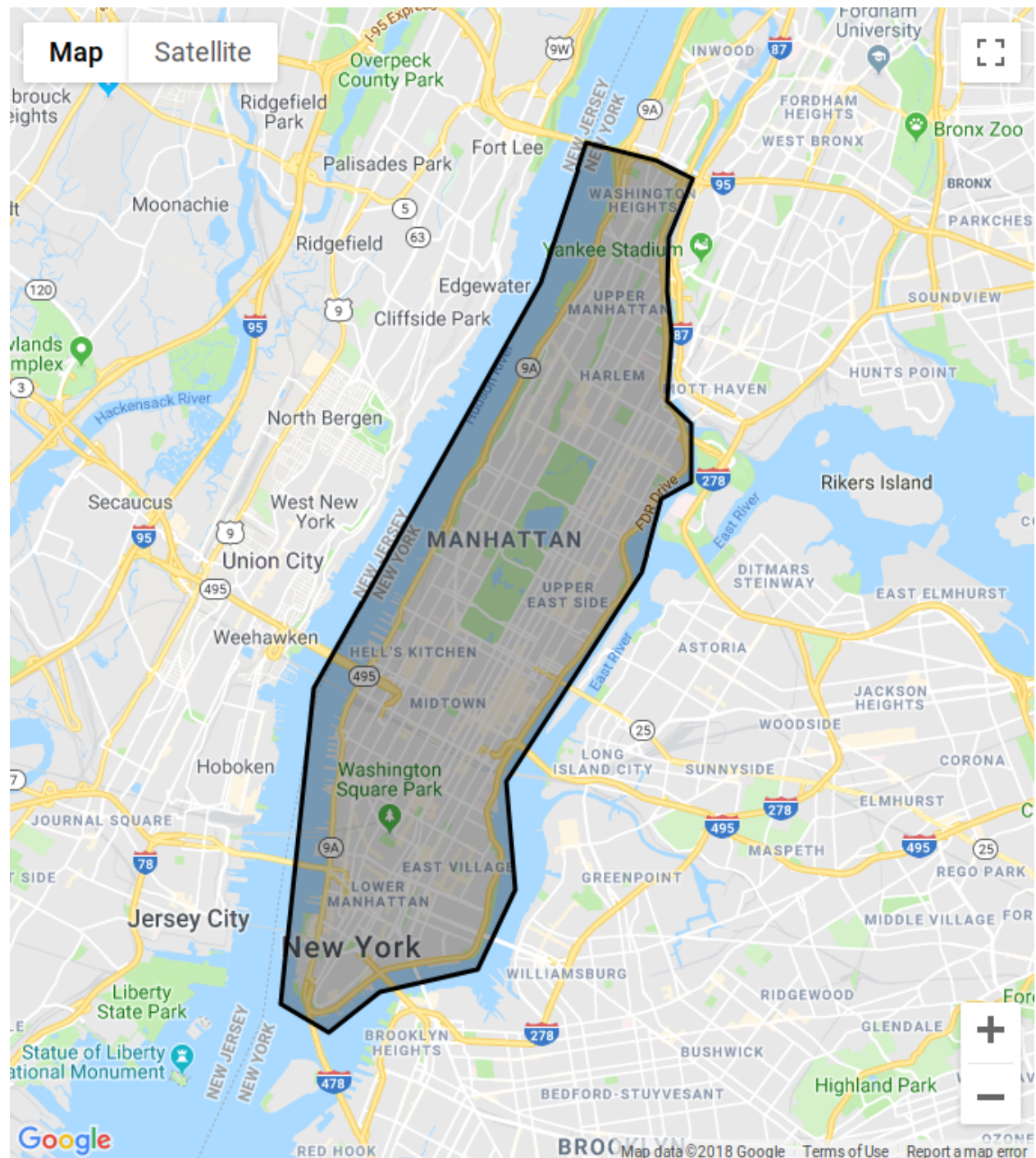
A geofence is a list of three or more coordinate pairs that together make up a polygon covering a certain geographical area. Geofences can be set up on the “Geofences” section on the administration pages. This documentation entry will guide you through creating and installing a geofence in FreeField, and then using it to restrict the map to only being used in certain regions.

7.1 Defining a boundary

The easiest way to define a boundary for a geofence is to use an online geofence editor. Here are links to a couple:

- <http://geo.jasparke.net/>
- <https://codepen.io/bgus/full/dXxLjp>

Draw an area on the geofence editor that corresponds to the area that you wish to cover with FreeField:



Once you have drawn the geofence, export the geofence to a list of coordinate pairs. In FreeField, navigate to the “Geofences” section of the administration pages. Click on *Add new geofence*, enter a name for your geofence to easily identify the area it covers, and paste the coordinate pair list in the “Vertices” column.

Label	Vertices	Actions
Manhattan	40.8516364639817,-73.9523844410642 40.848520094547,-73.9362482716306 ...	(no action)

Add new geofence
 Save settings

Finally, click *Save settings*. Your geofence is now saved and ready to be used elsewhere.

7.2 Limiting Pokéstop submission

A crucial step in securing your FreeField installation is restricting it to only be usable within the bounding area that your community covers. To do so, first define and install a geofence as defined in [Defining a boundary](#) that covers the entire area you wish to support FreeField for.

After you have defined a geofence, navigate to the “Map settings” section of the administration pages. Scroll down until you find the “Geofencing” header. In the “Pokéstop geofence” box, select the geofence you just defined, and click on *Save settings*.

Geofencing

For information on how to set up geofence restrictions, please [see the documentation](#).

Pokéstop geofence: Manhattan

Out-of-bounds Pokéstops: ☐ Hide all Pokéstops outside of the selected geofence

Tip: There is a setting that allows you to hide all Pokéstops outside of the selected geofence boundary. This is useful if you previously allowed Pokéstop submission outside the current boundary - all of those Pokéstops will be hidden, and can research can no longer be submitted on them while the geofence is in effect and while this setting is active.

Warning: If you delete the geofence that is used to limit Pokéstop submission, the setting will revert to the default setting (no geofences applied), again allowing Pokéstop submissions globally. Make sure to define a new geofence and to update this setting again if you ever delete this geofence.

7.3 Applying geofences to webhooks

Geofences can easily be applied to webhooks. Each webhook has an option labeled “Geofence” where you can select any geofence defined as described on this page. When you apply a geofence to a webhook, the webhook will only be triggered if the Pokéstop that the field research is reported on is within the bounds of the selected geofence.

Webhooks

A core component of research reporting is that community members are alerted to research tasks of interest. This can be accomplished with webhooks. A webhook is a handler for web requests that triggers some kind of action when called - e.g. a message can be sent in a channel on Discord, or an alert triggered in a Telegram group.

Webhooks can be set up from the “Webhooks” section on the administration pages. There are built-in presets for some types of webhooks, but in principle, any service can be the target of a webhook if the service supports setting up webhook endpoints for inbound requests.

8.1 Webhook basics

All webhooks share some common principles - when field research is reported on a Pokéstop, FreeField will evaluate the trigger conditions of all registered webhooks, and if the reported research matches all the defined conditions, the webhook is triggered. Every webhook has a target and a payload, and if the webhook is triggered, FreeField connects to the target and sends the payload.

Webhooks can be added from the “Webhooks” section on the administration pages by clicking on *Add new webhook*. A dialog box will open, prompting you for the type of webhook to add, and if you wish to use a payload preset.

Tip: Payload presets are pre-configured payloads that, if selected, will automatically fill in the payload for your webhook. This can save you from having to configure your webhook’s payload from scratch according to the documentation for the service that you are setting up a webhook for. Payload presets are described in greater detail under *Presets*.

Hint: You can refer to the documentation for *Common webhook targets* below to see what webhook types and payload presets you should choose for the most common webhook target services. You should refer to this documentation before you create your webhook.

Here is a general overview of each webhook type:

8.1.1 Post JSON

This type of webhook will make an HTTP POST request to the given target URL with the specified payload. The payload is encoded as JSON, with enforced syntax validation. If you enter invalid JSON in the payload field of the webhook, you will not be able to save the webhook settings.

Hint: This is the most common type of webhook, and most webhook endpoints support this type of request.

Tip: In addition to FreeField’s built-in JSON validation, you can use a third party service such as [JSONLint](#) to validate the JSON payload.

Requests sent with this webhook type will have the following HTTP headers:

```
Content-Type: application/json
User-Agent: FreeField/<FreeField_Version> PHP/<PHP_Version>
```

8.1.2 Send Telegram message

This type of webhook is specific to sending messages to groups in Telegram. It is a wrapper around the *Post JSON* webhook request type and functions more or less the same as Post JSON internally, but exists as a convenience for making Telegram webhooks easier to manage. Please refer to *Telegram webhooks* for more information about how to set up this webhook type.

8.2 Common webhook targets

This documentation offers examples for several common webhook targets, such as Discord. You are recommended to refer to the pages below if you are setting up webhooks for any of these services, as they explain webhook properties specific to these services, to help you get the most out of your webhooks.

8.2.1 Discord webhooks

This guide will help you set up Discord webhooks in FreeField.

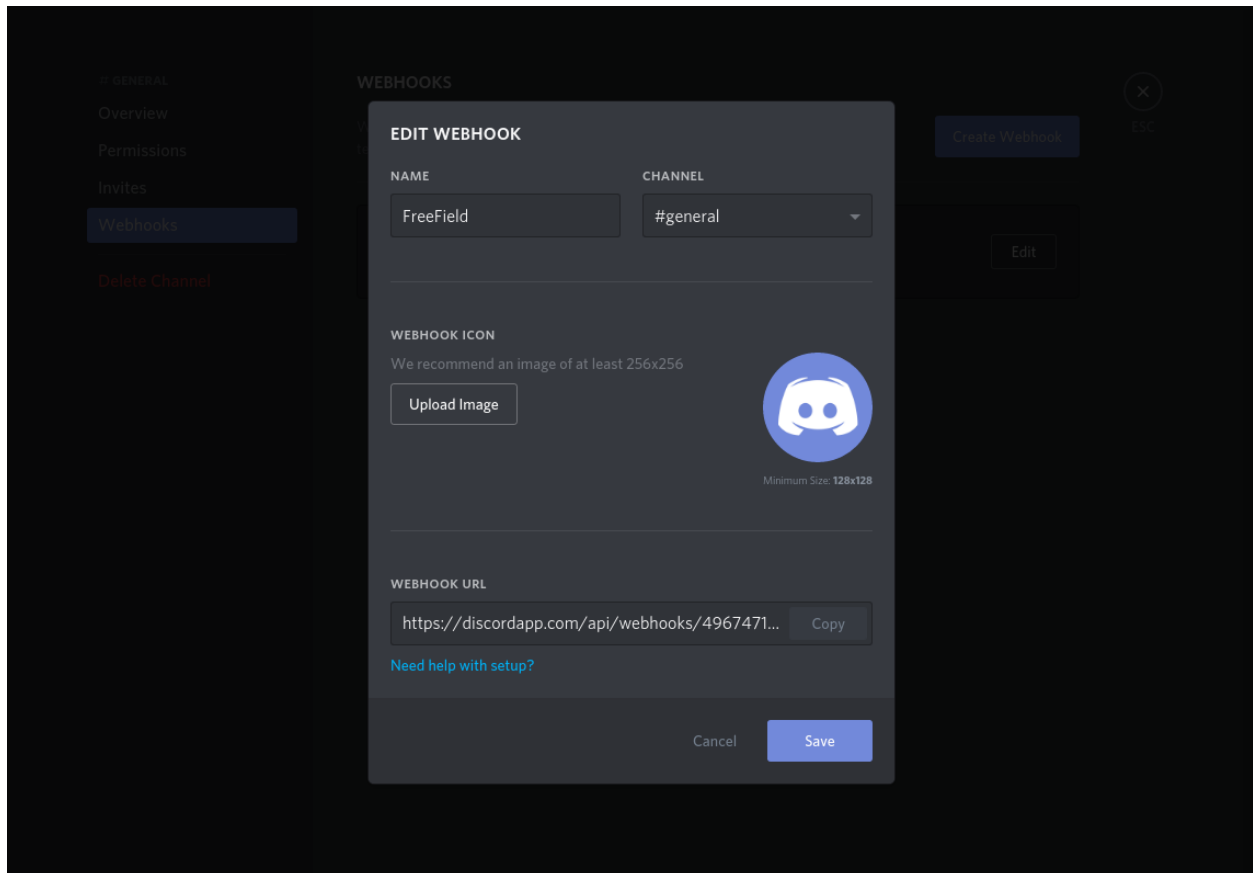
Important: You need the “Manage Webhooks” permission within the target Discord server in order to create webhooks on Discord.

Note: This page assumes you have read the general documentation on setting up webhooks, located on the *Webhooks* page. Discord also has specific documentation on webhook setup for end users in their [help center](#). The FreeField documentation on Discord webhooks builds upon this article.

Creating a webhook in Discord

1. Open Discord and find the channel that you wish for FreeField alerts to be posted in.
2. Open channel settings for the channel by clicking on the cog next to the channel name in the channels list.

3. In the “Webhooks” section in the dialog that appears, click on *Create Webhook*.
4. Give your webhook a name and optionally an avatar. The name of your webhook is what would normally appear as the username when a person talks in your Discord server.



5. Copy the webhook URL at the bottom of the webhook dialog, then click *Save*.

Configuring the webhook on FreeField

1. In the “Webhooks” section on the FreeField administration pages, click on *Add new webhook*.
2. Select “Post JSON” as the webhook type, and select an appropriate preset from the list of presets. Make sure you select a preset listed under *Supported presets*. Click *Done* to create your webhook trigger.
3. In the “Webhook URL” field, paste the webhook URL you copied from Discord above.
4. Configure the webhook’s settings and add filtering according to your own preferences.
5. Click on *Save settings* to save and activate your newly created webhook.

Payload syntax

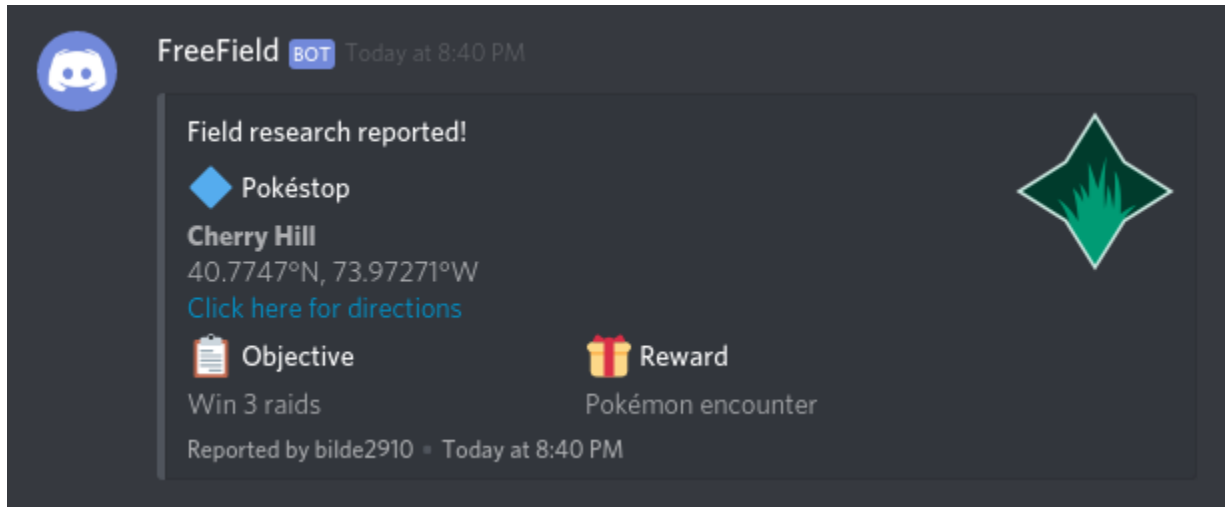
In order for Discord webhooks to function, the syntax of the payload has to be valid. Please refer to Discord’s [developer documentation on webhooks](#) for information on constructing a valid payload.

Warning: If your payload does not have a valid structure, it will silently fail.

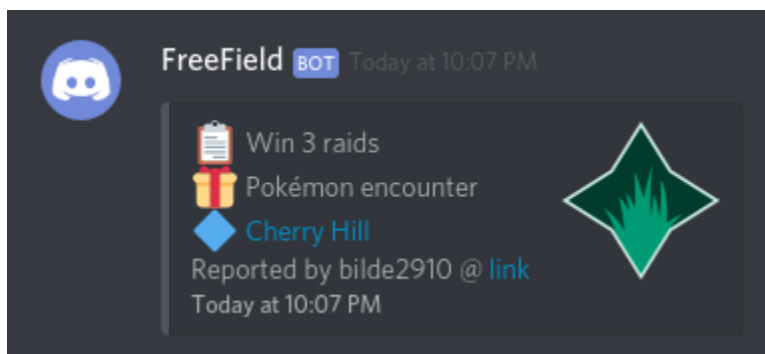
Supported presets

The following presets are supported by FreeField's Discord webhooks:

discord-embed-full.json



discord-embed-compact.json



8.2.2 Telegram webhooks

This guide will help you set up Telegram webhooks in FreeField. Creation of a Telegram bot is required to set up message broadcasting in Telegram groups from FreeField.

Note: This page assumes you have read the general documentation on setting up webhooks, located on the [Webhooks](#) page.

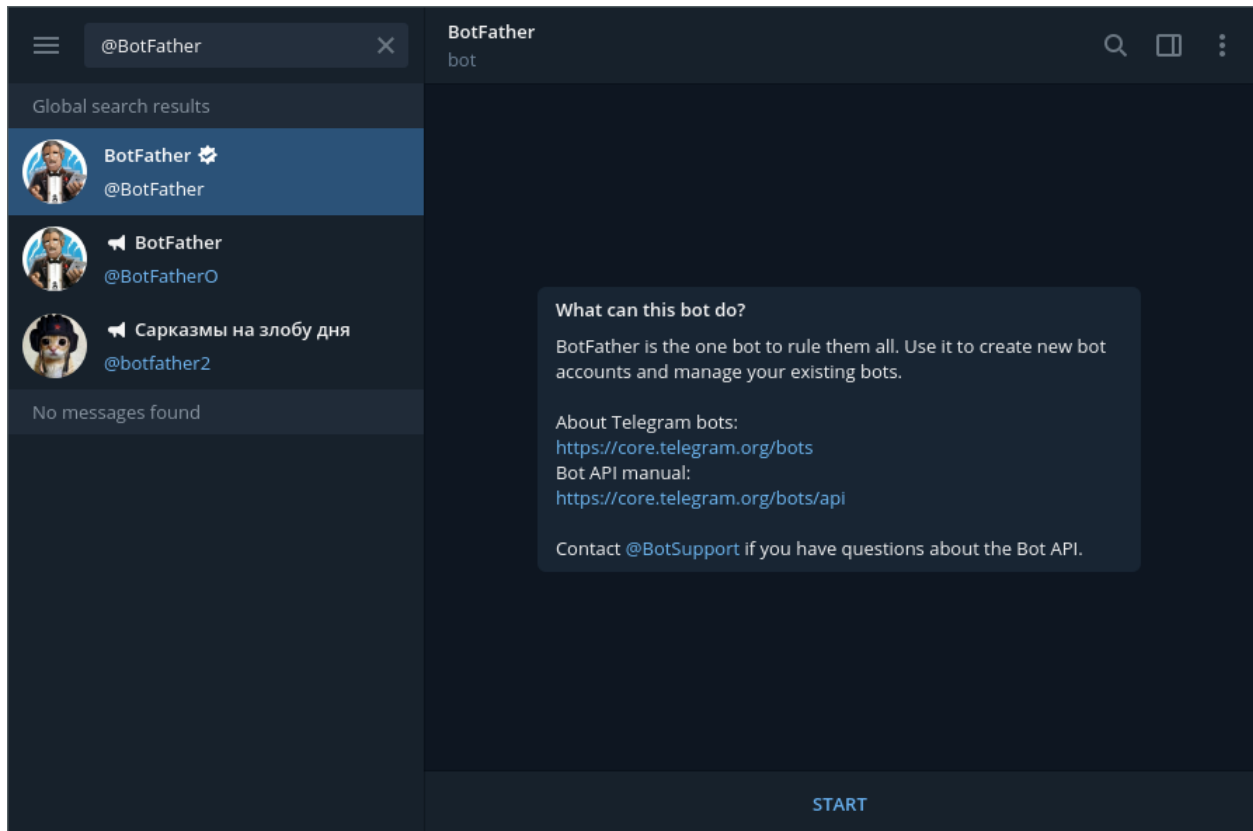
Danger: If you are using Telegram for authentication to FreeField, you should not re-use the same bot for webhooks. If your authentication bot's bot token is compromised, it could lead to account compromise as well as disruption, destruction, or full takeover of your FreeField installation. Create a separate bot for webhooks instead. See [Telegram authentication](#) docs for more information.

Note: Telegram webhooks can be slow, and may significantly increase the waiting time users experience when they submit a field research task that results in the webhook being triggered.

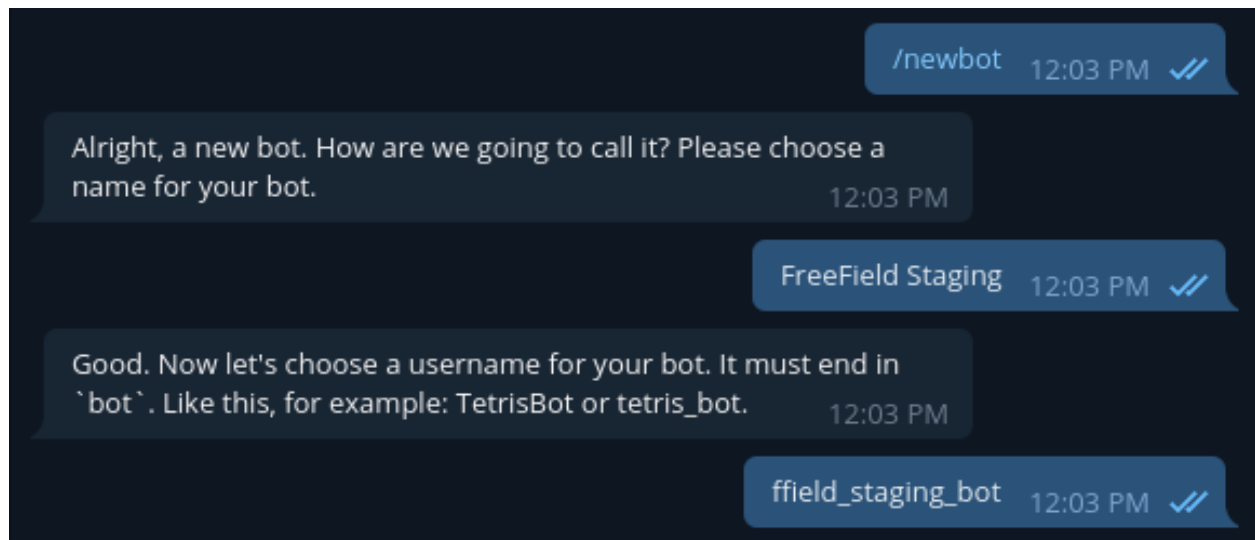
Creating a Telegram bot

A Telegram bot is required in order to set up message alerts in Telegram. To create a bot, please do the following:

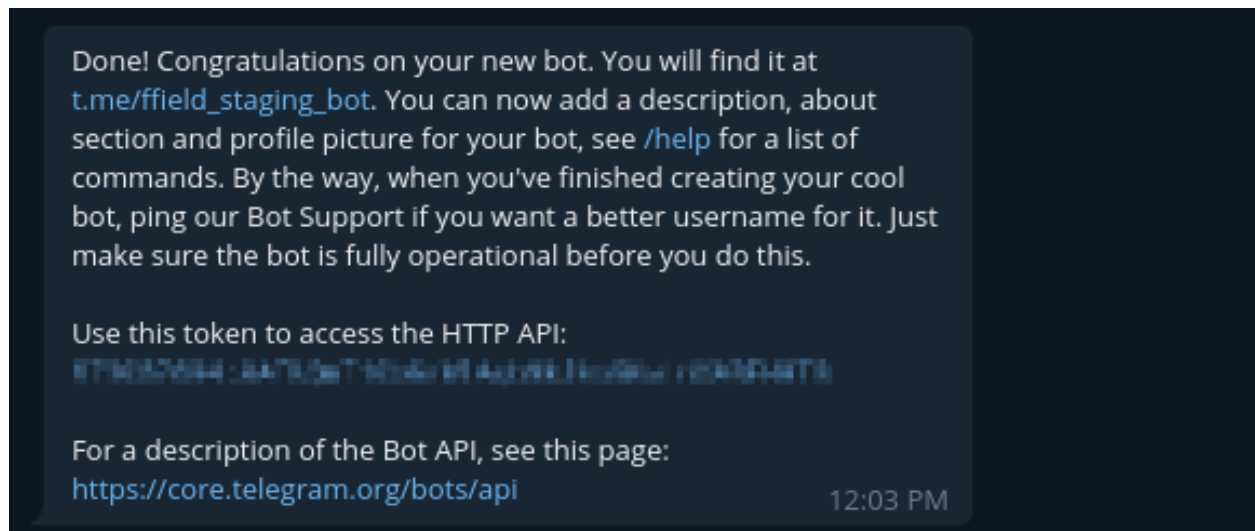
1. Search for the user @BotFather on Telegram and open a conversation with this bot, or [click here](#) to open a conversation directly. Click on the *Start* button at the bottom of the Telegram interface.



2. Issue the `/newbot` command in chat.
3. @BotFather will ask for a display name and username for your Telegram bot. Enter a display name and username. The display name can be anything, though the username must end with “bot”.



4. You should now be assigned a bot token for your bot by @BotFather. Note this token down and keep it safe.



Add the bot to your group

In order for FreeField to send messages to your group, you need to add the bot you just created to the Telegram group in which you want to receive field research alerts.

1. Open your Telegram group. In the top right corner of the window, look for the vertical ellipsis menu symbol. Click on it to open the menu, and select "Add members."
2. Enter the username you defined for your bot above. Click on the bot when it appears in the list, then click *Invite*.



Configuring the webhook on FreeField

1. In the “Webhooks” section on the FreeField administration pages, click on *Add new webhook*.
2. Select “Send Telegram message” as the webhook type, and optionally select a preset for the webhook payload. All of the listed presets are supported by the Telegram webhook type. You can see previews of each preset under *Supported presets*. Click *Done* to create your webhook trigger.
3. In the “Bot token” field, enter the bot token you were assigned by @BotFather above.
4. In the “Webhook URL” dropdown box, pick “Select group.”
5. FreeField will check which groups your bot has been added to. This might take a few seconds. When this is done, a popup will appear allowing you to select the group you want alerts sent to. Select your desired group, then click *Select*.

Note: If you cannot find the desired group in the list of groups, or if you get an alert that no groups could be found at all, FreeField was unable to determine its membership in your Telegram group(s). The most common causes of this are that no messages were sent in the group since your bot joined, or that it has been a long time since the last message was sent.

To fix this issue, issue the `/my_id` command in the Telegram group you would like to receive alerts in, then pick “Select group” again to re-search for group memberships.

6. Select the message format you would like to use when sending payloads to your Telegram chat. Telegram supports plain-text messages, as well as [messages formatted with Markdown and HTML](#).
7. Select whether or not you would like to disable link previews for messages sent by your webhook.

Tip: This can be useful to reduce clutter in your Telegram group.

8. Select whether or not you want members to receive notifications for messages sent by your webhook.

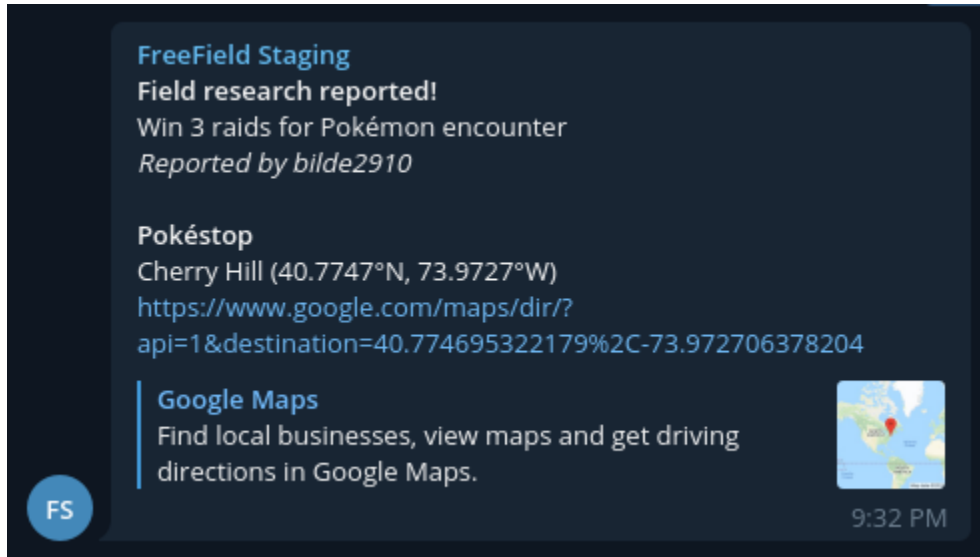
Hint: If you enable notifications, members of your group can still individually override this setting for your group by muting the group as a whole in their clients.

9. Configure the webhook’s settings and add filtering according to your own preferences.
10. Click on *Save settings* to save and activate your newly created webhook.

Supported presets

The following presets are supported by FreeField's Telegram webhooks:

telegram.md



8.3 Webhook properties

Once you have created your webhook, you have to specify additional properties to configure and customize its functionality. The main component is the webhook URL, but there are other properties as well, which are described below.

Hint: The “Send Telegram message” webhook type has additional properties which are not described in this section. These properties are described in detail on the [Telegram webhooks](#) subpage.

8.3.1 Webhook URL

The webhook URL is, along with the [Payload](#), the most important property of a webhook. The webhook URL is the location on the web that FreeField will post the payload to when the hook is triggered. This is typically an `http://` or `https://` URL, but may in certain specific circumstances be other protocols, such as `tg://` for Telegram webhooks.

Hint: When you create a webhook listener on an service, you will receive a URL that is associated with it. This URL should be put in the Webhook URL field on FreeField.

8.3.2 Language

You can choose which language should be used for your webhooks. The language you select will be used to localize various [Substitution tokens](#) in your payload. The language you choose will only be applied to these tokens, and not

strings of text that you define directly in your payload.

8.3.3 Icon set

The icon set you choose for your webhook is the icon set that will be used to generate icon URLs if you use any icon set image substitution tokens in your payload. Substitution tokens are explained in greater detail in [Substitution tokens](#), while implementation details specific to icon set image URLs can be found in the [Substitution token reference](#).

8.3.4 Species icons

To make it easier to spot particular rewards in webhook messages, FreeField configures webhooks to use species icons for webhooks when available by default. If enabled, an image representing a particular Pokémon species will be used instead of the default grass “encounter” icon when using reward icon substitution tokens, if the species rewarded by the research task is unambiguous and known. If the encounter species is ambiguous or unknown, the default encounter icon will be used instead. You can select which species icon set you want to use in the “Species icons” setting. This behavior can be disabled by unchecking the “Show icon for Pokémon” checkbox.

8.3.5 Geofence

FreeField webhook triggers support [Geofencing](#). If you select a geofence for your webhook, then the webhook will only be triggered if the Pokéstop that field research was reported for is within the bounds of the selected geofence. For information on defining geofences, please see [Geofencing](#).

8.4 Payload

The payload is, along with the [Webhook URL](#), the most important property of a webhook. When a webhook is triggered, the payload is what is sent to the webhook URL, and is the data that the receiving service will process to e.g. generate alerts. The syntax of the payload depends on the webhook type you have chosen, and the structure depends on the service whose webhooks you are targeting.

Hint: Refer to the documentation for [Common webhook targets](#), as well as the documentation available directly from your target webhook service, to learn how to properly structure your webhook’s payload.

8.4.1 Presets

FreeField has payload presets for several popular webhook target services. You get the option to select a payload preset in the dialog that appears when you first create your webhook. Presets are pre-defined payloads that, if one is chosen, will fill in the Payload field of your webhook with pre-written contents. This can save you from having to configure your webhook’s payload from scratch according to the documentation for the service that you are setting up a webhook for.

Not all presets work for all webhook service providers, so you should consult the documentation on [Common webhook targets](#) for information on which presets you can use for your target service.

Caution: If you wish to create your own presets, you should avoid storing them in the presets directory in FreeField. This directory is deleted and overwritten every time you update FreeField. If you have created a preset you feel others would find useful, you can submit an issue or pull request for it on the [issue tracker](#) on GitHub.

8.4.2 Substitution tokens

FreeField is very flexible in how you can structure your payloads. Once you have decided on a structure for your payload, you can use substitution tokens to actually insert data from FreeField into it. Substitution tokens are tags which are replaced with relevant values when the webhook is about to be triggered, and will update the payload for each webhook call to be specific to the field research that was reported.

To place a substitution token in your payload, enter the substitution token where you want it to appear. For example, if you are using Discord webhooks, and you want the title of the message sent from FreeField to contain the name of the Pokéstop that field research was reported at, and the footer to contain coordinates for the Pokéstop and the time at which the report was made, you could place the `<%POI%>`, `<%COORDS%>` and `<%TIME%>` substitution tokens in your webhook's payload like this:

```
1 {
2   "embeds": [{
3     "title": "Field research reported at <%POI%>",
4     "footer": {
5       "text": "Pokéstop coordinates: <%COORDS%>"
6     },
7     "timestamp": "<%TIME(c)%>"
8   }]
9 }
```

A reference of all available substitution tokens is available in a separate documentation page. Please refer to your desired article below.

Substitution token reference

This is a list of all payload substitution tokens available and implemented in FreeField. For information on implementing them into your payloads, please see [Substitution tokens](#) on the main webhooks documentation page.

Pokéstop information

These substitution tokens provide information about the Pokéstop on which field research was reported.

`<%POI%>` The name of the Pokéstop.

`<%LAT%>` The latitude of the Pokéstop.

`<%LNG%>` The longitude of the Pokéstop.

`<%COORDS%>` A coordinate pair in decimal degrees format (e.g. “42.63445°N, 87.12012°E”). You can specify the number of decimals in each coordinate by including it in parentheses directly after “COORDS”. For example, `<%COORDS (4) %>` will return the string “42.6344°N, 87.1201°E.”

Research task information

These substitution tokens provide information about the research task that was reported.

`<%OBJECTIVE%>` The research objective, for example “Catch 5 Pokémon.”

`<%REWARD%>` The research reward, for example “Pokémon encounter.”

You can also include information on the context of the report:

`<%REPORTER%>` The nickname of the user who reported the research task.

<%TIME (format) %> The exact time the report was received by FreeField. You have to specify a time formatting string when using this token. Replace *format* with a valid [PHP date\(\) string](#). For example, **<%TIME (Y-m-d H:i:s) %>** would result in a timestamp like “2018-10-02 15:38:55,” while **<%TIME (c) %>** would result in something like “2018-10-02T15:38:55+02:00.” Please refer to the aforementioned PHP manual for more format examples.

Navigation and links

<%NAVURL%> Inserts a link that users can click on to get turn-based navigation to the Pokéstop that research was reported on. By default, this uses the default navigation provider as configured in the “Map settings” sections of the administration pages in FreeField. You can specify that you wish use one particular navigation provider by passing it in parentheses directly after “NAVURL.” For example, **<%NAVURL (bing) %>** will override the default provider for navigation links, and instead create a link for navigation on Bing Maps. Valid navigation providers are *bing*, *google*, *here*, *mapquest*, *waze* and *yandex*.

<%SITEURL%> Inserts a link that users can click to visit the FreeField map.

Icon set images

Some services support displaying alerts with images and/or thumbnails (Discord is a good example of such a service). For these services, FreeField supports passing a URL that points to an image representing the reported research objective or reward.

<%OBJECTIVE_ICON (format, variant) %> Returns a URL to an image representing the reported field research objective.

<%REWARD_ICON (format, variant) %> Returns a URL to an image representing the reported field research reward.

Both of these tokens require a format and a variant. The format is the kind of image that should be returned - *vector* or *raster* - and which one you should use depends on what you will be using it for. Vectors, if present, are generally much clearer and scale better than raster (bitmap) images, but not all services support vector graphics. Raster images have much better compatibility, but they do not scale as well and will start looking pixelated or blurry if scaled too high. If you specify the *vector* format, but the *Icon set* you have chosen does not offer vector variants of its icons, *raster* will be used as a fallback.

The icon tokens also require a variant. This is either *light* or *dark* - you should choose the one that fits best with the context in which the icons are to be displayed.

Note: Not all icon sets have separate light and dark icons. If you use an icon set that uses the same graphics for both light and dark icons, then the icons returned by these substitution tokens will be the same regardless of which variant you have chosen.

Localization tokens

If the webhook triggers an alert in a chatroom in your community, you may want the message to contain some phrases that describe the research that was just reported. FreeField supports substitution of localization tokens for webhooks for this purpose. This allows FreeField to use placeholder values for various strings that are then localized to the correct language before the webhook is triggered.

<%I18N (token [, arg1 [, arg2 . . .]] %> Replaced by the localized value of *token*, passing the given *arg1* . . . *n* arguments to the localization function. E.g. **<%I18N (webhook.report_title) %>** is replaced

with “Field research reported!” Arguments can be other substitution tokens, e.g. `<%I18N(webhook.reported_by,<%REPORTER%>)%>` is resolved to “Reported by,” followed by the nickname of the user who reported field research.

For more information on what localization tokens and arguments are, please refer to [Internationalization](#) in the developer documentation.

Tip: Support for substitution of localization tokens was created in order for FreeField to support multiple languages out of the box for webhooks, by avoiding hardcoding English strings in payload presets. If you want to place strings like “Reported by” in your own payload, you could simply write out those strings in your native language directly in the payload, without using localization tokens, as you most likely won’t change the language of a webhook later. Even if you do, you could just manually replace the strings with matching strings in the other language. In most cases, this is less work overall than setting up your webhook payloads to be completely internationalized.

Advanced report context data

Substitution tokens for report context data can be useful if you need to dig deep into the internal workings of the reported research task.

Caution: These tokens extract data from FreeField at a much lower level than most other tokens, and as such, they might stop working or change behavior in future updates. If you use these tokens, pay particular attention to breaking changes announced in update changelogs when you update FreeField. These tokens may break even across stable updates.

`<%OBJECTIVE_PARAMETER(param[, index])%>` **`<%REWARD_PARAMETER(param[, index])%>`**

Returns the value of the given parameter `param` of the reported research objective or reward. Parameters can be any parameters listed in the developer documentation on research parameters. `index` is optional and used only to extract a particular value from the parameter data if the given parameter is internally represented by an array (please refer to the [List of parameters](#) to see if this is the case).

- If the requested parameter is not present in the reported research objective or reward, this token is substituted with an empty string.
- Otherwise, and if the parameter is not an array type, the value of the parameter is substituted in. `index` is ignored.
- If the parameter is an array type, and `index` is not specified, the substituted value will be the array joined together with commas (e.g. `[1, 2, 3]` becomes “1,2,3”).
- Otherwise, if `index` is specified, assuming `index = n` and `n = 1` is the first item (the array is one-indexed), if there is no n^{th} element in the parameter array (i.e. the index is out of bounds), an empty string is returned for the substitution.
- Otherwise, if there is an n^{th} index in the parameters array, the value at that index is returned for the substitution.

Example: `<%OBJECTIVE_PARAMETER(type, 2)%>` returns the 2nd reported Pokémon type in the reported field research objective. If the reported objective is “Catch 5 Water- or Grass-type Pokémon,” this substitution would return “grass.” If the objective is “Catch 5 Water-type Pokémon,” it would return an empty string since there is no 2nd reported Pokémon type. It would also return an empty string for objectives which do not offer the `type` parameter, such as “Catch 5 Pokémon.”

`<%OBJECTIVE_PARAMETER_COUNT(param)%>` **`<%REWARD_PARAMETER_COUNT(param)%>`** Returns the number of submitted entries for the given parameter `param` of the reported research objective or reward. Pa-

rameters can be any parameters in the [List of parameters](#) in the developer documentation on research data. The behavior of this substitution token is as follows:

- If the requested parameter is not present in the reported research objective or reward, this token is substituted by 0.
- Otherwise, and if the parameter is internally represented by an array type (please refer to the [List of parameters](#) in the developer documentation to see if this is the case), this token is substituted by a number representing the size of that array.
- If the parameter is present and is not internally represented by an array, this token is substituted by 1.

Example: `<%OBJECTIVE_PARAMETER_COUNT (type) %>` returns the number of different Pokémon types in the reported field research objective. E.g. if the reported objective is “Catch 5 Water- or Grass-type Pokémon,” this substitution would return the number 2. If the objective is “Catch 5 Water-type Pokémon,” it would return 1. If it the objective does not offer the `type` parameter, such as “Catch 5 Pokémon,” 0 is returned.

Another example: `<%REWARD_PARAMETER_COUNT (quantity) %>` returns 1 if the reported field research reward has an associated quantity, E.g. if the reported reward is “3 Revives,” this substitution token would return the number 1. If the reported reward does not offer a quantity, such as for “Pokémon encounter,” 0 is returned.

Conditional substitution

`<%IF_EQUAL(expr, value, ifTrue[, ifFalse]) %>` **`<%IF_NOT_EQUAL(expr, value, ifTrue[, ifFalse]) %>`**

Checks whether `expr` is or is not equal to `value`. If true, the value of `ifTrue` is substituted in, otherwise, `ifFalse` is used. `ifFalse` is optional and defaults to an empty string.

`<%IF_LESS_THAN(expr, value, ifTrue[, ifFalse]) %>` **`<%IF_LESS_OR_EQUAL(expr, value, ifTrue[, ifFalse]) %>`**

Converts `expr` and `value` to floating-point numbers e and v , and evaluates $e < v$, $e \leq v$, $e > v$, or $e \geq v$ depending on your selected operation. If true, the value of `ifTrue` is substituted in, otherwise, `ifFalse` is used. `ifFalse` is optional and defaults to an empty string.

`<%IF_EMPTY(expr, ifTrue[, ifFalse]) %>` **`<%IF_NOT_EMPTY(expr, ifTrue[, ifFalse]) %>`**

Short-hand for `<%IF_EQUAL(expr, , ifTrue, ifFalse) %>`. Checks whether `expr` is or is not an empty string. If true, the value of `ifTrue` is substituted in, otherwise, `ifFalse` is used. `ifFalse` is optional and defaults to an empty string.

`<%FALLBACK(expr, fallback) %>` Short-hand for `<%IF_NOT_EMPTY(expr, expr, fallback) %>`. Checks whether or not `expr` is an empty string. If it is not, `expr` is substituted in, otherwise, `fallback` is used.

String manipulation

`<%SUBSTRING(string, start[, length]) %>` Returns a `length` long substring of `string` starting from the character index `start`. `length` is optional.

- If `start` is negative, the substring starts at `start` index of characters relative to the end of the string.
- If `start` is beyond the end of the string, an empty string is returned.
- If `length` is not provided, the returned substring will end at the end of the string rather than enforcing a particular substring length.
- If `length` is negative, the given number of characters will be cut from the end of the string.

`<%LENGTH(string) %>` Returns the length of the given `string`.

`<%PAD_LEFT(string,length[,padString])%>` `<%PAD_RIGHT(string,length[,padString])%>`
Left- or right-pads the given `string` to the given `length` using `padString`. If `padString` is not specified, " " (space) is used. E.g. `<%PAD_RIGHT(TestString,15,1)%>` will right pad "TestString" to 15 characters using "1" as the padding string, thus returning "TestString11111."

`<%LOWERCASE(string)%>` Converts the given input `string` to lowercase.

`<%UPPERCASE(string)%>` Converts the given input `string` to uppercase.

Tip: A quick reference of the most common substitution tokens are available directly from the webhook configuration section in FreeField. You can access it by clicking on *Show help* in the Payload section of your webhook.

8.5 Task-based filtering

By default, FreeField will trigger webhooks regardless of what kind of research is reported. However, you can restrict them to only being triggered when a particular combination of research objectives and rewards are reported. This is done using objective and reward filtering.

You can filter by both objectives and rewards in the same webhook. The webhook will only be triggered if both the objectives and rewards component of the filter are passed.

To add a filter for a particular type of objective or reward, click the + button next to the "Objectives" or "Rewards" sections underneath the webhook payload. A popup will appear that lets you select the type of objective or reward that you want to filter. Select one.

Most objectives and rewards have additional parameters, such as the quantity of items awarded by a reward, or the species of Pokémon that one must catch to complete an objective. When reporting research, it is mandatory to fill out all of these parameters, but when you set up webhook filters, these parameters can be omitted. If you wish to omit a parameter from the filter, simply uncheck the checkbox next to the parameter on the dialog box.

If a parameter has been omitted in the filter, FreeField will skip checking the reported research task against it when determining whether or not to trigger the webhook. For example, consider the following two reward filters; "3 Rare Candy" and "[n] Rare Candy." Both of these reward filters will be triggered by the "Rare Candy" reward, but while the former will only trigger if exactly three Rare Candy are awarded by a research task, the latter will be triggered regardless of the quantity of rare candies rewarded. Hence, the former reward filter will only pass if exactly three Rare Candies are rewarded, while the latter will also pass if the reported quantity is e.g. 1 or 5.

When you are done adding an objective or reward filter, click *Done*. The newly created filter will be added to the list of active filters for this webhook. You can add as many filters as you want, edit them whenever you wish, and delete them if you no longer want them.

8.5.1 Filtering modes

Webhooks have two task filtering modes - whitelisting and blacklisting. The default setting is whitelisting for both objectives and rewards.

If you choose the whitelisting mode, the webhook will only be triggered if any one of the given objective or reward filters match the reported research task. If you choose the blacklisting mode, the webhook will only be triggered if *none* of the filters match the reported research task.

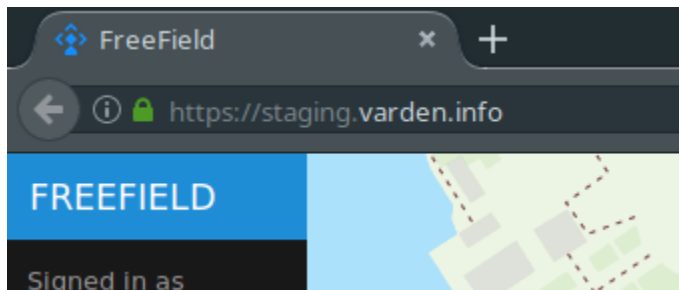
You can switch between them using the selection boxes at the top of the "Objectives" and "Rewards" filter sections of the webhook.

Site appearance

FreeField allows you to customize its visual appearance. This page will explain the various settings you can use to change FreeField's appearance.

9.1 Site name

You can change the site name from the “Site settings” section of the administration pages. The site name is displayed in the title bar of the browser on every page in FreeField. You can also change the sidebar header here - this is the text that appears in the blue field at the top of the sidebar on the map page, as well as on the administration pages. In the following screenshot, the site name is set to “FreeField” and the sidebar header set to “FREEFIELD.”



Note: There are no restrictions on what characters you can use in these strings, or on how long they can be. You should, however, pick a sidebar header string that is short enough to fit within its box in the sidebar.

9.1.1 Using an image in the sidebar

It is possible to use an image in the sidebar header instead of text. To upload an image, go to the “Site settings” section of the administration pages and look for the “Sidebar image” setting. Upload your preferred image, then select a sidebar header style underneath the file upload box that shows the image instead of a text header.

Note: Sidebar images should must be at least 130 px wide to ensure it is displayed clearly in 1:1-scaled browsers, though 400 px or higher is strongly recommended to ensure it also displays in high resolution on mobile devices. Images must be in PNG, GIF, JPEG or SVG format and must not exceed 128 KiB.

9.2 Message of the Day

You can set a Message of the Day, or MotD, that is displayed when users visit the FreeField map. You can configure this message to only display once, or to display every time a user visits. The MotD can be configured in the “Site settings” section of the administration pages.



Display mode Set how the message should be displayed.

Hint: If the MotD is enabled, users can manually view the message by clicking on “Show MotD” in the sidebar on the map page.

Custom message title By default, and if this field is left blank, the title of the MotD popup will display “Message of the Day.” You can override this default by entering a custom title here.

Message content The message that is displayed in the MotD message popup. Markdown formatting is supported.

Tip: A preview of the message will be displayed in the preview box below the input field. This allows you to preview the message before you save it.

9.3 Search indexing

By default, FreeField will tell search engines such as Google and Bing to not index your installation. You can change this under the “Crawling” header in the “Site settings” section of the administration pages.

The setting that changes search indexing behavior is “Robots policy.” The possible values are as follows:

Allow all indexation and crawling (all) Allows web spiders to crawl and index your page. If the spiders find links on the page, such as in the *Message of the Day*, they will attempt to follow those links and index the target pages as well.

Allow indexation, but do not follow page links (nofollow) Like the above setting, web spiders can crawl and index the page, though they will no longer attempt to follow links found on the page.

Deny indexation, but allow following page links (noindex) Web spiders may look for off-site links on the page that they can follow and index, though they will not index FreeField itself.

Deny all indexation and crawling (noindex,nofollow) Web spiders and crawlers will not index FreeField, and are additionally told not to attempt following and indexing links that are found on the page.

Note: This setting only applies to the FreeField index page, i.e. the map itself. All pages that require authentication are `noindex,nofollow` regardless of the setting you choose here.

Note: Not all web crawlers respect this setting. Non-compliant or malicious spiders may still attempt to visit and index your site even if you have denied them access using this setting.

9.4 Group labels and colors

User groups can be assigned non-default labels and colors that change how they appear in various places in FreeField. For more information on how to do this, see *Group settings*.

9.5 PWA theming

If you use Progressive Web Apps (PWA), you can change the appearance of the PWA loading screen, icons and names before enabling PWA. Please see *Progressive Web Apps (PWA)* for more information.

9.6 Favicon

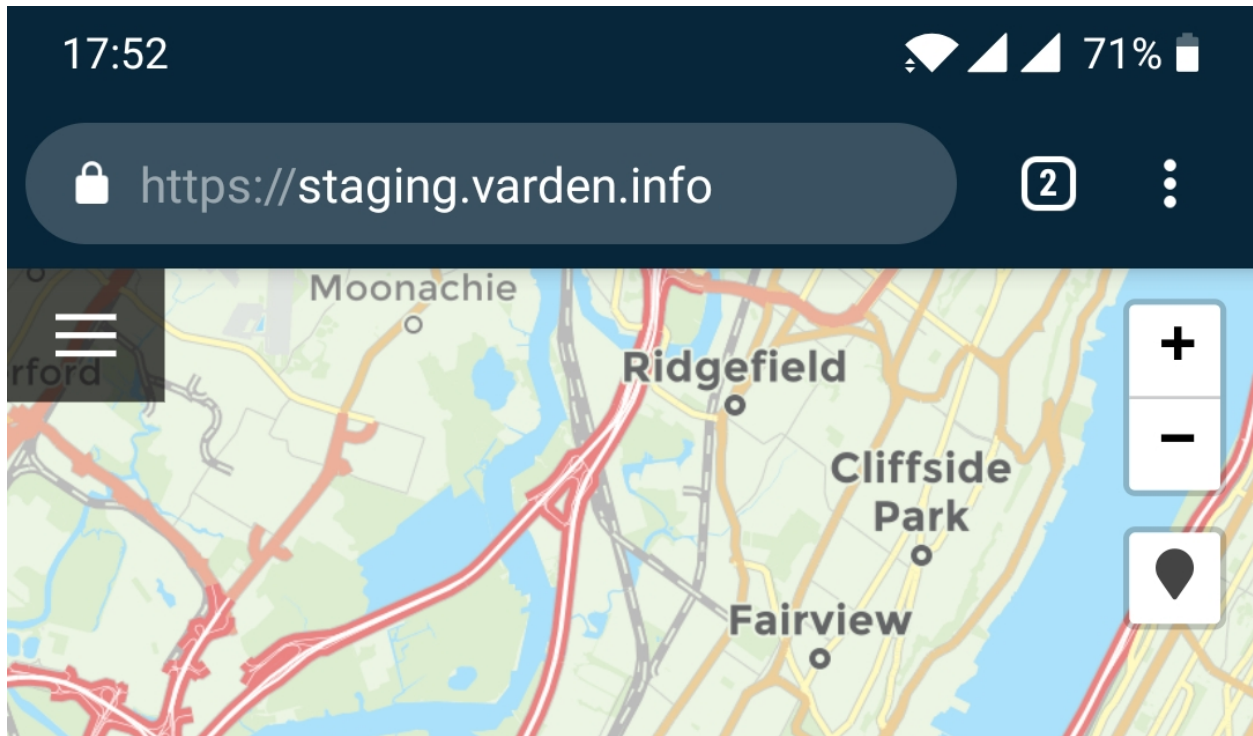
The favicon is the icon that is displayed in the address bar and in bookmarks when using FreeField. You can change this icon in the “Appearance” section of the administration pages.

Favicons should be PNG, GIF or ICO files and should have square dimensions.

9.7 Title bar theme color

Some mobile browsers (notably Chrome) support changing the color of the address bar when visiting FreeField. You can change this color in the “Appearance” section of the administration pages.

The following screenshot shows the “Title bar theme color” set to the default dark blue background, $r=8$, $g=38$, $b=58$, in Chrome on Android.



9.8 Site theme color

The default FreeField theme color is light blue, but this can be freely configured to any color. The theme color is used in the sidebar, for links, and on some buttons on the user interface.

Note: The default FreeField color is #1f8dd6 ($r=31$, $g=141$, $b=214$).

Tip: Choose a moderately light color to ensure UI elements have enough contrast and are shaded correctly. For reference, FreeField’s default color has brightness and saturation values of 84% and 86%, respectively.

9.9 UI and map themes

You can configure color themes for the user interface, as well as default map styles, in the “Appearance” section of the administration pages. The following settings are listed under the “Color theme” header on that section.

Administration interface The color theme, dark or light, used on the administration pages. This setting affects everyone browsing the administration pages, and cannot be set individually.

User settings default The color theme, dark or light, used for dialogs and the settings menu on the map page. Users can individually override this setting for their own devices unless the “Settings personalization” option is disabled.

Settings personalization If checked, users are allowed to select a different color theme than the default above for their own devices.

Default map provider theme The theme that is used for the map itself.

Tip: To see how each theme looks, either refer to the documentation for the map provider you are using, or simply test each map theme on the user settings for your own account. To do so, ensure that “Map personalization” below is enabled, then go to the main map page, click on “Settings” in the sidebar, select a different map theme, then click *Save settings*, and evaluate the appearance of each theme. FreeField comes with defaults that are chosen to be easily understood by users, and that work well in daylight conditions.

Map personalization If checked, users are allowed to choose a different map theme than the default above for their own devices.

9.10 Map markers

FreeField comes with one set of map markers by default. If you have several sets installed, you can choose which one is the default for your users, and whether or not users are allowed to choose a different set for themselves. This can be configured under the “Map markers” heading of the “Appearance” section in the administration pages.

When you select a map marker theme, you will get a preview of all the icons the icon pack contains.

You can choose to deny users their ability to choose their own icon set. Keep in mind that users will only be able to choose from the installed icon sets either way, and cannot upload or use their own custom icons on your site.

In addition to a set of map markers, you can select a set of species markers, which are displayed in place of the standard grass “encounter” icon when the species rewarded by a research task is unambiguous and known. When the species rewarded by a research task is ambiguous or unknown, the encounter icon is displayed instead. FreeField comes with one species set by default; this can be configured from the same place as the standard map markers under the “Map markers” heading of the “Appearance” section in the administration pages.

CHAPTER 10

Progressive Web Apps (PWA)

FreeField has built-in support for the Progressive Web Apps standard. When PWA is enabled in your FreeField installation, community members who browse your site will be given an option to add FreeField to their home screens as a shortcut. In practice, this makes it much easier for your users to find FreeField and report research quickly, rather than having to navigate to the website URL every time.

PWA support is disabled by default, but you can enable it from the “Mobile” section of the administration pages.

Warning: Once you’ve enabled PWA functionality, it can be very difficult to apply changes to the PWA settings later. Most clients typically cache the launch screen for a long time, and invalidating this cache can be difficult. This means that you should not enable PWA until you have made a final decision on the icon, color scheme, etc. of your app. See [Configuration](#) for more details.

10.1 How it works

When a user installs the PWA app on their device (see [Installation preview](#)), a shortcut is created on the home screen of the user’s device. This shortcut contains a downloaded loading page for FreeField. When the shortcut is opened, a loading page is displayed. The loading page has the word “Connecting” underneath the FreeField logo in a spinning loading icon (see the preview below). This is the “launch screen” of the PWA, and will open even if the device is offline.

The purpose of the loading page is to check whether or not the device has an Internet connection. It checks whether or not the device is reported online, and if not, waits for it to go online. Once the online status has been confirmed, the loading page will attempt to actually connect to FreeField. If it still cannot connect (for example, the server could be down, a firewall could be blocking the connection, or the device could be connected to a Wi-Fi network with a captive portal for authentication), it will try again at intervals of three seconds. As soon as the loading page gets a positive response from your FreeField instance that it can be reached, the user is redirected to the map page itself.

Fig. 1: The loading screen that is shown when the PWA app is opened. This screen will persist until the app has confirmed the user is online and able to connect to your FreeField instance.

This connection checking might take a second or two to complete in many cases, but the user experience improvement of not being presented user-unfriendly error messages, plus automatic reconnection attempts in case of failure, will in most cases greatly outweigh this minor delay. The loader will check for a connection immediately upon launch, meaning in most cases, the loading screen is displayed for less than a second.

After the loading page has been displayed, the user will be brought to the main page of FreeField, i.e. the map. Viewing the map in the PWA app (and interacting with FreeField in general) is functionally the same as using FreeField in a web browser. However, the app can be configured to hide the browser address bar and even navigation buttons (see [Configuration](#)), giving users the impression that they have installed FreeField as a stand-alone app. Users can also find and launch FreeField from the application launcher of their device.

10.2 Configuration

The appearance of your PWA app can be freely customized. The following settings are available for PWA apps - you can find these settings in the “Mobile” section of the administration pages.

10.2.1 Enable PWA

This checkbox turns PWA on and off. Once this box is checked, PWA setup prompts will appear in the browsers of your users’ devices when they next visit your site.

Note: Existing PWA installations will continue to work if you disable this setting, though users will not be able to reinstall the app if they delete it from their device.

Caution: Once you enable PWA, it will be difficult to change the other settings listed here, as most browsers cache the PWA setup permanently. For this reason, PWA is disabled by default. Please do not enable PWA until you have read through the behavior of all settings on this page, and are absolutely certain about your choice of PWA icons and colors.

10.2.2 Name

The name of the PWA application is a name that represents your PWA app. There are generally no length restrictions on this name.

Tip: You should choose a name that reflects the community you have set up FreeField for. A good idea is to use the name of your community, or a location-specific name such as “New York FreeField.”

Note: If you change this setting, it is only applied to new installations of the PWA app. Existing PWA users must uninstall the shortcut from their device and re-add the shortcut for this change to take effect.

10.2.3 Short name

A short version of the name you specified earlier. This is the string that is displayed underneath the launcher icon on the device’s app launcher and home screen.

Important: The name you choose here should be short, so it is not cut off (see e.g. the example screenshots in the [Installation preview](#) section below, where “NYC FreeField” was cut off and displayed as “NYC FreeFi...”). Aim for 10 characters or less to avoid text overflow.

Do not use the default “FreeField” string here! If a user is a member of several communities that use FreeField, and both of them use the same name and icon, they will be displayed identically, making it extremely difficult to tell them apart.

Note: If you change this setting, it is only applied to new installations of the PWA app. Existing PWA users must uninstall the shortcut from their device and re-add the shortcut for this change to take effect.

10.2.4 Description

A string that explains what the application is used for. You can set this to any description you wish, though keeping the default setting works just as well.

Note: If you change this setting, it is only applied to new installations of the PWA app. Existing PWA users must uninstall the shortcut from their device and re-add the shortcut for this change to take effect.

10.2.5 Display mode

PWAs in FreeField support four different display modes:

Fullscreen The app is displayed completely full-screen. Devices which have on-screen navigation buttons will hide those, and the bar at the top of the screen that displays time, battery status etc. will be hidden.

Warning: Selecting this display mode will negatively affect the usability of your site, especially for devices with on-screen navigation buttons. It is therefore strongly recommended that you do not choose this display mode.

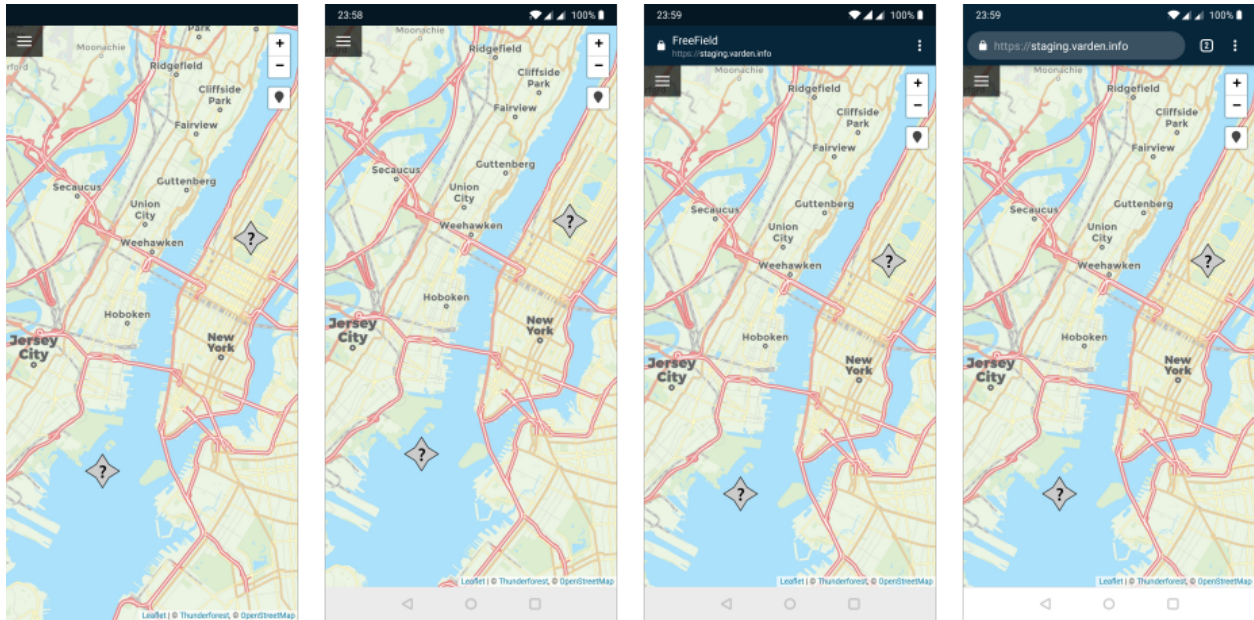
Standalone application The app will appear and function as if it was a standalone app on the user’s device. Basic navigation controls will be displayed for devices which have on-screen navigation buttons. The address bar of the browser will be hidden.

Standalone application with browser navigation controls The appears as if it was a standalone app, but more closely resembles a browser in that it may e.g. display the page URL and title and more browser navigation controls, such as a reload button. Users are not able to edit the URL in the address bar.

Conventional browser window The app acts a shortcut only, which simply opens the standard web browser. Users will be able to use the browser interface as usual.

Note: If you change this setting, it is only applied to new installations of the PWA app. Existing PWA users must uninstall the shortcut from their device and re-add the shortcut for this change to take effect.

Below is a side-by-side preview of the four different display modes, in the order they were presented above.



10.2.6 App icons

FreeField PWA apps use three different sizes or types of icons:

192px and 512px icons These icons are the small and big icons for your PWA app. Generally, the small icon (192px) is displayed as the icon for your app on user's home screens. You should use the same image for both of these icons, at their respective resolutions.

Attention: You are responsible for ensuring the icons you set are the correct resolution. The 192px icon should measure 192x192 pixels, while the 512px icon should have dimensions of 512x512.

Note: If you change this setting, it is only applied to new installations of the PWA app. Existing PWA users must uninstall the shortcut from their device and re-add the shortcut for this change to take effect.

Loading screen icon This icon is used as the icon displayed inside the loading spinner on the loading page for the app. A preview of the loading page with the default loading icon `default-pwa-launch.svg` is displayed in the [How it works](#) section above. If you replace this icon, the icon would appear where the FreeField logo appears in that preview.

Caution: This setting is cached in browsers. If you change this setting, it will not be applied for any devices that have at any point visited the FreeField site since PWA was enabled, *even if they reinstall the PWA app*. As such, ensure that you have made a final choice about the icon that you want to use before you enable PWA.

10.2.7 Background color

The background color you choose for PWA apps are used in two places. When the PWA app starts in some browsers, an interstitial screen is displayed, showing the full name and icon of the app. The background of this interstitial page, along with the background of the loading screen, will assume the color used here.

A preview of the loading page with the default dark blue background, $r=8$, $g=38$, $b=58$, is displayed in the *How it works* section of this page.

Caution: This setting is partially cached in browsers. If you change this setting, it will not be applied **on the loading screen** for any devices that have at any point visited the FreeField site since PWA was enabled, *even if they reinstall the PWA app*.

However, the background color used **on the interstitial page**, if applicable, will be updated if the user uninstalls and reinstalls the app.

As such, ensure that you have made a final choice about the color that you want to use before you enable PWA.

10.2.8 Foreground color

The foreground color you choose for PWA apps are used on the loading page. The color is applied to the “Connecting” string, as well as the spinning quarter circles around the loading screen icon. A preview is displayed in the *How it works* section of this page.

Caution: This setting is cached in browsers. If you change this setting, it will not be applied for any devices that have at any point visited the FreeField site since PWA was enabled, *even if they reinstall the PWA app*. As such, ensure that you have made a final choice about the color that you want to use before you enable PWA.

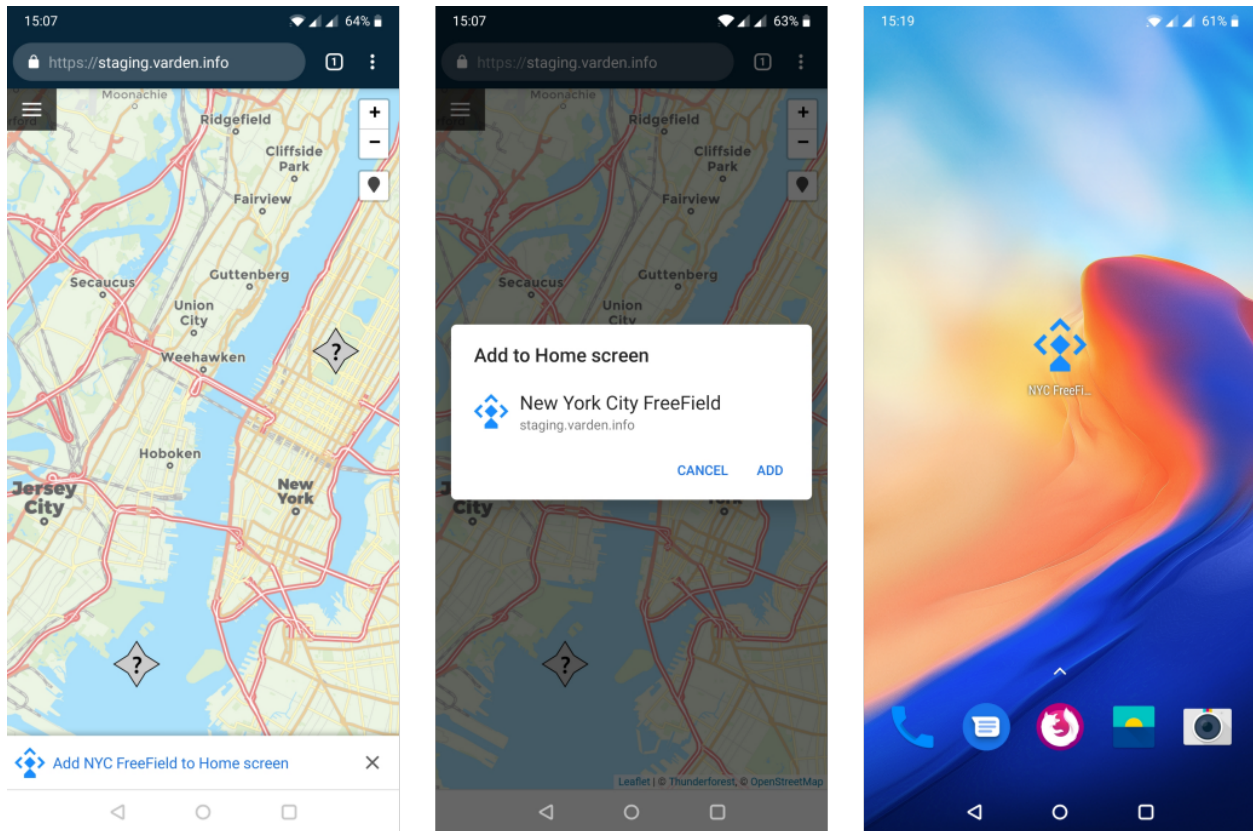
10.3 Installation preview

The installation steps for PWA apps for end users is different for each browser. The following previews show the process for Chrome and Firefox on Android.

10.3.1 Chrome installation process

When users visit your PWA-enabled website, they will be prompted to add a shortcut to their home screen through a banner at the bottom of the page. The name of the shortcut will be the short name you set in your configuration. If the user acknowledges the prompt, a popup will open, displaying the full name of the app, and prompting for confirmation that the user wants to add the shortcut. If the user accepts, a shortcut will be created, and after a few seconds, it is added in the first available slot on the user’s home screen.

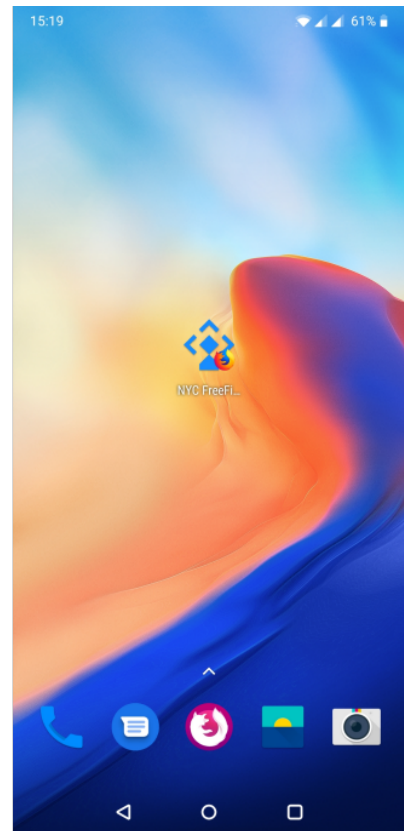
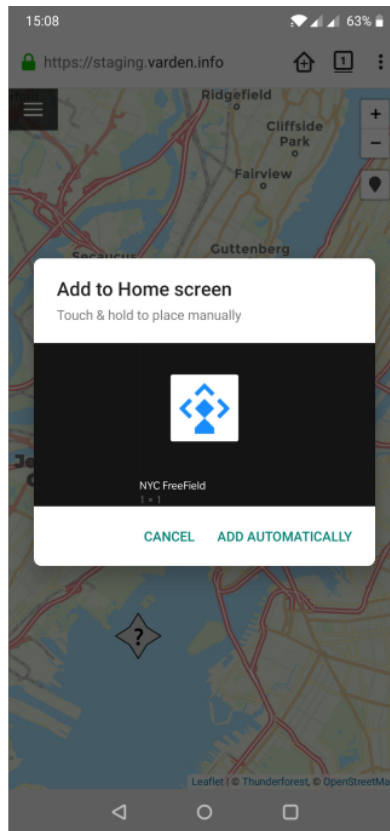
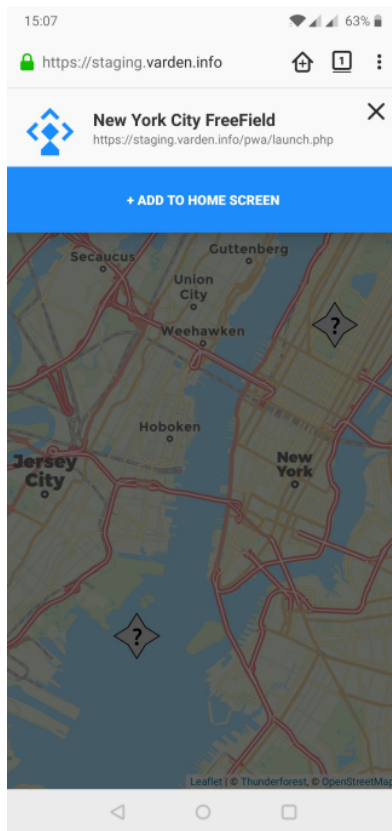
When the user opens the shortcut, the icon of the app will briefly be displayed along with its full name before the loading page is displayed. When the loader has verified that the user has an Internet connection, the user is redirected to the map, where they can view and submit research tasks.



10.3.2 Firefox installation process

When users visit your PWA-enabled website, there will be a house-like icon in the address bar that the user can tap on to add the icon to their home screens. Tapping this icon will display a banner at the top of the page showing the full name of your app, with an + *Add to home screen* button. When the user taps this button, they will be prompted to drag the FreeField icon to their home screens. Users can also have Firefox automatically position the icon on their home screens.

When the user opens the shortcut, the loading page opens. When the loader has verified that the user has an Internet connection, the user is redirected to the map, where they can view and submit research tasks.

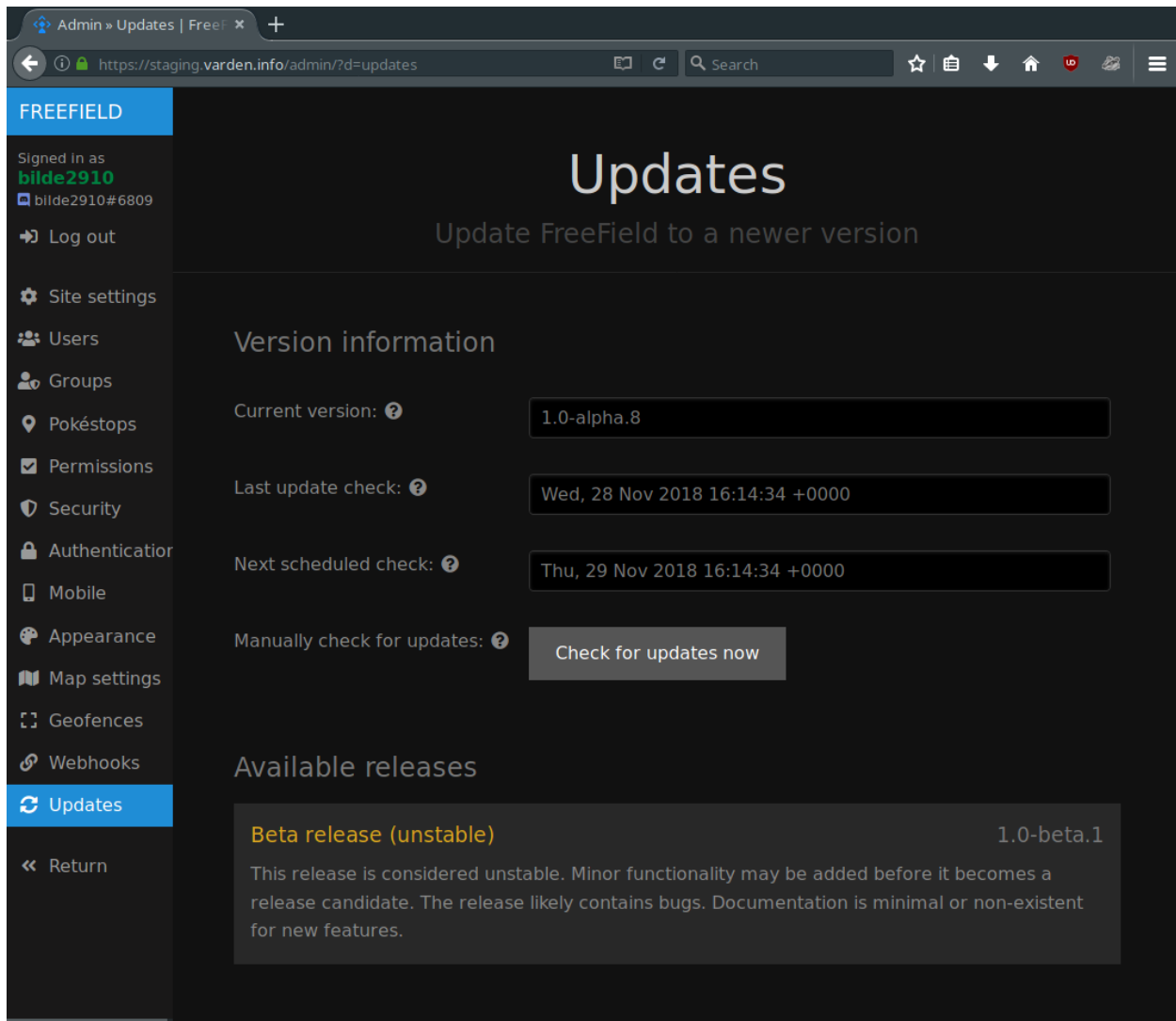


CHAPTER 11

Installing updates

The FreeField developers periodically release updates that improve the performance or add new features to FreeField. It is recommended that you always install stable updates when released. If you have installed FreeField using releases from GitHub, updates can be installed from the updates section of the administration interface. If you installed FreeField using `git clone`, you must update using `git pull` instead.

The updates section on the administration lists the current version, and any currently available versions, offering you to install them.



11.1 Update branches

FreeField offers several release branches that can be chosen when updating, ranging from alpha to stable. The alpha release will often contain the latest bleeding-edge releases very soon after they are added to the codebase, but may have significant bugs and can break unexpectedly. Once an alpha release reaches feature parity for its corresponding version milestone, it is pushed to beta. New beta releases will only contain minor updates and bugfixes. When beta versions are deemed stable for production use, they are pushed to the release candidate (rc) channel, and documentation will be written. Once documentation has been written, and any remaining reported bugs are fixed, the update is considered complete, and is pushed to the stable channel, prompting update alerts in all FreeField installations.

Minor updates may skip one or more release channels. Updates to the field research list, for example, are generally pushed directly to the stable branch, given their extremely small likelihood to cause issues, and time-sensitive nature of publication.

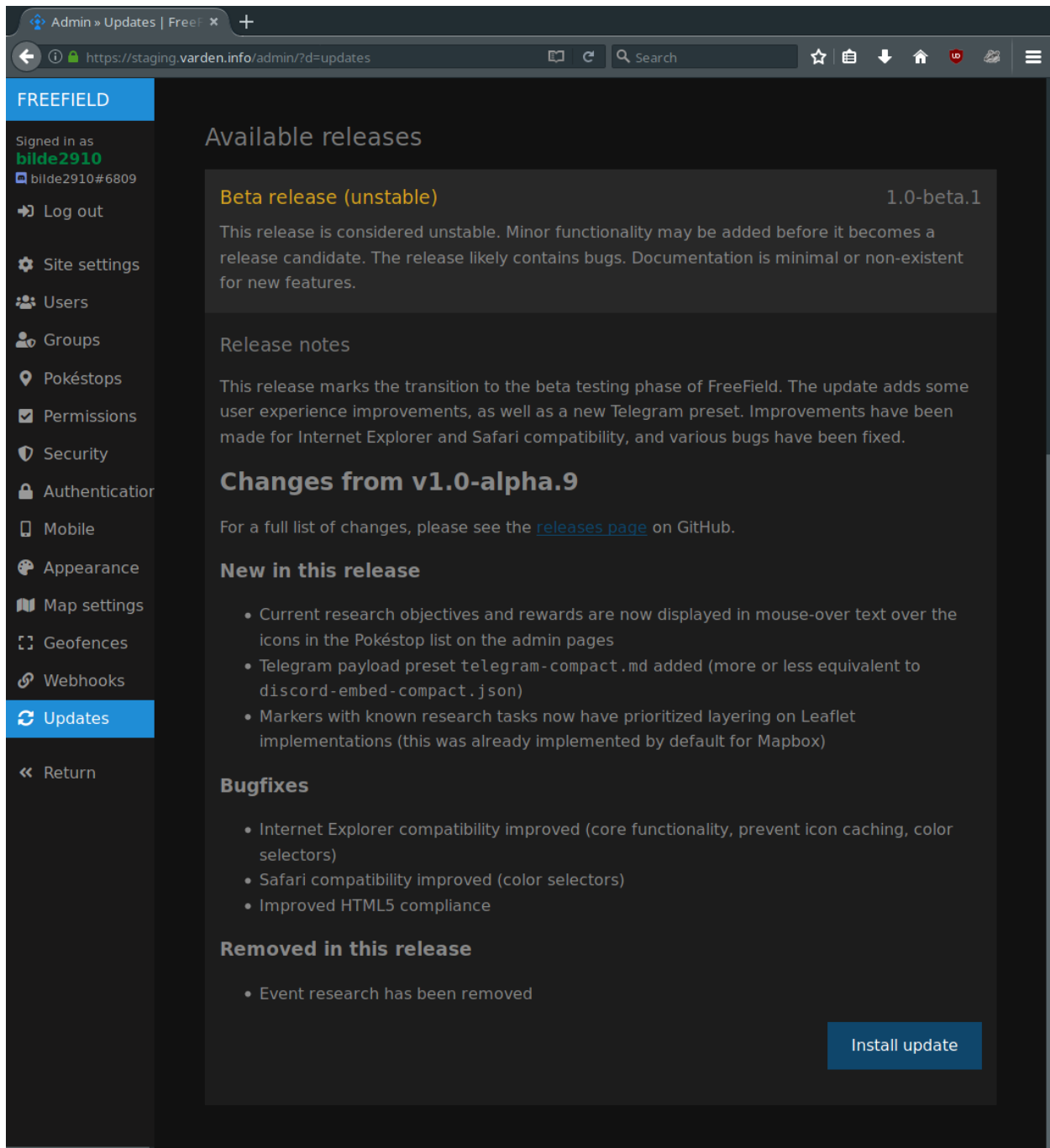
Most users are recommended to use the release candidate (rc) or stable branches for production usage. Administrators who wish to try out new features quickly may opt to use beta or even alpha release channels, offering new features sooner, at the cost of reduced stability.

11.2 Installing an update

Navigate to the updates section of the administration pages and click on *Check for updates now* to check for new updates. If an update is found, its release channel will be displayed in a list on that page. Click on your desired release to find changelogs for the update and a button to install the update.

Caution: The FreeField developers cannot guarantee the stability of any updates or the update process itself. If you proceed with the installation of any updates, including updates considered stable, you do so at your own risk and responsibility.

Attention: Read the changelog carefully before installing updates. Updates may contain breaking changes, and changelogs contain important information that may affect the way you use FreeField.



A window will pop up, asking you to confirm installation of the update. If you accept and initiate the update, FreeField will download and install the update automatically.

Danger: Once you have initiated an update, do not close the browser window until the installation is complete and you are told by the updater that it is safe to exit. **Failure to do so may result in corruption of your FreeField installation.**

A successful update will look like this (the exact steps displayed may vary):

Upgrading FreeField...

Please wait - this may take a while. Do not close this page.

```
Deleting temporary updatepkg.json... done
Ensuring FreeField root is writable... ok
Cleaning up previous upgrade attempt... ok
Opening file handler... done
Preparing to download update...
Downloading update... (0.00 B)
Downloading update... (0.00 B)
Downloading update... (0.00 B)
Downloading update... (0.00 B)
Downloading update... (0.00 B)
Downloading update... (0.00 B)
Downloading update... (0.00 B)
Downloading update... (0.00 B)
Downloading update... (3.45 MiB)
Download completed (downloaded 6.15 MiB).
Decompressing archive... done
Extracting archive... done
Verifying update scripts... ok
Checking dependencies...
Checking for curl... ok
Checking for gd... ok
Checking for HTTPS... ok
Determining if updater is broken... no
Warning: If the upgrade fails after this point, your installation may become irreversibly broken!
Removing current installation... ok
Installing upgrade... ok
Finalizing upgrade...
Cleaning up... ok
Installation completed.
```

[✓] Upgrade was successful!

Update completed. [Click here](#) to return to FreeField.

11.3 Troubleshooting

Installation of updates is a sensitive process, and errors have the potential to break your installation. Update failures can be classified into two categories.

11.3.1 Non-fatal errors

If the following warning message **does not appear** in the update log:

```
Warning: If the upgrade fails after this point, your installation may become
↪irreversibly broken!
```

then the updater script has experienced a **non-fatal error**, and your installation has not been changed in any way. You can try to simply re-install the update and see if the cause was intermittent. If it still does not work, you can try waiting for a while before redoing the update. Alternatively:

- Read the update log in the browser to determine if any checks fail, and fix the corresponding issues before retrying the update.

- Click on *Check for new updates* before retrying. FreeField caches the list of available updates, and if an updated release has been pulled from the repository for stability reasons, and FreeField uses a cached download link for this update that thus no longer works, the updater will not find the release file, causing the update to fail.
- Check that your server’s firewall allows outbound connections. FreeField connects to `api.github.com` over HTTPS (port 443/tcp) to check for and download update packages.
- Check that you have the latest CA certificate bundle installed on the server (e.g. package `ca-certificates` on Debian-based systems), or disable HTTPS validation in the “Security” section of the administration pages if connection errors persist (dangerous - not recommended).
- Search for issues on the issue tracker on GitHub, or create a new one, pasting the output of the update log in the issue to seek further help.

11.3.2 Fatal errors

If the following warning message **appears** in the update log:

```
Warning: If the upgrade fails after this point, your installation may become
↳irreversibly broken!
```

and the installation has **failed**, then the updater script has experienced a **fatal error**, and your installation may be corrupted. You will likely have to reinstall FreeField. Please thoroughly check whether your FreeField installation works before proceeding with the instructions below.

Danger: This is a disaster recovery procedure. If you follow the below instructions, please note that you do so **at your own risk** and that the FreeField developers are **not to be held responsible** for any damage caused to your system as a result of following these steps.

A basic understanding of your server’s operating system is required. Do not execute any commands listed here unless you are fully aware of their effects and accept any risk associated with executing them.

First, take a backup of your entire FreeField installation, using e.g.

```
user@host:/var/www/html$ tar czvpf /tmp/freefield-backup.tgz *
```

Then, remove all files and directories in the FreeField installation directory, except for the `includes` directory. Within `includes`, delete all files and directories except `userdata`. Download the latest version of FreeField and extract it on top of the old installation directory. When complete, your installation should contain an `includes/userdata` directory with configuration files already present. Attempt to access the FreeField installation again; it should now be in a working state.

If the installation does not work, clear the installation directory completely and restore the backup, using e.g.

```
root@host:/var/www/html# rm -rf *
root@host:/var/www/html# tar xzvf /tmp/freefield-backup.tgz .
```

Remove the same files as above, then download and install the version of FreeField you were updating from (not the latest version) in the same way as above.



Reporting field research

Reporting field research in FreeField is easy. When you report submit a report, the research task will be displayed on the map, allowing anyone to easily find tasks of interest. Administrators can also set up research alerts, automatically alerting other members in your community about interesting research tasks in your area.

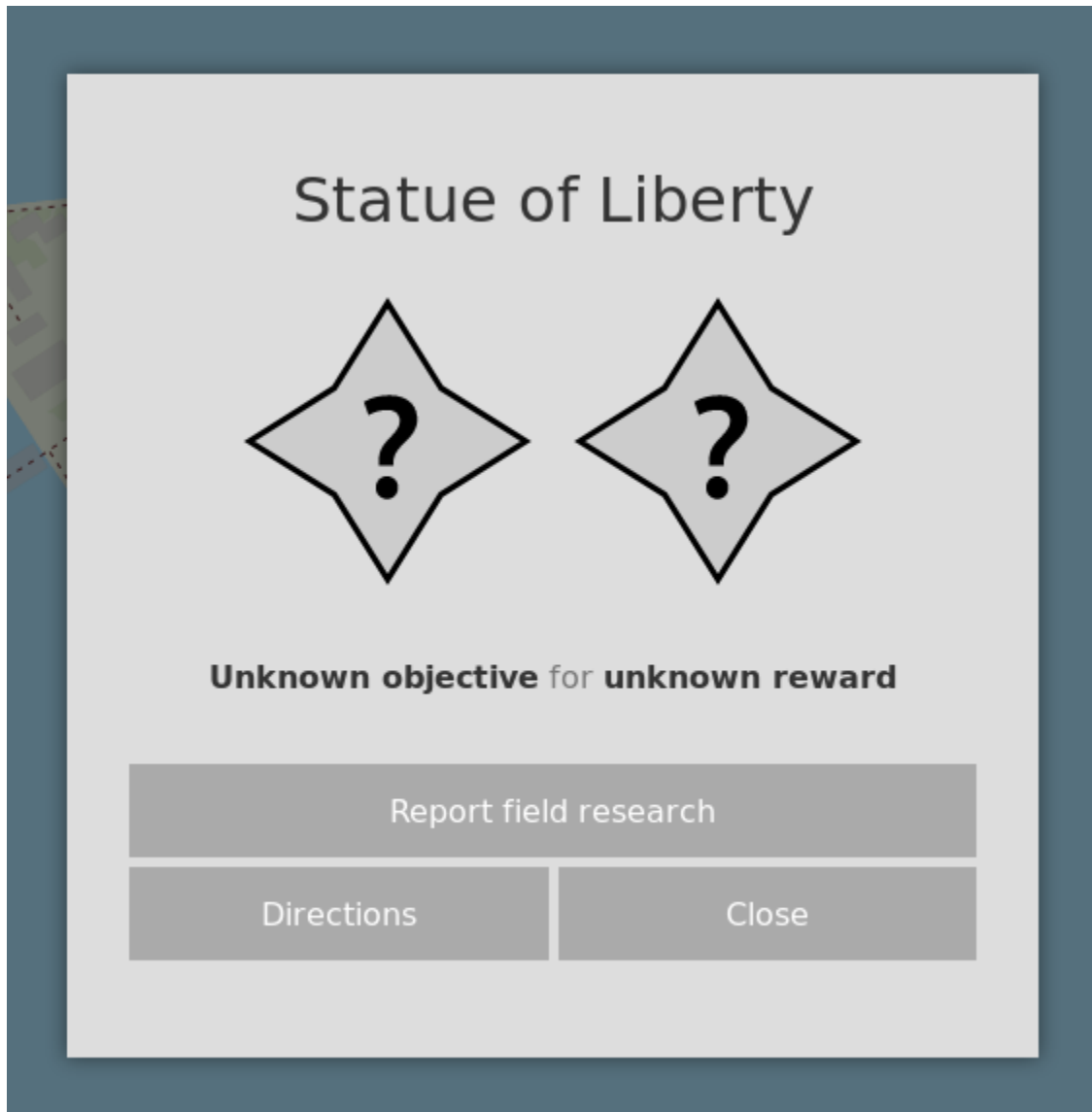
12.1 Submitting a report

1. Find the Pokéstop that you wish to report field research for on the map.

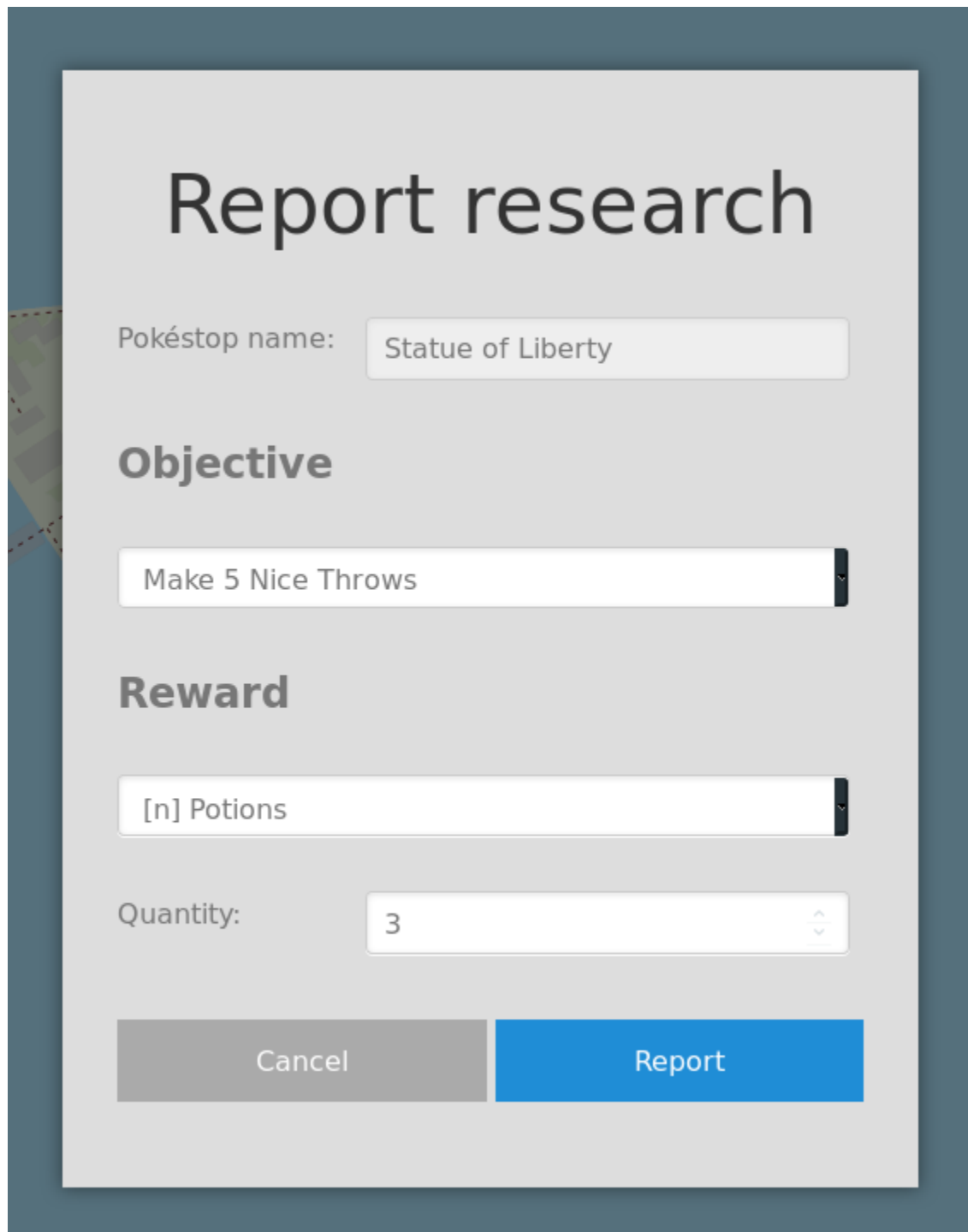


Tip: You can click on the geolocation button  or  to pan the map to your current location, letting you quickly locate Pokéstops near you.

2. Tap or click on the Pokéstop to open the Pokéstop details window. Click *Report field research*.



3. Select the research objective and associated reward. For example, to report the “Make 5 Nice Throws” objective with 3 Potions as the reward, the following can be entered. If the research objective you want to report is not listed, please see [Reporting unlisted research objectives](#) below.



Report research

Pokéstop name:

Objective

Reward

Quantity:

4. Click on *Report*. A confirmation banner should appear, and the map marker will be updated to reflect the new research task.



The next time someone clicks on the Pokéstop, the research task you just reported will be displayed. Research tasks are automatically reset at midnight.

12.2 Reporting unlisted research objectives

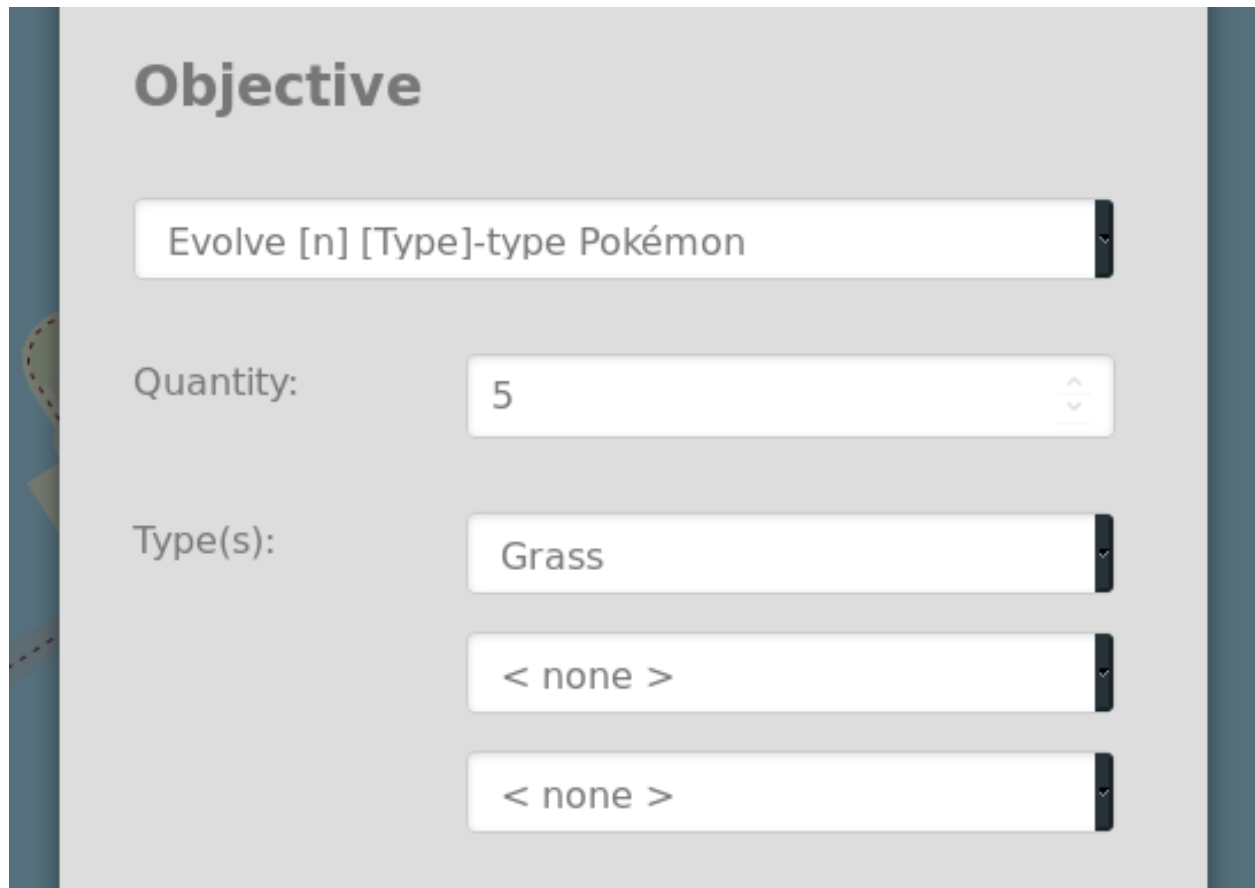
Sometimes, particularly after research task rotations, the research objective you want to report is not listed in the “Current objectives” section at the top of the list of reportable objectives. You can still report the research task, but you’ll have to fill in more details about the research objective manually.

In the list of research objectives, scroll down until you find the generic research objectives:

Objective

- Battle objectives**
 - Battle in a Gym [n] times
 - Win [n] Gym battles
 - Use a Super Effective Charged attack in [n] Gym battles
- Raid objectives**
 - Battle in a raid [n] times
 - Win [n] raids
 - Win [n] level [x] or higher raids
- Catch objectives**
 - Catch [n] Pokémon
 - Catch [n] Pokémon with Weather Boost
 - Catch [n] [Type]-type Pokémon

Select the research objective that matches the objective that you want to report. The objectives are grouped into categories, such as “Battle objectives,” “Catch objectives,” “Throwing skill objectives,” etc. Next, fill in the listed parameters. For example, to report the “Evolve 5 Grass-type Pokémon,” the following can be entered:



The screenshot shows a form titled "Objective" with a light gray background. On the left side, there is a vertical blue bar with a faint, stylized map of a region. The form contains the following elements:

- A text input field with the placeholder text "Evolve [n] [Type]-type Pokémon".
- A label "Quantity:" followed by a numeric input field containing the value "5".
- A label "Type(s):" followed by three stacked dropdown menus. The first dropdown menu is open and shows the selected option "Grass". The second and third dropdown menus are closed and show the placeholder text "< none >".

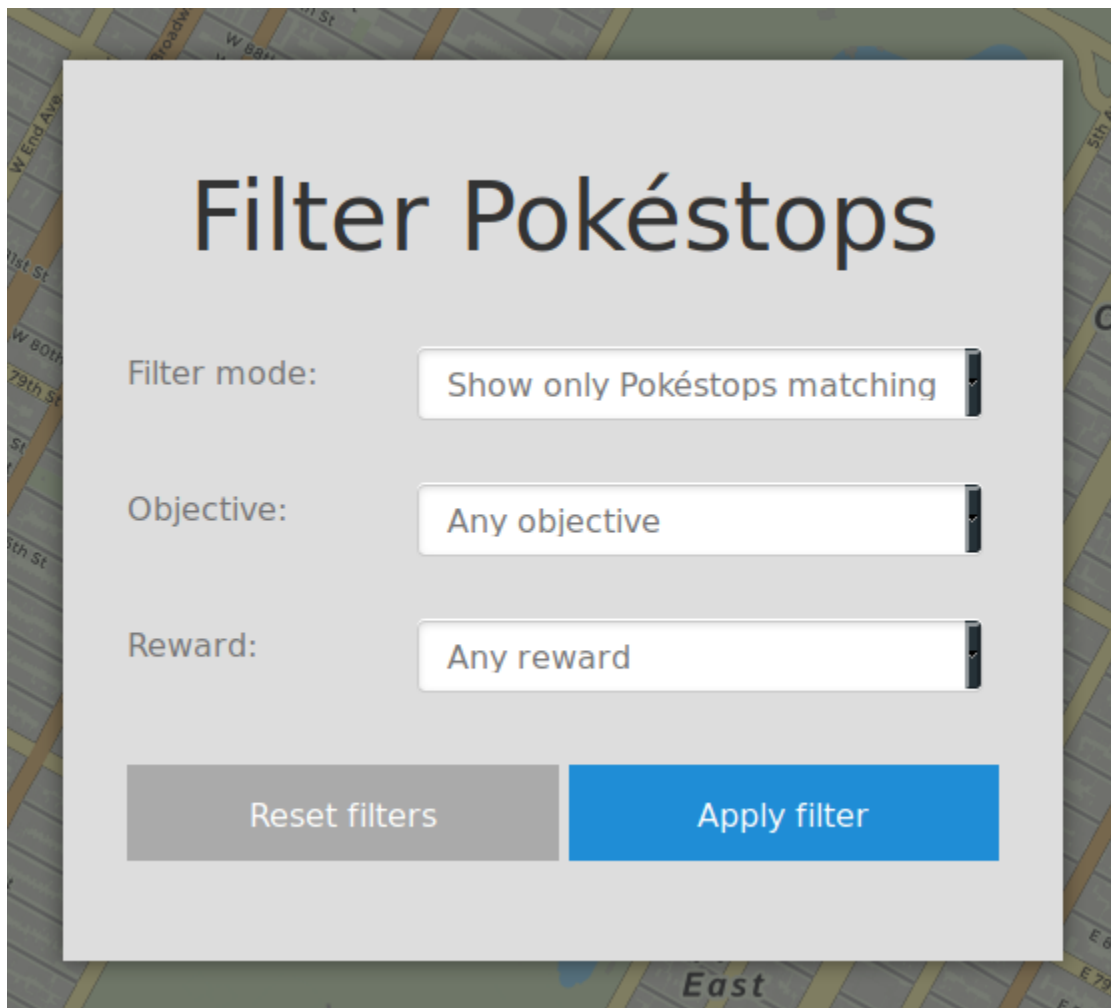
Next, fill in the reward as usual and submit the report by clicking *Report*.

CHAPTER 13

Filtering Pokéstops

If there are many Pokéstops visible on the map at the same time, it can be challenging to find a specific research task from all of the available tasks. To help with this, it is possible to filter the map so only Pokéstops with a certain type of research are available.

1. Click on “Filter” in the sidebar to open the Pokéstop filtering dialog:



2. Next to “Filter mode,” select how you wish to filter Pokéstops.
3. Next to “Objective,” specify the objective type you wish to filter. For example, if you wish to show only Pokéstops with the “Make 5 Great Curveball Throws in a row” objective, select “Make [n] Great Curveball Throws in a row.”

Note: Selecting an objective type will show all Pokéstops that match the selection. In the example above, this means that in addition to the “Make 5 Great Curveball Throws in a row” objective, Pokéstops with e.g. “Make 3 Great Curveball Throws in a row” will also be displayed.

4. Next to “Reward,” specify the reward type for the research task.

Tip: If you want to show a particular type of reward without care for the objective on the Pokéstop, you can choose “Any objective” from the Objective menu in the dialog. Combining this with e.g. “Rare candy” from the Rewards menu will show all Pokéstops that reward rare candies, regardless of the objective of the research task. This also works vice versa for objectives.

5. Click on *Apply filter*.

To reset the filters and show all Pokéstops, click on “Filters” in the sidebar to open the dialog window, then *Reset filters*.

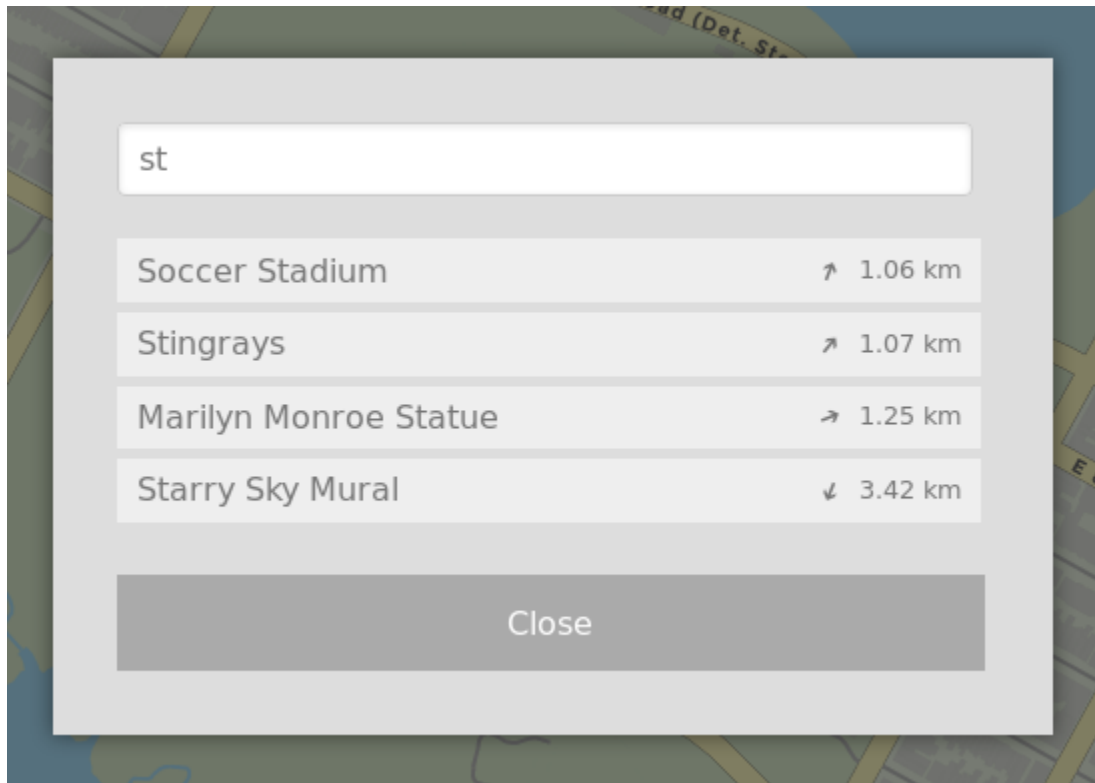
Tip: When filters are active, an additional filter icon will be visible underneath the sidebar hamburger menu (mobile only), and the “Filters” menu item will be highlighted in orange, to indicate that filters are active and that some Pokéstops are hidden as a result. You can click this filter icon to open the filters dialog directly without having to open the sidebar.

Note: Pokéstops that are hidden because of filters do not count against the hidden Pokéstops counter at the top of the page if clustering is in effect.

CHAPTER 14

Searching for Pokéstops

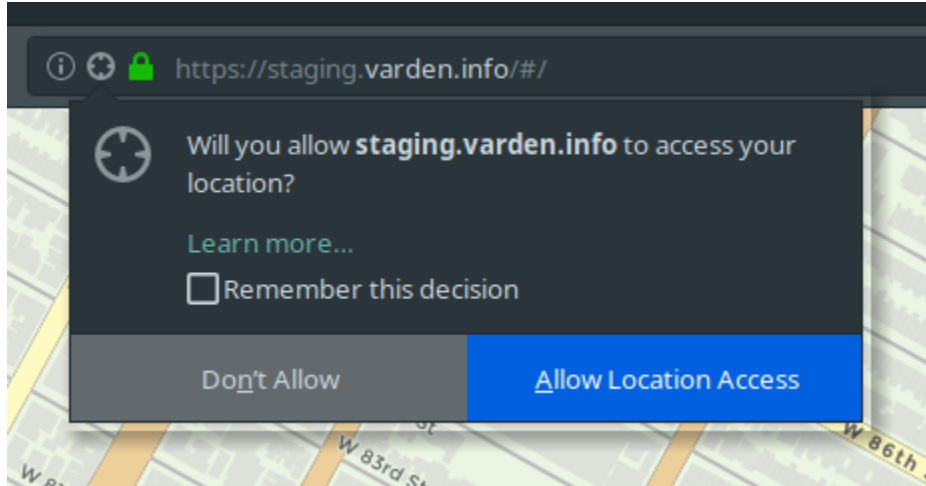
FreeField has a search function for finding Pokéstops on the map by name. Clicking on “Search” in the sidebar will open a search dialog where you can enter the full or partial name of any Pokéstop on the map. As you type, the list will update to show the best matches for your query.



The results list will show up to 10 Pokéstops, sorted by their relative distance to you. Each result will show the name of the Pokéstop, along with its distance and direction from your location. Clicking on any result will pan the map to the location of that Pokéstop.

Hint: Search requires the geolocation permission and a location fix in order to show results near you. Your browser may warn you and ask for permission to perform geolocation, which must be granted for this to work properly.

If FreeField cannot determine your location, search results will be sorted in descending alphabetical order, and show Pokéstop coordinates rather than their distances and directions.



Frequently asked questions

15.1 Why are some markers hidden from the map?

Starting from version 1.1, FreeField will start hiding markers when there are more than 100 markers in the visible area of the map in order to improve the map's performance. If you want to change this behavior, please do the following:

1. Open the settings menu from the sidebar.
2. Scroll down to the “Performance” section.
3. Next to “Maximum visible markers,” select how many markers you want to be visible on the map at the same time.
4. Click on *Save settings*.

Note: Increasing the number of markers on the map will cause the map to be sluggish when many of them are visible on the map.

FreeField has a RESTful API for third-party developers to interface with their instances. This document serves as the reference for this API.

16.1 Authentication

All requests to the API must be authenticated using an access token. To obtain an access token, go to the “API access” section of the administration pages and click on *Add new client*.

1. Define a name for your client. This name will be publicly visible when the client adds, modifies and deletes Pokéstops and other resources using the API. The client name is displayed in lieu of a user account’s nickname where such a nickname would otherwise be shown.
2. Optionally select a color for your client to accompany its name when displayed.
3. Click in the “Permissions” column for your client to define access controls for your client.
 - Check all permissions in the permissions list that your client needs to operate.
 - Set a reasonable permission level for the client’s capability of administering FreeField. For example, if your client has permission to manage and delete user accounts in your FreeField instance, the client can do so only for users up to and including the given permission level. The permission level is exclusively used for this purpose - permissions set on the “Permissions” section of the administration interface are overridden by the permissions specified for your client under “API settings.” The permission level setting is currently not in use, but will be used in a future version of FreeField.

Click on *Save settings* when you are done.

4. Click *Save settings* when you are done to generate an access token.
5. In the “Access token” column, click on *Click to view* to view and copy the access token for your client.

To authenticate your client against the API, set the X-Access-Token header of every request to the access token from step 5 above.

16.2 Response format

All requests to the API will return some kind of response. The response is sent as an object using `Content-Type: application/json`. Responses will use an appropriate HTTP response code depending on the success state of the request. Clients should treat all 2xx status codes as a successful request, all 4xx status codes as a client error, and all 5xx status codes as a server error.

Note: Requests resulting in a 204 No Content status code will not have a response body.

All responses that have a JSON body will have the field `ff_version` indicating the version number of the current FreeField installation. All error codes (4xx, 5xx) will additionally have a `reason` field containing an error code. Error codes and their meaning are explained in detail for each request type under *Method reference*.

16.3 Object reference

This is a reference of all objects referred to by the API documentation.

16.3.1 Object reference

This is a reference of all objects referred to by the API documentation.

POI object

An object which represents a single Pokéstop.

id The internal integer ID of the Pokéstop.

name The human-readable name of the Pokéstop.

latitude The latitude coordinate of the Pokéstop.

longitude The longitude coordinate of the Pokéstop.

objective An *Fully-defined objective object*.

reward A *Fully-defined reward object*.

updated An *Update object*.

Example

```
{
  "id": 2,
  "name": "Statue of Liberty",
  "latitude": 40.68925377062,
  "longitude": -74.044514894485,
  "objective": {
    "type": "battle_gym",
    "params": {
      "quantity": 1
    }
  }
}
```

(continues on next page)

(continued from previous page)

```

    },
    "reward": {
      "type": "pinap_berry",
      "params": {
        "quantity": 5
      }
    },
    "updated": {
      "on": 1553450414,
      "by": {
        "nick": "bilde2910",
        "color": "#008040"
      }
    }
  }
}

```

Fully-defined objective object

An object which represents all data about a particular research objective.

type A valid objective type, as defined in `objectives.yaml`. Example: `win_raid`.

params A key-value object representing data for the objective type specified in `type`. The list of parameters for each objective type are specified in the `objectives.yaml` file. See [List of parameters](#) for a complete list of all supported parameters.

Example

```

{
  "type": "win_raid",
  "params": {
    "quantity": 1
  }
}

```

Best-match objective object

An object which contains a string to be paired in the best possible way to a fully defined objective.

match A string containing a human-readable representation of the objective.

match_algo (*optional, default=2*) The algorithm to be used for pairing the objective:

- 1 Match against objectives found in `common-tasks.yaml` only. Matches very quickly, but can be inaccurate, particularly if attempting to match new objectives against an outdated common objectives list.
- 2 Match against all possible objectives defined in `objectives.yaml`. Highly accurate, but much slower than algorithm 1.

Example

```
{
  "match": "Make 5 Great Curveball Throws in a row",
  "match_algo": 2
}
```

Fully-defined reward object

An object which represents all data about a particular research reward.

type A valid reward type, as defined in [rewards.yaml](#). Example: *potion*.

params A key-value object representing data for the reward type specified in `type`. The list of parameters for each reward type are specified in the `rewards.yaml` file. See [List of parameters](#) for a complete list of all supported parameters.

Example

```
{
  "type": "encounter",
  "params": {
    "species": [
      56,
      66
    ]
  }
}
```

Best-match reward object

An object which contains a string to be paired in the best possible way to a fully defined reward.

match A string containing a human-readable representation of the reward.

match_algo (*optional, default=2*) The algorithm to be used for pairing the reward:

2 Match against all possible rewards defined in `rewards.yaml`.

Example

```
{
  "match": "3 Potions",
  "match_algo": 2
}
```

Update object

An object which contains details about when and who last updated something.

on A UNIX timestamp representing the time of update.

by A *User object*.

Example

```
{
  "on": 1553450212,
  "by": {
    "nick": "bilde2910",
    "color": "#008040"
  }
}
```

User object

An object which contains data about a user.

nick The nickname of the user.

color The display color of the user, as determined by group membership.

Example

```
{
  "nick": "bilde2910",
  "color": "#008040"
}
```

Location object

An object that specifies a particular location.

latitude The latitude coordinate of the location.

longitude The longitude coordinate of the new location.

Example

```
{
  "latitude": 40.68925377062,
  "longitude": -74.044514894485
}
```

16.4 Method reference

This is a reference of all methods available to call in the API.

16.4.1 Managing Pokéstops

Listing all Pokéstops

Method	GET
URL	/api/poi.php
Content-Type	application/json
Since	v1.1

Arguments

updatedSince (*optional*) A UNIX timestamp, or a delta-time difference in seconds.

Examples:

- 1546300800
Lists all Pokéstops that were updated since midnight on Jan 1, 2019.
- -120
Lists all Pokéstops that were updated in the last two minutes.

Response

ff_version The version of the FreeField instance.

pois An array of *POI object*.

id_list An array of integers representing the IDs of all Pokéstops on the map.

Example

Request

```
GET /api/poi.php?updatedSince=1546300800 HTTP/1.0
X-Access-Token: <access_token>
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "ff_version": "1.1-rc.1",
  "pois": [
    {
      "id": 1,
      "name": "Central Park",
      "latitude": 40.781386328336,
      "longitude": -73.967599868774,
```

(continues on next page)

(continued from previous page)

```

        "objective": {
            "type": "catch_type",
            "params": {
                "type": [
                    "electric",
                    "normal",
                    "poison"
                ],
                "quantity": 5
            }
        },
        "reward": {
            "type": "encounter",
            "params": {
                "species": [
                    56,
                    66
                ]
            }
        },
        "updated": {
            "on": 1553450212,
            "by": {
                "nick": "bilde2910",
                "color": "#008040"
            }
        }
    },
    {
        "id": 2,
        "name": "Statue of Liberty",
        "latitude": 40.68925377062,
        "longitude": -74.044514894485,
        "objective": {
            "type": "unknown",
            "params": []
        },
        "reward": {
            "type": "unknown",
            "params": []
        },
        "updated": {
            "on": 1550957320,
            "by": {
                "nick": "bilde2910",
                "color": "#008040"
            }
        }
    }
],
"id_list": [
    1,
    2
]
}

```

Errors

access_denied Permission has not been granted to your client, or you are not properly authenticating with the API.

database_error A server-side issue is preventing this request from being fulfilled.

Reporting field research

Method	PATCH
URL	/api/poi.php
Content-Type	application/json
Since	v1.1

Arguments

Pokéstop identifier Must one of the following sets of arguments. The first found identifier is used among those listed below.

Match by ID Matches exactly against a single, well-known Pokéstop using its ID.

id The unique numerical ID of the Pokéstop in FreeField's database.

Match by location Matches against the Pokéstop that is closest to the given location.

latitude The latitude coordinate of the Pokéstop.

longitude The longitude coordinate of the Pokéstop.

Match by name Attempts to find the best matching name out of all Pokéstops on the map.

name The name of the Pokéstop.

match_exact (*optional, default=false*) `true` if the name must be matched exactly, `false` otherwise.

match_case (*optional, default=true*) `true` if the name is case sensitive, `false` otherwise.

objective A *Fully-defined objective object* or a *Best-match objective object*.

reward A *Fully-defined reward object* or a *Best-match reward object*.

Response

Empty.

Example

Request

```
PATCH /api/poi.php HTTP/1.0
X-Access-Token: <access_token>
Content-Type: application/json
```



```
{
  "name": "Statue of Liberty",
  "objective": {
    "match": "Make 5 Great Curveball Throws in a row",
    "match_algo": 2
  },
  "reward": {
    "match": "3 Potions"
  }
}
```

Response

```
HTTP/1.1 204 No Content
```

Errors

access_denied Permission has not been granted to your client, or you are not properly authenticating with the API.

database_error A server-side issue is preventing this request from being fulfilled.

invalid_data The objective or reward objects you provided are malformed.

match_mode_not_implemented You specified an invalid match_algo for the objective or reward.

missing_fields You're missing either a Pokéstop identifier, an objective and/or a reward.

no_poi_candidates Your Pokéstop identifier did not match any Pokéstops in the database (e.g. supplying the ID of a Pokéstop which does not exist, or has been deleted).

poi_ambiguous Your Pokéstop identifier matched several Pokéstops equally well. A list of POI IDs for these are provided in `candidates` alongside this error response.

Clearing research from Pokéstop

Method	PATCH
URL	/api/poi.php
Content-Type	application/json
Since	v1.1

Arguments

Pokéstop identifier Must one of the following sets of arguments. The first found identifier is used among those listed below.

Match by ID Matches exactly against a single, well-known Pokéstop using its ID.

id The unique numerical ID of the Pokéstop in FreeField's database.

Match by location Matches against the Pokéstop that is closest to the given location.

latitude The latitude coordinate of the Pokéstop.

longitude The longitude coordinate of the Pokéstop.

Match by name Attempts to find the best matching name out of all Pokéstops on the map.

name The name of the Pokéstop.

match_exact (*optional, default=false*) `true` if the name must be matched exactly, `false` otherwise.

match_case (*optional, default=true*) `true` if the name is case sensitive, `false` otherwise.

reset_research Flag that specifies that you want to clear research for this Pokéstop. Set to any value, `true` is a reasonable choice.

Response

Empty.

Example

Request

```
PATCH /api/poi.php HTTP/1.0
X-Access-Token: <access_token>
Content-Type: application/json
```

```
{
  "name": "Statue of Liberty",
  "reset_research": true
}
```

Response

```
HTTP/1.1 204 No Content
```

Errors

access_denied Permission has not been granted to your client, or you are not properly authenticating with the API.

database_error A server-side issue is preventing this request from being fulfilled.

missing_fields You're missing a Pokéstop identifier or the `reset_research` flag.

no_poi_candidates Your Pokéstop identifier did not match any Pokéstops in the database (e.g. supplying the ID of a Pokéstop which does not exist, or has been deleted).

poi_ambiguous Your Pokéstop identifier matched several Pokéstops equally well. A list of POI IDs for these are provided in `candidates` alongside this error response.

Add a new Pokéstop

Method	PUT
URL	/api/poi.php
Content-Type	application/json
Since	v1.1

Arguments

name The name of the Pokéstop.

lat The latitude of the Pokéstop.

lon The longitude of the Pokéstop.

Response

ff_version The version of the FreeField instance.

poi A *POI object*.

Example

Request

```
PUT /api/poi.php HTTP/1.0
X-Access-Token: <access_token>
Content-Type: application/json
```

```
{
  "name": "Statue of Liberty",
  "lat": 40.68925377062,
  "lon": -74.044514894485
}
```

Response

```
HTTP/1.1 201 Created
Content-Type: application/json
```

```
{
  "ff_version": "1.1-rc.1",
  "poi": {
    "id": 2,
    "name": "Statue of Liberty",
    "latitude": 40.68925377062,
    "longitude": -74.044514894485,
    "objective": {
      "type": "unknown",

```

(continues on next page)

(continued from previous page)

```
        "params": []
    },
    "reward": {
        "type": "unknown",
        "params": []
    },
    "updated": {
        "on": 1550957320,
        "by": {
            "nick": "bilde2910",
            "color": "#008040"
        }
    }
}
```

Errors

access_denied Permission has not been granted to your client, or you are not properly authenticating with the API.

database_error A server-side issue is preventing this request from being fulfilled.

invalid_location The latitude and longitude coordinate pair you provided is outside the Pokéstop geofence. See [Limiting Pokéstop submission](#) for more information.

missing_fields You're missing a name, latitude or longitude.

name_empty The Pokéstop name you provided was empty.

Moving a Pokéstop

Method	PATCH
URL	/api/poi.php
Content-Type	application/json
Since	v1.1

Arguments

Pokéstop identifier Must one of the following sets of arguments. The first found identifier is used among those listed below.

Match by ID Matches exactly against a single, well-known Pokéstop using its ID.

id The unique numerical ID of the Pokéstop in FreeField's database.

Match by location Matches against the Pokéstop that is closest to the given location.

latitude The latitude coordinate of the Pokéstop.

longitude The longitude coordinate of the Pokéstop.

Match by name Attempts to find the best matching name out of all Pokéstops on the map.

name The name of the Pokéstop.

match_exact (*optional, default=false*) `true` if the name must be matched exactly, `false` otherwise.

match_case (*optional, default=true*) true if the name is case sensitive, false otherwise.

move_to A *Location object*.

Response

Empty.

Example

Request

```
PATCH /api/poi.php HTTP/1.0
X-Access-Token: <access_token>
Content-Type: application/json
```

```
{
  "name": "Statue of Liberty",
  "move_to": {
    "latitude": 40.68925377062,
    "longitude": -74.044514894485
  }
}
```

Response

```
HTTP/1.1 204 No Content
```

Errors

access_denied Permission has not been granted to your client, or you are not properly authenticating with the API.

database_error A server-side issue is preventing this request from being fulfilled.

invalid_data The latitude and longitude coordinate pair you provided is invalid.

invalid_location The latitude and longitude coordinate pair you provided is outside the Pokéstop geofence. See *Limiting Pokéstop submission* for more information.

missing_fields You're missing a Pokéstop identifier or the new location.

no_poi_candidates Your Pokéstop identifier did not match any Pokéstops in the database (e.g. supplying the ID of a Pokéstop which does not exist, or has been deleted).

poi_ambiguous Your Pokéstop identifier matched several Pokéstops equally well. A list of POI IDs for these are provided in `candidates` alongside this error response.

Renaming a Pokéstop

Method	PATCH
URL	/api/poi.php
Content-Type	application/json
Since	v1.1

Arguments

Pokéstop identifier Must one of the following sets of arguments. The first found identifier is used among those listed below.

Match by ID Matches exactly against a single, well-known Pokéstop using its ID.

id The unique numerical ID of the Pokéstop in FreeField's database.

Match by location Matches against the Pokéstop that is closest to the given location.

latitude The latitude coordinate of the Pokéstop.

longitude The longitude coordinate of the Pokéstop.

Match by name Attempts to find the best matching name out of all Pokéstops on the map.

name The name of the Pokéstop.

match_exact (*optional, default=false*) `true` if the name must be matched exactly, `false` otherwise.

match_case (*optional, default=true*) `true` if the name is case sensitive, `false` otherwise.

rename_to The new name for the Pokéstop.

Response

Empty.

Example

Request

```
PATCH /api/poi.php HTTP/1.0
X-Access-Token: <access_token>
Content-Type: application/json
```

```
{
  "name": "Statue of Liberty",
  "rename_to": "Giant Statue"
}
```

Response

```
HTTP/1.1 204 No Content
```

Errors

access_denied Permission has not been granted to your client, or you are not properly authenticating with the API.

database_error A server-side issue is preventing this request from being fulfilled.

missing_fields You're missing a Pokéstop identifier or the new name of the Pokéstop.

no_poi_candidates Your Pokéstop identifier did not match any Pokéstops in the database (e.g. supplying the ID of a Pokéstop which does not exist, or has been deleted).

poi_ambiguous Your Pokéstop identifier matched several Pokéstops equally well. A list of POI IDs for these are provided in `candidates` alongside this error response.

Deleting a Pokéstop

Method	DELETE
URL	/api/poi.php
Content-Type	application/json
Since	v1.1

Arguments

Pokéstop identifier Must one of the following sets of arguments. The first found identifier is used among those listed below.

Match by ID Matches exactly against a single, well-known Pokéstop using its ID.

id The unique numerical ID of the Pokéstop in FreeField's database.

Match by location Matches against the Pokéstop that is closest to the given location.

latitude The latitude coordinate of the Pokéstop.

longitude The longitude coordinate of the Pokéstop.

Match by name Attempts to find the best matching name out of all Pokéstops on the map.

name The name of the Pokéstop.

match_exact (*optional, default=false*) `true` if the name must be matched exactly, `false` otherwise.

match_case (*optional, default=true*) `true` if the name is case sensitive, `false` otherwise.

Response

Empty.

Example

Request

```
DELETE /api/poi.php HTTP/1.0
X-Access-Token: <access_token>
Content-Type: application/json
```

```
{  
  "name": "Statue of Liberty"  
}
```

Response

```
HTTP/1.1 204 No Content
```

Errors

access_denied Permission has not been granted to your client, or you are not properly authenticating with the API.

database_error A server-side issue is preventing this request from being fulfilled.

missing_fields You're missing a Pokéstop identifier.

no_poi_candidates Your Pokéstop identifier did not match any Pokéstops in the database (e.g. supplying the ID of a Pokéstop which does not exist, or has been deleted).

poi_ambiguous Your Pokéstop identifier matched several Pokéstops equally well. A list of POI IDs for these are provided in `candidates` alongside this error response.

Adding research data

This document will explain the steps behind adding field research data to FreeField.

17.1 Objectives

All available objectives in FreeField are stored in the `objectives.yaml` file. Objectives in this file appear in objective selection menus in the order they are presented in this file. When adding a new objective, should ensure that it is grouped with other similar objectives, i.e. don't just throw it at the end of the file. E.g. `win_raid` is placed underneath `battle_raid`, and together with `win_gym` and `battle_gym`, because these objectives are similar in nature.

The structure of a research objective entry is as follows:

```
objective_id:
  params:
    - param1
    - param2
    - ...
  categories:
    - main_category
    - parent_category
    - ...
```

17.1.1 Objective IDs

Each objective has a unique ID that is used to identify that specific objective internally. The objective ID should reasonably describe the objective that you are adding. Feel free to look at existing objective definitions in the file, and name your objective similarly to those that are already in the file.

For example, “Catch a Pokémon” has the ID `catch`. “Catch a Pokémon with Weather Boost” has the ID `catch_weather`.

All objective IDs are lowercase, with words separated by underscores.

17.1.2 Parameters

Many research objectives take one or more parameters. For example, `catch_type` (“Catch N X-type Pokémon”) takes two - the number of Pokémon to catch, and the type of Pokémon to catch. These parameters should be listed in the `params` key in the objective’s definition. If the objective takes no parameters, `params` is declared as an empty array like this:

```
objective_id:
  params: []
  categories:
    - main_category
    - parent_category
    - ...
```

You can only use parameters that have been implemented in FreeField. See [List of parameters](#) for a list of all parameters you can use.

If you use the `quantity` parameter, then that should be the last element in the parameters list, if multiple elements are present, for internationalization reasons.

17.1.3 Categories

Every objective has one or more assigned categories. Categories are used to group similar objectives together, and also serve as the declaration of which icon should be used to represent each objective on the map and elsewhere. Usage of categories is explained in [Icons and categories](#).

17.1.4 Internationalization

You must internationalize and localize to English any research objectives you submit to FreeField in the [localization files](#). These are located near the top of the files, underneath the species typing localizations. You must, as a minimum, add the tokens to the `en_US.ini` file, though if you speak another language, feel free to add them to the localization files of other languages you speak.

The tokens you need to add are:

```
objective.<objective_id>.singular = "Perform an action"
objective.<objective_id>.plural = "Perform {%n} actions"
```

The two tokens represent the singular and plural versions of the human-readable string that represents the research objective.

You need to add replacement tags for any parameters required by the objective. This is done by placing `{%n}` tags in the string at the locations that the parameters are to be substituted in, where *n* is chosen by the order in which the parameters are declared in the `params` field of the objective definition. For example, consider an objective that has defined its `params` array like this:

```
params:
  - species
  - type
  - quantity
```

In this case, `species` is assigned to `{%1}`, `type` to `{%2}`, and `quantity` to `{%3}`.

Note: Make sure you declare these internationalization tokens in the same place as the tokens declared for the other objectives. You can find them quickly by searching the file for “objective.”

17.2 Common objectives

In order to make it easy for users to report research, FreeField maintains a list of research objectives that are commonly reported, along with the specific Pokémon species those research tasks can award if reported with a Pokémon encounter reward. These objectives are stored in the `common-tasks.yaml` file.

An entry in the file may look like this:

```
-
  # Power up Pokémon 5 times
  type: "power_up"
  params:
    quantity: 5
  encounter_species:
    - 1 # Bulbasaur
    - 4 # Charmander
    - 7 # Squirtle
```

Each entry has a `type`, a list of `params` and an optional list of `encounter_species`.

type This is the ID of the objective that corresponds to the common task, as declared in the `objectives.yaml` file (see *Objectives* above).

params This is a list of parameters required by the objective specified in `type`. The value corresponds to the value of the parameter in the research objective. In the example above, the objective is `power_up` i.e. “Power up Pokémon [n] times” where $n = 5$, assigned to the `quantity` parameter, which is what is required by the definition of `power_up`.

encounter_species This field is optional, though should be present if the research task offers Pokémon encounters as a possible reward. If present, this is a list of one or more Pokémon species that are awarded upon completion of the research task.

Note: The species parameter accepts Pokédex numbers as input. To make it easier for readers to understand which species are required or awarded by the research task, you should comment the name of the Pokémon species after the Pokédex number (as seen in the example above with Bulbasaur, Charmander and Squirtle).

17.3 Rewards

All available rewards in FreeField are stored in the `rewards.yaml` file. Rewards in this file appear in reward selection menus in the order they are presented in this file. When adding a new reward, ensure that it is grouped with other similar rewards, i.e. don’t just throw it at the end of the file. E.g. `great_ball` is placed underneath `poke_ball`, and together with `ultra_ball`, because these objectives are similar in nature.

Categories of rewards should be ordered such that those rewards that most people are likely to report are placed at the top of the file. In FreeField, Pokémon encounters are at the top, while Rare Candies are second, because of their popularity and desirability for players.

The structure of a research reward entry is as follows:

```
reward_id:
  params:
    - param1
    - param2
    - ...
  categories:
    - main_category
    - parent_category
    - ...
```

17.3.1 Reward IDs

Each reward has a unique ID that is used to identify that specific reward internally. The reward ID should reasonably describe the reward that you are adding. Feel free to look at existing reward definitions in the file, and name your reward similarly to those that are already in the file.

For example, “Poké Balls” has the ID `poke_ball`. “Potions” has the ID `potion`.

All reward IDs are lowercase, singular, and with words separated by underscores.

17.3.2 Parameters

Many research rewards take one or more parameters. The majority of rewards offer some kind of quantity of itself - for example, `great_ball` (“Great Balls”) - the number of Great Balls that is awarded by a research task must be specified. These parameters should be listed in the `params` key in the reward’s definition. If the reward takes no parameters, `params` is declared as an empty array like this:

```
reward_id:
  params: []
  categories:
    - main_category
    - parent_category
    - ...
```

You can only use parameters that have been implemented in FreeField. See [List of parameters](#) for a list of all parameters you can use.

If you use the `quantity` parameter, then that should be the last element in the parameters list, if multiple elements are present, for internationalization reasons.

17.3.3 Categories

Every reward has one or more assigned categories. Categories are used to group similar rewards together, and also serve as the declaration of which icon should be used to represent each reward on the map and elsewhere. Usage of categories is explained in [Icons and categories](#).

17.3.4 Internationalization

You must internationalize and localize to English any research rewards you submit to FreeField in the [localization files](#). These are located underneath the objectives localizations and above the category localizations in the files. You must, as a minimum, add the tokens to the `en_US.ini` file, though if you speak another language, feel free to add them to the localization files of other languages you speak.

The tokens you need to add are:

```
reward.<reward_id>.general = "Item"
reward.<reward_id>.singular = "1 Item"
reward.<reward_id>.plural = "{%1} Items"
```

The two latter tokens represent the singular and plural versions of the human-readable string that represents the research reward. The former token, `general`, represents the singular version of the string, but without the number “1” that denotes quantity of rewards.

You need to add replacement tags for any parameters required by the objective. This is done by placing `{%n}` tags in the string at the locations that the parameters are to be substituted in, where *n* is chosen by the order in which the parameters are declared in the `params` field of the reward definition, in the same way they are chosen for objectives. In most cases, rewards only have the `quantity` parameter defined in their `params` arrays like this:

```
params:
- quantity
```

In this case, `quantity` is assigned to `{%1}`.

Note: Make sure you declare these internationalization tokens in the same place as the tokens declared for the other rewards. You can find them quickly by searching the file for “reward.”

17.4 Research parameters

Objectives and rewards are both declared with a list of required parameters. When an objective or reward definition lists a certain parameter in its `params` array, the user will be required to input data for that parameter when submitting research. Parameters are things such as the quantity of items rewarded by a task, or a particular species of Pokémon that must be e.g. evolved in order to complete a particular objective.

FreeField implements parameters using classes that define each parameter’s behavior. This includes how data is read from and displayed to the user, how the data for a parameter is stored in the database and configuration file, and input parsing and validation functions for data that is submitted by research reporters. All of these parameter classes are implemented in the `research.php` file.

Parameters must additionally be registered in the `PARAMETERS` constant in `research.php`, and internationalization tokens must be created for placeholders and labels. The sections below will assist you in setting up a class, registering it, and adding the required internationalization tokens.

17.4.1 Implementing a parameter class

When you implement a research parameter, it will become available for use as a requirement by objectives and/or rewards. Classes are placed in the `research.php` file. The definition of the `quantity` parameter follows below, and will be used to help you implement your own parameter by way of example.

```
/*
    Adds a number box to the field research box prompting the user for the
    quantity of items awarded in a reward/quantity of catches required for
    a catch quest, etc. This parameter is stored as an integer
*/
class ParamQuantity {
    public function getAvailable() {
```

(continues on next page)

(continued from previous page)

```

    return array("objectives", "rewards");
}
public function html($id, $class) {
    return '<p><input id="'. $id. '" class="'. $class. '" type="number" min="1"></p>';
}
public function writeJS($id) {
    return 'var val = parseInt($("#'. $id. '").val());
        if (isNaN(val)) return null;
        return val;';
}
public function parseJS($id) {
    return '$($("#'. $id. '").val(data);';
}
public function toString($data, $allParams) {
    return strval($data);
}
public function toStringJS() {
    return 'return data.toString();';
}
public function isValid($data) {
    return is_int($data) && $data >= 1;
}
}

```

The class must implement several required functions. These are explained in detail below.

getAvailable()

This function must return an array that has either one of, or both, of the following strings:

“objectives” If this string is in the returned array, this parameter will be made available as a parameter for research objectives. If not included, this parameter will not be possible to use with objectives.

“rewards” If this string is in the returned array, this parameter will be made available as a parameter for research rewards. If not included, this parameter will not be possible to use with rewards.

Hint: FreeField separates between objective and reward parameters to cut down on the size of the HTML response on the map. If a parameter is loaded for both objectives and rewards, it is rendered twice on the page - this does not matter much for parameters with a small output size, such as `quantity`, but for parameters with larger output sizes, e.g. `species`, this can have a big effect on the size and complexity of the output HTML. Therefore, ensure that you only declare the scopes that you actually need the parameter to be valid for - in most cases, it only has to be valid for objectives.

html(\$id, \$class)

This function must return an HTML string with input boxes that allow the user to input data for this parameter. For the `quantity` parameter, this is just a simple numerical input box. For `type`, this input consists of three rows of select boxes that allow the user to select a Pokémon type.

This function is passed two variables, `$id` and `$class`. You **must** use these variables as the element ID and class names of your **input box(es)/select box(es)** (i.e. *not* the parent `<p>` or other tags). If you have multiple input elements, you can append e.g. `“-1”`, `“-2”` etc. to the end of the ID. You can specify additional class names, but the classes provided by `$class` must be present, otherwise the input will not function correctly.

writeJS(\$id)

This function must return a JavaScript function body that fetches the value of the input elements returned by `html()` above and writes/parses it to the correct data format. The function should return this value, either as a string, an integer or floating point number, or as an array of any of these. If the returned value is `null` or an empty string, the value will be considered invalid and the user will not be allowed to submit research until the field has been populated with data. This can be used to e.g. check if an integer value is `NaN` and return `null` in that case.

The `$id` parameter is the same as the one that is passed to the `html()` function, and should be used to find and select the correct input element(s) to read data from.

The JavaScript function for `quantity` looks like this:

```
var val = parseInt($("#<ID>").val());
if (isNaN(val)) return null;
return val;
```

This function reads the value of the input box with the given ID and attempts to parse it to an integer. If it fails, it returns `null` (i.e. field is empty or invalid), otherwise it returns the value.

parseJS(\$id)

This function must return a JavaScript function body that does the inverse operation of `writeJS()`, i.e. it takes input data, parses it and fills it into the input boxes with the given `$id`.

The JavaScript function is passed a parameter `data` which contains the data in the same format as the function in `writeJS()` returns. Use this data to set the value of input boxes in the DOM.

The JavaScript function for `quantity` looks like this:

```
$("#<ID>").val(data);
```

It simply takes the `data` object, which is an integer, and sets it as the value of the input box with the given ID.

toString(\$data, \$allParams)

This function should convert the `$data` (which is in the format returned by `writeJS()`) to a human-readable string suitable for substitution into the localized string before being returned to the user. For example, the objective “Catch a [Pokémon Species]” (e.g. “Catch a Bulbasaur”) uses this function to return the actual string “Bulbasaur,” “Charmander,” etc. for substitution into the “Pokémon species” tag. If you return strings, these **must be localized by your implementation of this function**. The `quantity` parameter only needs to convert the integer `$data` to a string value, hence a simple `strval()` is sufficient. For a more complex example that includes localization, take a look at e.g. the [implementation for the species parameter](#).

The `$allParams` parameter is an array with all parameters passed to the given research objective or reward that your parameter is used for. The array can e.g. look like this:

```
$allParams = array(
    "quantity" => 5,
    "species"  => [1, 4, 7]
);
```

In most cases, you can ignore this parameter. It can be used to determine whether the singular or plural form of an internationalized string should be resolved. This is currently only used in practice in `encounter_item` through its parent class `ParamReward` (see [here](#)), to determine whether the singular or plural form of the name of the required item should be used, depending on the quantity of items required to use, through checking the `quantity` parameter.

toStringJS()

This function should return a JavaScript function body ported from the PHP code in `toString()` above. The function is passed `data` and the `allParams` object, and should return the same string representation as `toString()`. This function is used client-side, while `toString()` is used server-side for e.g. *Research task information* in web-hook payloads.

isValid(\$data)

This function should perform server-side input validation on `$data` before it is accepted and stored in the database. There is no guarantee that `$data` is of the same format as the object returned from `writeJS()`, hence you must ensure that the format is acceptable as well. The validation function of `quantity` checks whether or not the input data is an integer, and also checks if it is 1 or higher (it makes no sense to report 0 or a negative quantity of an objective or reward). If you return an array from `writeJS()`, check that it is an array with `is_array()`, and that it has a valid number of elements. If the output of `writeJS()` is a string, check that `$data` is a string, etc.

17.4.2 Registering the parameter class

In order for a parameter class to be usable, it must be registered in the `PARAMETERS` array in the `Research` class in `research.php`. The array may look like this:

```
const PARAMETERS = array(  
    // Class mappings for each parameter  
    "quantity" => "ParamQuantity",  
    "min_tier" => "ParamMinTier",  
    "species" => "ParamSpecies",  
    "type" => "ParamType",  
    "encounter_item" => "ParamEncounterItem"  
);
```

Assign your parameter an ID that reflects the kind of data that it holds. For example, the quantity of items rewarded by a task, or quantity of Pokémon needed to complete a task, etc. has been assigned the ID `quantity`. The parameter that stores the type of Pokémon that must be caught/evolved/etc. for a task is called `type`. The name you choose must be in snake_case and is the **key** of your parameter in the `PARAMETERS` array.

The **value** of this key is the name of the class you implemented above. The class name should, for consistency, begin with “Param,” and be followed by the ID of the parameter converted to UpperCamelCase. For example, `min_tier` becomes `ParamMinTier`.

17.4.3 Adding internationalization tokens

You need to define two internationalization tokens in the *localization files* for your parameter. These are located above the Pokémon species names list, and underneath the objectives, rewards and categories in the files. You must, as a minimum, add the tokens to the `en_US.ini` file, though if you speak another language, feel free to add them to the localization files of other languages you speak.

The tokens are:

```
parameter.<parameter_id>.label = "Label of parameter"  
parameter.<parameter_id>.placeholder = "[Placeholder]"
```


The `label` sub-key should contain a short human-readable label that describes the parameter. E.g. for `quantity`, this is simply “Quantity.” This is the string that appears next to the input boxes when users report research with the parameter.

The `placeholder` sub-key is displayed in the list of research objectives and/or rewards for objectives/rewards which implement the parameter, but for which no value is currently known. E.g. for `quantity`, the placeholder is “[n].” This means that the parameter could be displayed as e.g. “Catch [n] Pokémon” if no quantity is currently known for that task.

Note: Make sure you declare these internationalization tokens in the same place as the tokens declared for the other parameters. You can find them quickly by searching the file for “parameter.”

17.4.4 List of parameters

The following is a list of all available parameters. When you add a new parameter type, please document it here.

`quantity`

Quantity of items rewarded by a task, quantity of Pokémon to catch to complete an objective, etc. Can be used for both objectives and rewards.

The data of this parameter are stored as **integers** that are 1 or greater.

`min_tier`

Minimum tier of a raid that must be completed (used in “Win a level [x] or higher raid”). Can only be used for objectives.

The data of this parameter are stored as **integers** that are in the range 1 through 5.

`species`

One or more species of Pokémon that must be caught, evolved, etc. up to three different species. Can only be used for objectives.

The data of this parameter are stored as **arrays of integers** that are at least one, and up to three elements long. Each element is in the range 1 through the Pokédex number of the last Pokémon in the latest generation of Pokémon that are currently implemented in FreeField. The numbers correspond to the Pokédex number of Pokémon.

`type`

One or more types of Pokémon that must be caught, evolved, etc. up to three different types. Can only be used for objectives.

The data of this parameter are stored as **arrays of strings** that are at least one, and up to three elements long. The elements are picked from `TYPES`:

```
const TYPES = array(
    "normal",    "fighting", "flying",
    "poison",    "ground",   "rock",
    "bug",       "ghost",    "steel",
```

(continues on next page)

(continued from previous page)

```
"fire",      "water",    "grass",  
"electric",  "psychic",  "ice",  
"dragon",    "dark",     "fairy"  
);
```

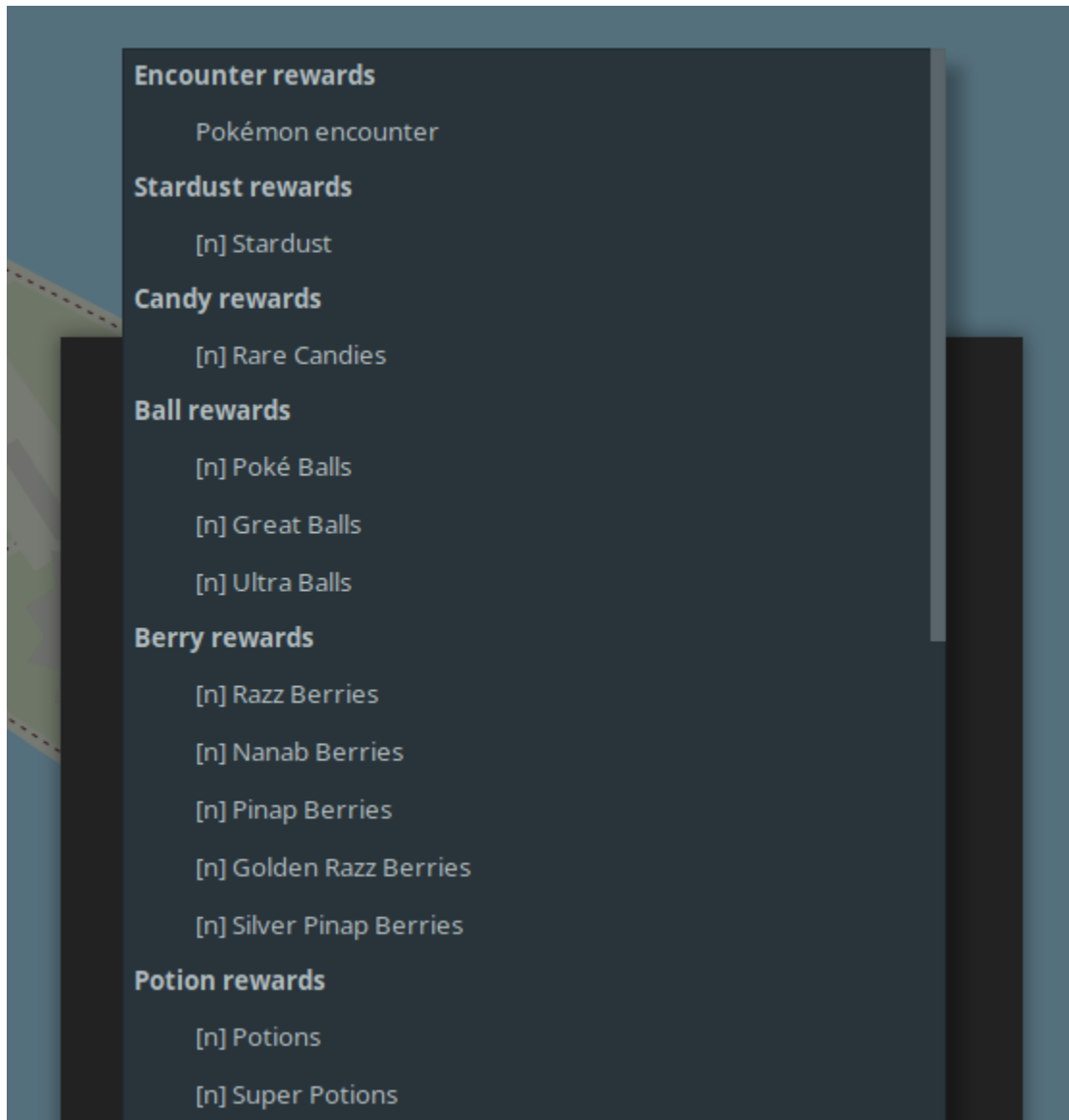
`encounter_item`

An item that can be used on the Pokémon encounter screen (i.e. balls and berries). This is used for the “Use a [Item] while catching Pokémon” objective. Can only be used for objectives.

The data of this parameter are stored as **strings**, selected from the set of all rewards that are assigned to the categories “ball” or “berry” in the `rewards.yaml` file.

17.5 Icons and categories

Objectives and rewards are both assigned to one or more categories together with other similar objectives or rewards. Elements that are categorized appear together in the list of reportable objectives and rewards:



Categories are defined using the `categories` field in objective and reward definitions:

```
# Great Balls
great_ball:
  params:
    - quantity
  categories:
    - ball
```

If several categories are specified, only the category listed at the top will be used to group the objectives/rewards in the selection menus.

Categories are also used to determine the icon that is displayed on the map for research that is reported with a specific objective or reward. FreeField will look for an icon matching the top-most category in the icon set's `pack.ini` file, and continue down the list looking for the first one that is present in `pack.ini` if the top-most icon is not found. If no icons are found, FreeField falls back to the `default` icon. If there is an entry in `pack.ini` specifically matching the ID of the objective or reward, that icon will be used regardless of the categories in the definition. For more information on how icons work, please refer to the documentation on creating icon sets.

This is the definition for the `win_raid` objective:

```
# Win a raid
win_raid:
  params:
    - quantity
  categories:
    - raid
    - battle
```

When FreeField attempts to load an icon for this objective, it will try the following icons in order until a matching entry in the `pack.ini` of the selected icon set is found:

- `win_raid`
- `raid`
- `battle`
- `default`

17.5.1 Adding a new category

You can declare a new category simply by using it in the definition of an objective or reward. When declaring a new category, ensure that you follow the `snake_case` naming convention of IDs in the data files.

You must declare an internationalization token in the [localization files](#) and at a minimum, localize it to English if you add a new category. Category tokens are located above the parameter tokens, and underneath the objectives and rewards in the files. If you speak other languages than English, feel free to localize the tokens to the other languages as well. The token is one of the following:

```
; For objectives:
category.objective.<category_id> = "Amazing objectives"

; For rewards:
category.reward.<category_id> = "Amazing rewards"
```

The token contains the header that is displayed for the category in the objective/reward selection boxes. You must declare this token even if you do not intend to use the category as a top-level category. Use either the “objective” or “reward” token, depending on whether you are applying the category to objectives or rewards.

Note: Make sure you declare these internationalization tokens in the same place as the tokens declared for the other categories. You can find them quickly by searching the file for “category.”

This document serves as the main authoritative style guide for contributions to FreeField, and goes into great detail on the coding standards that all contributors to the project should adhere to.

Hint: This guide is comprehensive, and may be overwhelming. If you fail to comprehend, or simply forget some conventions, don't be afraid to make submissions to the repository. If anything, take a look at the section on *Non-compliance*, which describes what will happen if your code does not comply with the code style outlined in this document. In most cases, nothing bad will happen at all - in the worst case, you will be asked in a friendly manner to fix some specific things before your contribution is accepted. This guide simply exists to help you avoid those pitfalls.

18.1 Non-compliance

If your submission does not meet the style requirements outlined in this document, several things can happen.

The two most important parts of this guide is **follow indentation requirements**, and **do not have copyrighted assets in the commit history of your contribution** if you submit pull requests. If either of those are the case, your contribution will be rejected, and you will have to fix the submission so that it is compliant.

If your contribution is small, it might be accepted even if it is non-compliant, in which case the errors will be corrected by FreeField maintainers. In other cases, you may be asked to correct it yourself, in a friendly manner.

If your contribution is significant, or has a large number of, or significant, code style violations, you will likely be asked to correct those yourself in a friendly manner, as the maintainers do not always have the resources to make such corrections.

When making contributions, you will always be treated respectfully and in line with FreeField's [Code of Conduct](#).

18.2 Indentation

For **all** code files (except reStructuredText), indentation must be done with four spaces. reStructuredText files are indented with three spaces. Code snippets within reStructuredText documentation should be indented with four spaces (see e.g. the source code of this file, [codestyle.rst](#), on GitHub).

Under no circumstances should tabs be used. Contributions which do not satisfy this requirement **will always** be rejected.

You may indent a line with more than four spaces if doing so is reasonable to improve the readability or visual structure of your code. E.g. this is okay:

```
$string = "This is a really long string that will wrap across two lines in "  
        . "the source code.";
```

18.3 Spacing

- Segments of code that do unrelated things should be separated with blank lines.
- Use one space around operators:

```
// Do this:  
$var = 1;  
$var = 2 + 5;  
$var = ($bool ? 10 : 20);  
for ($i = 0; $i < 10; $i++)  
if ($x == $y)  
  
// Don't do this:  
$var=1;  
$var = 2+5;  
$var = ($bool?10:20);  
for ($i=0; $i<10; $i++)  
if ($x==$y);
```

- Parentheses in block statements should be wrapped with one space on each side:

```
// Use one space on either side of (true) - i.e. don't do this:  
while      (true)      {  
    execute();  
}  
// Also try not to do this:  
while(true){  
    execute();  
}
```

18.4 Code blocks

This is the accepted format for code blocks (if, for, switch, etc.):

```
while (true) {  
    execute();  
}
```

In detail, this means:

- Opening curly braces should be on the same line as the statement that opens it:

```
// Don't do this:
while (true)
{
    execute();
}
```

- The ending curly brace should be on its own line, at the same indentation level as the line that starts the block.

```
// Don't do this:
while (true) {
    execute(); }

// Don't do this either:
while (true) {
    execute();
}
```

- Short form blocks are allowed if the *entire* statement is short enough to fit on one line. Never mix the short-form with the curly brace form.

```
// This is okay:
while (true) execute();

// This is not okay:
if ($bool) executel();
else {
    execute2();
}
```

18.4.1 switch statements

- Use switch instead of if where reasonably possible:

```
// Do this:
switch ($var) {
    case 1:
        executel();
        break;
    case 2:
        execute2();
        break;
    default:
        execute();
        break;
}

// .. instead of this:
if ($var == 1) {
    executel();
} elseif ($var == 2) {
    execute2();
} else {
```

(continues on next page)

(continued from previous page)

```
execute();  
}
```

- case statements are indented:

```
switch ($var) {  
    case 1:  
        execute();  
        break;  
    case 2:  
        execute();  
        break;  
}
```

18.5 Wrapping

Code, including comments, should not exceed 80 characters per line. If splitting the code or comments over multiple lines is unreasonable (for example, if the code is already indented >50 characters), an exception can be made. In those cases, limit the lines to 100, 120, 140 etc. characters, depending on what you consider reasonable.

If you want to split long lines, there are several ways you do that. Here are some examples that are used consistently in existing FreeField source code:

18.5.1 Splitting HTML tags with many attributes

```
<input type="text"  
    id="sampleInput"  
    name="sampleInput"  
    class="some-long-class-name another-long-class-name"  
    value="This is the value of the input box">
```

18.5.2 Splitting arrays

```
$array = array(  
    "element1" => 1,  
    "element2" => 2,  
    "element3" => 3  
);
```

18.5.3 Splitting block statements

```
if (  
    $int >= 3 &&  
    substr($str1, 0, 10) !== substr($str2, 0, 10)  
) {  
    executeA();  
    executeB();  
}
```


18.5.4 Splitting ternary operators

```
$var = (
    isSomeStatementTrue()
    ? "valueIfTrue"
    : "valueIfFalse"
);
```

18.6 Naming

- Classes use `CamelCase`, functions and variables use `lowerCamelCase`.
- All variables, functions, objects, classes, etc. must be written in **English only**.
- Always substitute “Pokémon” with “species,” “Pokéstop” with “POI,” and “gym” with “arena” in all contributions. Do not reference individual Pokémon species by name. This does not apply to strings in the localization files, the documentation, and screenshots. It also does not apply to comments in the `common-tasks.yaml`, `objectives.yaml` and `rewards.yaml` files. It *does* apply to all comments in all other files.

18.7 Miscellaneous

- Never use the short form opening tag `<?`. Always use `<?php`.
- All strings that are displayed to the user must be internationalized using the `I18N` class. Look for examples in existing code if you’re not sure.
- Your code must run with no errors, notices, etc. using `E_ALL`.
- PHP code must also run under PHP 5.6. If this is not possible, raise an issue explaining why this is the case before you start making your contribution.
- Do not use Composer dependencies. Feel free to raise an issue asking for guidance on alternatives if you want to add a contribution that requires a Composer dependency. Do this before you start writing your contribution.
- Use a CDN for CSS and JavaScript libraries (e.g. [CDNJS](#)) wherever possible.
- Files must be in UTF-8.
- Outputted HTML must be W3C-compliant to the HTML5 standard (try e.g. [this validator](#)).
- **Comply with the license of FreeField.** This means that assets protected by copyright, including Pokémon imagery, must never be part of your contribution, **including in its commit history** if you are submitting a pull request. If your pull request contains, or has at any point contained, copyrighted assets for which usage in FreeField has not been granted, your pull request will be closed and will not be opened again, even if you remove the offending assets from your contribution. In such cases, re-fork the repository and re-add your commits without ever including the offending assets.
- **Test your code** before submitting it. Broken code will be rejected.

18.8 Commenting

All of the code in your contribution should be commented in such a way that people other than yourself, who has never seen your code before, can understand what it does.

In practice, this means that you:

- Must describe the **purpose of each code file** in a comment at the top of the file. This comment must be in multi-line format `/* ... */`, even if the comment only spans one line - this is for consistency.
- Must describe the **purpose of each function and class** on lines immediately preceding that function/class, in multi-line format.
- Should describe the **arguments and return values of functions** if they are not immediately clear/obvious.
- Must describe the **purpose of variables** in classes and globally if they are not immediately clear/obvious. This must be done on separate lines preceding the variable(s).
- Must explain **non-trivial/hard to understand code and algorithms** in detail - what it does, why it is there, an explanation of the calculations it performs, etc. Visualizations using ASCII-style graphics are welcome if you feel like making them, and if they contribute to a better understanding of the algorithm, but they are by no means required.
- Should *not* comment what is obvious. E.g. if you have a `while ($bool)` in your code, you should not say “loops while \$bool is true.” You are welcome to comment on the *purpose or effects* of the code instead, if you believe it can promote better understanding of your code.

18.8.1 Style requirements for comments

- Comments must always be made on lines preceding the statement it comments, and never shifted to the right on the same line:

```
// This is good comment placement
$var = true;

$var = true;      // This is bad comment placement
```

- Variable names, functions, classes, etc. in comments must be wrapped in backticks ``:

```
// The variable `$foo` is passed to the `bar()` function.
```

- Multi-line comments must have the starting `/*` and ending `*/` tags on lines by themselves.
- Languages that do not support single-line comments `//` natively can substitute that functionality with multi-line comments. In those cases, the start and end tags are placed inline with the comment body, each of them separated by one space from the text, and the text of any subsequent lines indented to match the starting position of the first line (see [CSS](#) and [HTML](#) below for examples).
- The body of multi-line comments must be indented once (i.e. by 4 spaces).
- Lines in multi-line comments must not start with asterisks.

18.8.2 Examples

PHP and JavaScript

```
/*
    This is a good multi-line comment. The opening and closing tags of the
    comment are on separate lines, and lines in the comment body are indented
    and do not start with asterisks.
*/

/*
```

(continues on next page)

(continued from previous page)

```
This is not a good multi-line comment - the body text is not indented by four spaces.
*/

/*
 * This is not a good multi-line comment - lines should not start with
 * asterisks.
 */
```

CSS

```
/* This is an inline comment for CSS. CSS does not support single-line
   comments, so an inline style of multi-line comments is used as a
   substitution. */

/* This is not a good CSS comment. Subsequent lines for this comment are not
   indented properly. */

/*
   For multi-line comment blocks, the same comment style is used as for PHP
   and JavaScript.
*/
```

HTML

```
<!--
   Multi-line comment blocks follow the same style as CSS, PHP and
   JavaScript.
-->

<!-- Single-line comment blocks follow the same style as CSS. -->
```


CHAPTER 19

Configuration

All configuration in FreeField is stored in `includes/userdata/config.json` and managed using the `Config` class. To import the `Config` class for usage in a script, the following should be placed at the top of the script file:

```
require_once("../includes/lib/global.php");
__require("config");
```

Settings are defined in the `defs.php` file. An example entry may look like this:

```
/*
    The default zoom level of the map.
*/
"map/default/zoom" => array(
    "domain" => "map",
    "section" => "default",
    "default" => 14.0,
    "option" => new FloatOption(0.0, 20.0)
),
```

The key of the array element is the path to the setting in the configuration file, and should be placed such that is grouped with other similar/related settings.

To retrieve a value from a setting, use `Config::get("path/to/setting")->value()`. Storing a setting is always handled using the administration pages, and should **never be saved directly from code**. For completeness of the documentation, however, it should be stated that the function to manually save configuration is `Config::set()`, taking, at minimum, an argument that is an associative array of the form `"path/to/setting" => value`. Please see commentary for the `Config::set()` function before you ever use this to learn more about how it works and the meaning of arguments that are passed to it.

Danger: Never use `->value()` as part of output! See [Security in FreeField](#) for information on `->valueHTML()` and other functions that should be used instead for output in order to prevent security/XSS vulnerabilities.

19.1 Internationalization

All setting strings are internationalized. Setting names are assigned the key `setting.<path>.name`, and descriptions helping the user understand the purpose of each setting are `setting.<path>.desc`, where `<path>` is the path from the key of the setting array with slashes `/` converted to dots `.` and dashes `-` converted to underscores `_`. For example, `map/default/zoom` are assigned the internationalization tokens `setting.map.default.zoom.name` for its label and `setting.map.default.zoom.desc` for its description. Please see [Internationalization](#) for more information.

19.2 Domains and sections

Every setting in FreeField has an associated **domain** and **section** under which it should be displayed. The **domain** is the sub-page of the administration pages where the setting should appear. That page is further denominated by various headers called **sections**, which are groupings of similar or related settings. For example, the setting “Zoom level,” defining the default zoom level when viewing the map, is organized under the “Defaults” section of the “Map settings” domain.

19.2.1 Adding new domains

Domains always have a label (page title) and a short description (subtitle). These must be added to the localization files under the tokens `admin.domain.<domain>.name` and `admin.domain.<domain>.desc` respectively.

There are two types of domains; **standard** and **custom** domains. Standard domains have all their settings stored in the configuration definitions file `defs.php`, and will render these on the page. Custom domains are not from `defs.php`, but are instead rendered from custom output files `includes/admin/<domain>.php`. For example, the `users` domain does not refer to or host any settings in `defs.php`, but instead renders a list of all users of the FreeField instance in the database. This logic is handled in `users.php`, which is `include'd` to the administration page upon page load. If you are adding a new domain for usage within `defs.php`, use the standard domain type rather than custom domain.

The domain you are adding must be listed in the `Config::listDomains()` function. An example of such an entry:

```
// Map provider settings (e.g. map API keys)
"map" => array(
    "icon" => "map",
    "custom-handler" => false
),
```

`custom-handler` should be set to `true` if you are adding a custom page, or `false` if you are adding a standard domain. `map` corresponds to a FontAwesome icon that should represent the domain the administration pages sidebar.

When adding a new domain, you must also add a domain access permission setting to `defs.php` to restrict access to the domain by default. This permission looks like this:

```
"permissions/level/admin/{<domain>/general" => array(
    "domain" => "perms",
    "section" => "admin",
    "default" => PermissionOption::LEVEL_ADMIN,
    "option" => new PermissionOption()
),
```

Make sure to select a default permission level that is appropriate to the types of settings you are adding.

Tip: This commit contains an example of the “Mobile” domain being added to FreeField, along with its associated functionality and settings. You can use this as a rough template on how to implement a new domain.

19.2.2 Adding new sections

Like domains, sections always have a label (section header), though a description is optional. Labels use the internationalization token `admin.section.<domain>.<section>.name` where `<domain>` is the parent domain of the section.

Note: Sections are not added to custom-type domains - they are declared directly within the `includes/admin/<domain>.php` output file, but they should still follow the general internationalization conventions as other settings.

A section will have a description if, and only if, the section has an entry in the `SECTIONS_WITH_DESCRIPTIONS` array in `config.php`. Please see the commentary for that array to learn how to add descriptions to sections.

When adding a new section, you must also add a section access permission setting to `defs.php` to restrict access to the section by default. This permission looks like this:

```
"permissions/level/admin/{<domain>}/section/{<section>}" => array(
    "domain" => "perms",
    "section" => "admin",
    "indentation" => 1,
    "default" => PermissionOption::LEVEL_ADMIN,
    "option" => new PermissionOption()
),
```

Make sure to select a default permission level that is appropriate to the types of settings that are manageable under this section.

19.3 Options and data types

Every setting is of a certain **option** type. Available options are declared in `types.php`. The option type declares the type of data that is stored for the setting, and provides parsing, storage and validation functions specific to that option type. Instructions on implementing new options are available as commentary at the top of that file.

19.3.1 Available option types

This is a list of all available option types in FreeField. Please add any new options you add to this list.

StringOption

For storing short strings. Can be initialized with an optional regular expressions pattern via the constructor, which, if specified, will reject all strings that do not match this pattern as invalid.

Valid initializers

```
// Accept any string:
"option" => new StringOption()

// Using regex to e.g. only accept strings without spaces:
"option" => new StringOption('^[^s]+$')
```

Valid defaults

Any string, matching the regex if provided.

ParagraphOption

For storing longer strings. Can optionally be initialized with the string "md" to display a live Markdown preview.

Valid initializers

```
// Plain-text paragraph input:
"option" => new ParagraphOption()

// Paragraph input with Markdown preview:
"option" => new ParagraphOption("md")
```

Valid defaults

Any string.

PasswordOption

For storing passwords and other sensitive data. Stored in encrypted form in the configuration file to prevent data leakage from misconfigured HTTP servers.

Potentially unexpected behavior

This option cannot store the string `oqXb_&WkMrdHtRZ_@}qBM=?WheuO6Y`. This string is subject to change in the future. The reason is that this string is returned in lieu of the actual string in the configuration page when echoed to the page to give the user a visual impression that it is set to an existing value, as it will fill the input box with black dots. If this string is returned from the browser, it indicates that the input was not changed by the user, and is thus discarded.

Valid initializers

```
"option" => new PasswordOption()
```


Valid defaults

An empty string.

BooleanOption

For storing boolean values; displayed as a checkbox with a separate label next to it.

Attention: This option requires that an additional internationalization token is declared for the label, i.e. `setting.<path>.<label>`. This string is displayed next to the checkbox.

Valid initializers

```
"option" => new BooleanOption()
```

Valid defaults

true or false.

IntegerOption

For storing integers. Can be initialized with optional minimum and maximum values (both inclusive).

Valid initializers

```
// Accept any integer:
"option" => new IntegerOption()

// Accept an integer with a certain minimum value (e.g. 10):
"option" => new IntegerOption(10)

// Accept any integer up to a certain maximum value (e.g. 20):
"option" => new IntegerOption(null, 20)

// Accept any integer in a range from a minimum to a maximum value:
"option" => new IntegerOption(10, 20)
```

Valid defaults

Any integer, within the range if provided.

FloatOption

Similar to `IntegerOption`, but allows storing floating-point/decimal numbers. Can be initialized with optional minimum and maximum values (both inclusive).

Valid initializers

```
// Accept any number:
"option" => new FloatOption()

// Accept a number with a certain mininum value (e.g. 10):
"option" => new FloatOption(10.0)

// Accept any number up to a certain maximum value (e.g. 20):
"option" => new FloatOption(null, 20.0)

// Accept any number in a range from a minimum to a maximum value:
"option" => new FloatOption(10.0, 20.0)
```

Valid defaults

Any floating-point/decimal number, within the range if provided.

GeofenceOption

For selecting and storing references to a particular geofence, as defined by the user on the geofencing section of the administration pages; see [Geofencing](#).

Running `->value()` on settings of this option type will return a `Geofence` object instance, or `null` if set to “<none>” or invalid. See [geo.php](#) for implementation and usage details.

Valid initializers

```
"option" => new GeofenceOption()
```

Valid defaults

`null`.

SelectOption

For storing one value from a list of selectable valid values. Must be initialized with an array of items and an optional data type (“string” or “int”; default is “string”).

Attention: This option requires additional internationalization tokens for each of the options in the supplied items array, i.e. `setting.<path>.option.<option>`. Internationalization can be suppressed by passing `true` to the third parameter of the constructor of this option, though this is strongly recommended against unless there is a legitimate need to have unlocalized elements in the selection box.

Valid initializers

```
// Accept any item from given list of items:
"option" => new SelectOption(array("one", "two", "three"))

// Specify element data type:
"option" => new SelectOption(array(24, 48, 72), "int")
```

Valid defaults

Any element in the provided list of options, e.g. "one" for the first example above, or 72 for the second example.

PermissionOption

For selecting a user group; see *Permissions and groups* for more information. Renders as a selection box of all available groups in the FreeField installation.

Valid initializers

```
"option" => new PermissionOption()
```

Valid defaults

Any one of the following:

```
PermissionOption::LEVEL_HOST
PermissionOption::LEVEL_ADMIN
PermissionOption::LEVEL_MODERATOR
PermissionOption::LEVEL_SUBMITTER
PermissionOption::LEVEL_REGISTERED
PermissionOption::LEVEL_READ_ONLY
PermissionOption::LEVEL_ANONYMOUS
```

IconSetOption

For selecting an installed set of *Map markers*. Renders as a selection box of all available marker sets in the FreeField installation. An option for “default marker set” can be added to this selection box by passing an internationalization token as argument to the constructor to indicate a string that should be displayed to label the default option. If this is not passed, no default option is provided to the user.

A preview box is displayed for the selected icon set at all times. If a default option is selected, no preview is displayed, and an empty string will be returned from this option type.

Valid initializers

```
// Standard marker set selection box with no "Default" option:
"option" => new IconSetOption()

// Selection box with a default option denoted by an I18N display label:
"option" => new IconSetOption("setting.path_to_setting.option.default")
```

Valid defaults

A globally available icon set, i.e. only "freefield-3d-compass" is currently permitted.

FileOption

For uploading files to FreeField as part of the configuration. Used for e.g. the favicon. Uploaded files are stored in includes/userdata/files. The path of the setting this option is used for must be passed as the first argument. An optional array of file types and extensions can be passed, along with a maximum file size.

Running `->value()` on settings of this option type will return a `FileOptionValue` object instance. This class is declared in `types.php` and has the following methods:

```
// The following basic file-I/O functions exist:
getExtension()           // Returns e.g. ".jpg"
getFilename()            // Returns local filename, e.g. "path.to.setting.png"
getUploadName()          // Returns origin filename, e.g. "My awesome image.png"
getPath()                // Returns local file path, e.g. "/var/html/path.to.
↳setting.png"
getMimeType()            // Returns MIME type, e.g. "image/png"
getLength()              // Returns file size in bytes
getUploadTime()          // Returns UNIX timestamp of time and date file was last_
↳changed

// In addition, the following functions exist to provide file integrity:
getHexEncodedSHA256()    // Returns hexadecimal-encoded SHA256 hash of file
getBase64EncodedSHA256() // Returns base64-encoded SHA256 hash of file

// Finally, the following functions exist to read the file:
outputWithCaching()      // Sets caching headers, echoes file, then terminates
getDataURI()             // Returns file as a base64-encoded data URI
```

Valid initializers

```
// Accept any file
"path/to/setting" => array(
    /* ... other fields ... */
    "option" => new FileOption(
        "path/to/setting"
    )
),

// Accept only image files:
"path/to/setting" => array(
    /* ... other fields ... */
    "option" => new FileOption(
```

(continues on next page)

(continued from previous page)

```

        "path/to/setting",
        array(
            // This array is of format MIME type => default file extension.
            "image/png" => "png",
            "image/gif" => "gif",
            "image/jpeg" => "jpg"
        )
    ),
),
// Accept any file up to 256 KiB:
"path/to/setting" => array(
    /* ... other fields ... */
    "option" => new FileOption(
        "path/to/setting",
        null,
        256 * 1024
    )
),
// Accept only image files, and only up to 256 KiB:
"path/to/setting" => array(
    /* ... other fields ... */
    "option" => new FileOption(
        "path/to/setting",
        array(
            // This array is of format MIME type => default file extension.
            "image/png" => "png",
            "image/gif" => "gif",
            "image/jpeg" => "jpg"
        ), 256 * 1024 // Max 256 KiB
    )
),

```

Valid defaults

An array of the following format:

```

"default" => array(
    "type"    => "image/png",
    "name"    => "default-file-name.png",
    "size"    => 2044,
    "sha256" => "0a330b612466ea389359db56ce93f2a5faaa89359087926335c7bcab45b539e4"
)

```

The default file must be placed in [this directory](#). The filename must match the setting path with slashes / converted to dots .. The size and SHA-256 hash of the file must be included in the `default` array as indicated in the example above.

ColorOption

For selecting and storing an RGB color value. Displayed as a color picker, with indicators for the current values of the red, green and blue color channels for the selected color.

Valid initializers

```
"option" => new ColorOption()
```

Valid defaults

A hexadecimal color code string in the format #rrggbb.

CHAPTER 20

Internationalization

All user-visible strings in FreeField must be internationalized and localized. Internationalization is done using the `I18N` class server-side, or using the `clientside-i18n.php` client-side. To make `I18N` available for usage in a script use the following:

```
require_once("../includes/lib/global.php");
__require("i18n");
```

To include `clientside-i18n.php` for use in JavaScript, add the file to the HTML header of the page it is being used on, after jQuery, but before other scripts:

```
<head>
  <!-- Meta tags, etc. -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"
    integrity="sha256-FgpCb/KJQlLNfOu91ta32o/NMZxltwRo8QtmkMRdAu8="
    crossorigin="anonymous"></script>
  <script src="./js/clientside-i18n.php" async defer></script>
  <!-- Other scripts -->
</head>
```

For all static and server-side dynamically generated HTML, `I18N` should be used. `clientside-i18n.php` is only for use in standalone JavaScript files which are not parsed by the PHP parser.

20.1 String assignment

All strings are stored in the localization files assigned to a key (an “internationalization token”). Internationalization tokens act as identifiers for strings, and can consist of lowercase alphanumeric letters, periods and underscores. Here is an example of the structure of the localization file, showing the three tokens `ui.button.close`, `ui.button.cancel` and `ui.button.select` in the English localization files:

```
ui.button.close = "Close"
ui.button.cancel = "Cancel"
ui.button.select = "Select"
```

When requesting a localized string, you must use the internationalization token to refer to the string.

20.2 Usage

There are two types of strings that can be localized - simple strings, and parameterized strings. Simple strings are static, unchanging strings, and are used for most UI elements. Parameterized strings are strings that can change depending on data arguments passed to them, and can be identified by the appearance of substitution tokens within the string. A substitution token is a percentage sign followed by a number, both wrapped together in curly braces - for example, `{%1}`. These substitution tokens are substituted by other data, such as a number or another string at runtime. There can be any number of substitution tokens in a string, starting from `{%1}` with sequentially increasing numbers for each token.

Note: Substitution tokens are normally ordered in increasing order from their natural placement in the English-language variant of strings, but this may not always be the case. Some strings do not start with `{%1}` as the first token, and they may not necessarily be in increasing order. While it is strongly advisable that the tokens do appear in increasing order, switching them around may be preferable in some cases where it is sensible due to programming or style.

One example of this is the `quantity` parameter for research objectives. For singular variants of the strings, the substitution token for `quantity` is typically replaced by an article such as “a” or “an.” To ensure that tokens start numbering from 1, and to avoid “gaps” in the numbering sequence of tokens, `quantity` is given the highest substitution token number to ensure that the numbering makes sense if the token is omitted, even though it in most cases would appear as the first token of the string and therefore be assigned `{%1}`.

Important: When adding parameterized strings to the localization files, examples and explanations should be made for each substitution token in your string to assist translators in accurately translating the string.

Important: Any strings you want to use in `clientside-i18n.php` must be declared in that file on the server - they will not work unless their namespace have already been declared in the file from before.

Note: Strings for use in JavaScript on the administration pages must be declared with internationalization tokens starting with `admin.clientside.` and not under `admin.` directly.

20.2.1 Using simple strings

The basic I18N function to localize strings is `I18N::resolve($token)`. This function takes an internationalization token as input and returns a localized string for the currently active language.

```
$str = I18N::resolve("ui.button.close");  
echo $str; // Echoes "Close"
```

Danger: Never use `resolve()` as part of output! See [Security in FreeField](#) for information on `resolveHTML()` and other functions that should be used instead for output in order to prevent security/XSS vulnerabilities.

The JavaScript equivalent with `clientside-i18n.php` is `resolveI18N()`:

```
var str = resolveI18N("ui.button.close");
console.log(str); // Writes "Close" to console
```

Danger: Never use `.innerHTML` or jQuery `.html()` to assign string content to elements - use `.textContent` or jQuery `.text()` instead to prevent security/XSS vulnerabilities and character encoding issues. Please see [Security in FreeField](#) for more information.

20.2.2 Using parameterized strings

The `I18N` function to localize parameterized strings is `I18N::resolveArgs($token, ...$args)`. This function takes an internationalization token along with a list of arguments as input, and returns a localized string for the currently active language.

```
$str = I18N::resolveArgs("webhook.reported_by", "JohnDoe45");
echo $str; // Echoes "Reported by JohnDoe45"
```

Danger: Never use `resolveArgs()` as part of output! See [Security in FreeField](#) for information on `resolveArgsHTML()` and other functions that should be used instead for output in order to prevent security/XSS vulnerabilities.

The JavaScript equivalent with `clientside-i18n.php` is the same as for simple strings:

```
var str = resolveI18N("webhook.reported_by", "JohnDoe45");
console.log(str); // Writes "Reported by JohnDoe45" to console
```

Danger: Never use `.innerHTML` or jQuery `.html()` to assign string content to elements - use `.textContent` or jQuery `.text()` instead to prevent security/XSS vulnerabilities and character encoding issues. Please see [Security in FreeField](#) for more information.

20.3 Localization file structure

If you want to add new internationalization tokens to the localization files, you should stay in line with the structure of the file. Pay attention to headers and ensure that strings you add are grouped with similar strings. For example, generic UI elements/buttons should be localized under the “USER INTERFACE” header near the top of the file. Please do not randomly tack any additional strings to the start or end of the file.

Icon sets are sets of map marker assets that FreeField uses to display specific types of icons on the map, in webhooks and various places in the user interface. There are two types of assets - icons, which represent classes of objectives and rewards, and species icons, which represent various species of Pokémon. This document will explain the file structure of both types of assets, and where and how to install them in FreeField.

21.1 Location

Icon sets can be placed in two locations, relative to the root of the FreeField installation:

themes/ All icon sets that ship with FreeField are located here. This directory is deleted and re-created with the latest set of icons whenever FreeField is updated. Thus, changes made here are lost on every update.

includes/userdata/themes/ All user-defined icon sets are located here. This directory does not exist by default, but it can be created, and custom icon sets installed in it by local administrators, and they will show up fully usable in FreeField. Icon sets placed here will persist between updates.

If you are making an icon set for inclusion in the public upstream FreeField repository for others to use, it should be placed and staged in the root themes directory. If you are making an icon set for personal use in your local FreeField instance only, it should be placed in the userdata themes directory.

Within each themes directory, there are two subdirectories `icons` and `species`. The former contains icon sets for objective and reward classes, while the latter contains icon sets that represent Pokémon species.

21.2 File structure

Each icon set has its own root directory. This directory is located within the `icons` or `species` directories of either of the themes folder locations. The name of this directory will be the ID of the icon set. **This ID must be unique across both the root and the userdata themes directories.**

Within each icon set's root directory, there must be a file named "pack.ini" containing declarations of the file tree and naming structure of the icon set. The icons themselves are also located in this directory, either directly in the root, or in one or more subdirectories.

Icon sets support usage of different variants of icons. For example, when using dark theme, FreeField will attempt to use dark theme icons if the icon set supports it. These can be distinguished by either putting them in separate directories, or by appending "light" and "dark" to the file names of the assets themselves.

21.3 The pack.ini metadata file

"pack.ini" is a metadata file containing a description of the file structure of asset image files, as well as a description of the icon set itself. The format is slightly different for icon sets and species sets. Both files share a common header format, but have different body formats for the rest of the file.

21.3.1 Header

name The name of the icon set as displayed to users in FreeField.

author The author of the icon set.

link A URL pointing to the origin/source location of the icons used in the icon set.

supported_variants A comma-separated list of supported icon color variants (as of the latest FreeField version, `light` and `dark` are the only valid options).

logo (optional) A logo to display to users when this icon set is selected in FreeField. If omitted, no logo is displayed. You can use the `{%variant%}` placeholder to substitute "light" or "dark" into the file path.

Example header

This header will create an icon set named "FreeField 3D Compass" that supports light and dark theme. The logo will be loaded from `./light/logo.svg` if the end user is using light theme, and `./dark/logo.svg` if using dark theme, relative to the root of the icon set.

```
; FreeField 3D Compass icon set
; Created by bilde2910 (github.com/bilde2910)

name = "FreeField 3D Compass"
author = "bilde2910"
link = "https://github.com/bilde2910/FreeField"

supported_variants = "light,dark"
logo = "{%variant%}/logo.svg"
```

21.3.2 Body for icon sets

The body follows the header, and contains two sections `[vector]` and `[raster]` under which icon declarations are made. The sections are the same, except for the file paths you use. The `[vector]` section is optional, and if omitted, or if any of the icons in `[raster]` are missing from `[vector]`, those icons will fall back to using the raster versions.

The `[raster]` section is required. Files listed in the `[vector]` section must be SVG format, while `[raster]` allows PNG, GIF and JPEG. All file paths can use the `{%variant%}` placeholder to substitute “light” or “dark” into the file path depending on the color theme the user is using.

Under each section, key-value pairs are made that correspond an icon with a file path to that icon in the icon set. For example, to assign an file to the ball icon, you could put something like `ball = "ball.png"`. A list of all usable icons is available under [List of icon IDs](#) below.

Example body

This body provides a list of icons in both vector and raster format in light and dark variants.

```
[vector]

; Scalable vector versions

poke_ball = "{%variant%}/reward/poke-ball.svg"
great_ball = "{%variant%}/reward/great-ball.svg"
ultra_ball = "{%variant%}/reward/ultra-ball.svg"

razz_berry = "{%variant%}/reward/razz-berry.svg"
nanab_berry = "{%variant%}/reward/nanab-berry.svg"
pinap_berry = "{%variant%}/reward/pinap-berry.svg"
golden_razz_berry = "{%variant%}/reward/golden-razz-berry.svg"
silver_pinap_berry = "{%variant%}/reward/silver-pinap-berry.svg"

potion = "{%variant%}/reward/potion.svg"
super_potion = "{%variant%}/reward/super-potion.svg"
hyper_potion = "{%variant%}/reward/hyper-potion.svg"
max_potion = "{%variant%}/reward/max-potion.svg"
revive = "{%variant%}/reward/revive.svg"
max_revive = "{%variant%}/reward/max-revive.svg"

sun_stone = "{%variant%}/reward/sun-stone.svg"
kings_rock = "{%variant%}/reward/kings-rock.svg"
metal_coat = "{%variant%}/reward/metal-coat.svg"
dragon_scale = "{%variant%}/reward/dragon-scale.svg"
up_grade = "{%variant%}/reward/up-grade.svg"
sinnoh_stone = "{%variant%}/reward/sinnoh-stone.svg"

fast_tm = "{%variant%}/reward/fast-tm.svg"
charge_tm = "{%variant%}/reward/charge-tm.svg"
stardust = "{%variant%}/reward/stardust.svg"
rare_candy = "{%variant%}/reward/rare-candy.svg"
encounter = "{%variant%}/reward/encounter.svg"

battle = "{%variant%}/objective/battle.svg"
raid = "{%variant%}/objective/raid.svg"
catch = "{%variant%}/objective/catch.svg"
throwing_skill = "{%variant%}/objective/throwing-skill.svg"

hatch = "{%variant%}/objective/hatch.svg"
buddy = "{%variant%}/objective/buddy.svg"
explore = "{%variant%}/objective/explore.svg"

power_up = "{%variant%}/objective/power-up.svg"
evolve = "{%variant%}/objective/evolve.svg"
```

(continues on next page)

(continued from previous page)

```

trash = "{%variant%}/objective/trash.svg"
item = "{%variant%}/objective/item.svg"

social = "{%variant%}/objective/social.svg"

unknown = "{%variant%}/unknown.svg"
default = "{%variant%}/default.svg"

[raster]

; Bitmap versions

poke_ball = "{%variant%}/reward/poke-ball.png"
great_ball = "{%variant%}/reward/great-ball.png"
ultra_ball = "{%variant%}/reward/ultra-ball.png"

razz_berry = "{%variant%}/reward/razz-berry.png"
nanab_berry = "{%variant%}/reward/nanab-berry.png"
pinap_berry = "{%variant%}/reward/pinap-berry.png"
golden_razz_berry = "{%variant%}/reward/golden-razz-berry.png"
silver_pinap_berry = "{%variant%}/reward/silver-pinap-berry.png"

potion = "{%variant%}/reward/potion.png"
super_potion = "{%variant%}/reward/super-potion.png"
hyper_potion = "{%variant%}/reward/hyper-potion.png"
max_potion = "{%variant%}/reward/max-potion.png"
revive = "{%variant%}/reward/revive.png"
max_revive = "{%variant%}/reward/max-revive.png"

sun_stone = "{%variant%}/reward/sun-stone.png"
kings_rock = "{%variant%}/reward/kings-rock.png"
metal_coat = "{%variant%}/reward/metal-coat.png"
dragon_scale = "{%variant%}/reward/dragon-scale.png"
up_grade = "{%variant%}/reward/up-grade.png"
sinnoh_stone = "{%variant%}/reward/sinnoh-stone.png"

fast_tm = "{%variant%}/reward/fast-tm.png"
charge_tm = "{%variant%}/reward/charge-tm.png"
stardust = "{%variant%}/reward/stardust.png"
rare_candy = "{%variant%}/reward/rare-candy.png"
encounter = "{%variant%}/reward/encounter.png"

battle = "{%variant%}/objective/battle.png"
raid = "{%variant%}/objective/raid.png"
catch = "{%variant%}/objective/catch.png"
throwing_skill = "{%variant%}/objective/throwing-skill.png"

hatch = "{%variant%}/objective/hatch.png"
buddy = "{%variant%}/objective/buddy.png"
explore = "{%variant%}/objective/explore.png"

power_up = "{%variant%}/objective/power-up.png"
evolve = "{%variant%}/objective/evolve.png"
trash = "{%variant%}/objective/trash.png"
item = "{%variant%}/objective/item.png"

social = "{%variant%}/objective/social.png"

```

(continues on next page)

(continued from previous page)

```
unknown = "{%variant%}/unknown.png"
default = "{%variant%}/default.png"
```

21.3.3 Body for species sets

The body follows the header, and contains one or more range declarations that specify the location of icons for all or a subset of Pokémon species. The body may consist of any number of `[range|n]` sections, where *n* is a counter, and a `[default]` fallback section for icons which are not covered by a range.

`[range|n]`

Range blocks define icons for a range of Pokémon species. They have four settings, three of which are required:

range_start The pokedex ID of the first Pokémon to be covered by the range.

range_end The pokedex ID of the last Pokémon to be covered by the range.

vector (*optional*) A file path template to refer to vector graphics for icons in this range. `{%n%}` can be used to extract the pokedex ID, and `{%variant%}` to extract “light” or “dark” depending on the user theme. If omitted, FreeField will fall back to raster graphics for this icon range.

raster A file path template to refer to raster graphics for icons in this range. `{%n%}` can be used to extract the pokedex ID, and `{%variant%}` to extract “light” or “dark” depending on the user theme.

`[default]`

The default block is used as a catch-all or fallback for icons which have not already been matched by a range block. This block has two settings, of which one is required:

vector (*optional*) A file path template to refer to vector graphics for icons in this range. `{%n%}` can be used to extract the pokedex ID, and `{%variant%}` to extract “light” or “dark” depending on the user theme. If omitted, FreeField will fall back to raster graphics for this icon range.

raster A file path template to refer to raster graphics for icons in this range. `{%n%}` can be used to extract the pokedex ID, and `{%variant%}` to extract “light” or “dark” depending on the user theme.

Example

This body declares that icons should be separated into subfolders for each generation, up to Generation II. All other icons from later generations will show a fallback “unknown” icon instead.

```
[range|1]

range_start = 1
range_end = 151
vector = "{%variant%}/vector/gen1/{%n%}.svg"
raster = "{%variant%}/raster/gen1/{%n%}.png"

[range|2]

range_start = 152
range_end = 251
```

(continues on next page)

(continued from previous page)

```
vector = "{%variant%}/vector/gen2/{%n%}.svg"
raster = "{%variant%}/raster/gen2/{%n%}.png"

[default]

vector = "{%variant%}/vector/unknown.svg"
raster = "{%variant%}/raster/unknown.png"
```

21.4 List of icon IDs

Icons in FreeField are pulled from the `objectives.yaml` and `rewards.yaml` files located in the `includes/data` directory. Objectives and rewards are in turn organized into categories. You can define icons for any objective, reward or category of these.

The following tree represents a list of all icons currently available in FreeField. Declaring an icon for a node in this tree will also result in that icon being applied to all children of that node, unless specifically overwritten by a child node. The only icon that is required is `default`, but for the icon set to actually have some use, some individual icons and/or category icons must be declared as well.

```
default
├── unknown
├── battle
│   ├── battle_gym
│   ├── win_gym
│   ├── raid
│   │   ├── battle_raid
│   │   ├── win_raid
│   │   └── level_raid
│   └── se_charge
├── catch
│   ├── catch
│   ├── catch_weather
│   ├── catch_type
│   ├── catch_specific
│   └── catch_daily
├── item
│   ├── use_berry
│   └── use_item_encounter
├── buddy
│   └── buddy_candy
├── hatch
│   └── hatch
├── evolve
│   ├── evolve
│   ├── evolve_type
│   ├── evolve_evolution
│   ├── evolve_specific
│   └── evolve_item
├── trash
│   └── transfer
│       └── transfer
├── throwing_skill
│   ├── throw_simple_nice
│   └── throw_simple_nice_chain
```

(continues on next page)

(continued from previous page)

```

├─ throw_simple_great
├─ throw_simple_great_chain
├─ throw_simple_excellent
├─ throw_simple_excellent_chain
├─ throw_curve
├─ throw_curve_chain
├─ throw_curve_nice
├─ throw_curve_nice_chain
├─ throw_curve_great
├─ throw_curve_great_chain
├─ throw_curve_excellent
├─ throw_curve_excellent_chain
├─ explore
├─   └─ visit_poi
├─   └─ new_poi
├─   └─ visit_daily
├─ social
├─   └─ send_gift
├─   └─ trade
├─ encounter
├─   └─ encounter
├─ stardust
├─   └─ stardust
├─ candy
├─   └─ rare_candy
├─ ball
├─   └─ poke_ball
├─   └─ great_ball
├─   └─ ultra_ball
├─ berry
├─   └─ razz_berry
├─   └─ nanab_berry
├─   └─ pinap_berry
├─   └─ golden_razz_berry
├─   └─ silver_pinap_berry
├─ potion
├─   └─ potion
├─   └─ super_potion
├─   └─ hyper_potion
├─   └─ max_potion
├─ revive
├─   └─ revive
├─   └─ max_revive
├─ tm
├─   └─ fast_tm
├─   └─ charge_tm
├─ evolution_item
├─   └─ sun_stone
├─   └─ kings_rock
├─   └─ metal_coat
├─   └─ dragon_scale
├─   └─ up_grade
├─   └─ sinnoh_stone

```

21.4.1 Example

In this example, we've declared the following in our pack.ini file:

```
[raster]

default = "default.png"
battle = "battle.png"
raid = "raid.png"
level_raid = "level_raid.png"
catch_type = "catch_type.png"
```

The icons that will be used in this case are:

Icon	Inherits from	Image path	Usage defined
default		default.png	Explicitly
unknown	default	default.png	By inheritance
battle	default	battle.png	Explicitly
battle_gym	battle , default	battle.png	By inheritance
win_gym	battle , default	battle.png	By inheritance
raid	battle, default	raid.png	Explicitly
battle_raid	raid , battle, default	raid.png	By inheritance
win_raid	raid , battle, default	raid.png	By inheritance
level_raid	raid, battle, default	level_raid.png	Explicitly
se_charge	battle , default	battle.png	By inheritance
catch	default	default.png	By inheritance
catch_weather	catch, default	default.png	By inheritance
catch_type	catch, default	catch_type.png	Explicitly
catch_specific	catch, default	default.png	By inheritance
catch_daily	catch, default	default.png	By inheritance

The other icons in the tree all fall back to `default` and are not listed above for brevity.

This document aims to explain various considerations that have been made with regards to security in FreeField, and steps that need to be taken to ensure this security standard is upheld.

22.1 Cross-site scripting (XSS)

When outputting user-provided data, the data must be escaped to prevent XSS attacks. XSS attacks let users input data in such a way that when the data is later output, their result forms some kind of HTML tag or JavaScript statement that allows arbitrary code execution. For example, a user could use `<script>alert("Hello world!");</script>` as their username and, if not properly handled, this would result in the execution of the `alert()` statement when a browser visits a page with unescaped output.

Sanitation of data is required for all data that is:

- Obtained directly from end users (e.g. usernames, Pokéstop names, etc.)
- Obtained via third party APIs (e.g. user provider identities, Telegram group names, etc.)
- Stored in configuration files or databases
- Strings from localization files

If the data is converted to a format that does not allow script execution, such as a string converted to an integer via `intval()`, the converted output does not require XSS sanitation, but bounds checking may still be required to ensure values are within an acceptable range.

22.1.1 General handling

If no specific function exists for automatically escaping values as listed below, you should use the following functions to escape data:

```
// Output to HTML (returns string with escaped characters):
echo htmlspecialchars($unsafeStr, ENT_QUOTES);

// Output to JavaScript (returns JSON encoded data):
echo json_encode($unsafeStr);

// Output as part of URL (e.g. for Location HTTP header):
echo "./foobar.php?value=".urlencode($unsafeStr);
```

22.1.2 Configuration values

When displaying values from the FreeField configuration directly on the page, use the following functions:

```
// Output to HTML:
echo Config::get("setting/path")->valueHTML();

// Output to JavaScript:
echo Config::get("setting/path")->valueJS();

// Output as part of URL:
echo Config::get("setting/path")->valueURL();

// NEVER do this:
echo Config::get("setting/path")->value();
```

22.1.3 Localized strings

When displaying localized strings on HTML pages, use the following functions:

```
// Without arguments to HTML:
echo I18N::resolveHTML("token.path");

// Without arguments to JSON/escaped and quoted string:
echo I18N::resolveJS("token.path");

// With arguments to HTML:
// - If `$args` contains unescaped user data, then use:
echo I18N::resolveArgsHTML("token.path", true, $args);
// - Otherwise, if you are 100% certain all data in `$args` cannot contain
//   user provided data, or the data in `$args` is already escaped, use:
echo I18N::resolveArgsHTML("token.path", false, $args);

// With arguments to JSON/escaped and quoted string:
// - If `$args` contains unescaped user data, then use:
echo I18N::resolveArgsJS("token.path", true, $args);
// - Otherwise, if you are 100% certain all data in `$args` cannot contain
//   user provided data, or the data in `$args` is already escaped, use:
echo I18N::resolveArgsJS("token.path", false, $args);
```

The second argument to the `resolveArgs` functions determines whether the complete and fully localized string should be escaped (`true`), or whether only the base string should be escaped, and the `$args` data should be substituted in without escaping (`false`). If `$args` intentionally contains HTML tags, but also contains user data, escape the user data in the array using `htmlspecialchars($string, ENT_QUOTES)`.

Client-side, all strings are resolved using `resolve()` and `resolveArgs()`. When outputting them to the page:

```
// Do this:
$("#element").text(resolve("token.path"));

// NEVER do this:
$("#element").html(resolve("token.path"));
```

22.2 Cross-site request forgery (CSRF)

A CSRF attack involves a user voluntarily or involuntarily making a request to FreeField from a site hosted elsewhere, such as by submitting a form on a third party site that points to a script on FreeField. This can cause users to perform unwanted actions, such as a form tricking them to e.g. send an email that instead is submitted to FreeField with hidden fields that cause a malicious user's privileges to be elevated.

All forms must use CSRF protection. This also applies to anchor tags that perform some kind of server-side action, such as anchors to auth/logout.php. This can be implemented as such:

1. Add this to the top of the PHP script that contains the input form or anchor, before any other output is written to the browser:

```
__require("security");
Security::requireCSRFToken();
```

2. Do either of the following:

- For HTML forms, output a CSRF field:

```
<form method="post" action="foo.php">
  <?php echo Security::getCSRFInputField(); ?>
  <!-- More form fields here -->
</form>
```

- For anchors, add the CSRF parameter to the URL:

```
<a href="./foo.php<?php echo Security::getCSRFUrlParameter(); ?>">
  <!-- Anchor content -->
</a>
```

3. In the target script that processes the form contents or anchor, do this before any processing takes place to check if there is a CSRF failure, and cancel processing if that is the case:

```
__require("security");
if (!Security::validateCSRF()) {
    // Validation failed, redirect user to where they came from
    header("HTTP/1.1 303 See Other");
    header("Location: /return/path.php");
    exit;
}
```

22.3 Securing authentication

Always make use of any CSRF protection mechanisms provided by the authentication provider's API, typically via OAuth2 with the `state` parameter:

```
// auth/oa2/*.php

__require("vendor/oauth2");
$opts = array(
    /* ... other OAuth2 options go here ... */
    "params" => array(
        "state" => $state = bin2hex(openssl_random_pseudo_bytes(16))
    )
);

include(__DIR__."/../includes/auth/oauth2-proc.php");
```

The `oauth2-proc.php` script automatically handles CSRF protection for OAuth2 providers. If the script is still executing after the `include` line for that file, then all checks have passed and authentication is genuine and successful.

22.3.1 Considerations for Telegram bot tokens

The guide for setting up *Telegram authentication* makes a clear warning statement not to use the bot token that is used for authentication for any other purpose and to never share it with any third parties, citing possible attack vectors associated with other users being able to impersonate the Telegram service and other users when authenticating to FreeField with Telegram.

The technical reason behind this warning is the way in which Telegram handles validation and signing of authentication requests. When a user authenticates to Telegram, their web browsers are given a set of fields that they in turn automatically pass on to FreeField, including their username and identifying information, along with a hash field that is used to check the authenticity of the received data. Citing the [Telegram documentation on login widgets](#):

You can verify the authentication and the integrity of the data received by comparing the received hash parameter with the hexadecimal representation of the HMAC-SHA-256 signature of the **data-check-string** with the SHA256 hash of the bot's token used as a secret key.

Data-check-string is a concatenation of all received fields, sorted in alphabetical order, in the format `key=<value>` with a line feed character (`'\n'`, `0xA0`) used as separator – e.g., `'auth_date=<auth_date>\nfirst_name=<first_name>\nid=<id>\nusername=<username>'`.

Since the bot token is used as the secret key, anyone with the bot token will be able to construct a data-check-string with the name, ID and username of any user and then sign it using the bot token to get a valid hash value.

In general, this method of authentication is secure, but it requires that the bot token is kept secret and closely guarded - which it *would* be in most applications that implement the Telegram API, since most likely, the developers of those applications are the only ones who'd generate tokens. However, for FreeField, the risk is greater that end user administrators don't realize the full potential usage area for the tokens, given that most end users would have little to no developer experience and that Telegram, upon issuance of the token, does not state that it has to be stored securely.

This is also the reason that bot tokens are masked on webhooks that trigger Telegram messages. Despite our insistence in the *Telegram webhooks* documentation to not re-use the authentication bot token for webhooks, some users will inevitably do it anyway, and in an effort to prevent the authentication bot token from being leaked through the webhook list, we've chosen to always treat the token as if it was used for authentication, i.e. always masking it when displayed to users, never sending it to the web browser, and storing it in encrypted form in the configuration files.

22.4 Permissions

Many functions in FreeField are not supposed to be accessible by regular users, and a permissions system is implemented to granularize and enforce access restrictions for those resources.

Permissions are stored in the configuration files as a settings path of the `PermissionsOption` type:

```
// includes/config/defs.php

"permissions/level/admin/updates/general" => array(
    "domain" => "perms",
    "section" => "admin",
    "default" => PermissionOption::LEVEL_HOST,
    "option" => new PermissionOption()
),
```

All permissions are stored as sub-keys under `permissions/level` and are assigned a default permission level that corresponds to one of the default permission levels in the `PermissionOption` class.

```
// includes/config/types.php

class PermissionOption extends DefaultOption {
    /*
     * Constants representing the default permission levels.
     */
    const LEVEL_HOST = 250;
    const LEVEL_ADMIN = 200;
    const LEVEL_MODERATOR = 160;
    const LEVEL_SUBMITTER = 120;
    const LEVEL_REGISTERED = 80;
    const LEVEL_READ_ONLY = 40;
    const LEVEL_ANONYMOUS = 0;

    /* ... more functions ... */
}
```

Permissions must be checked for the current user before performing potentially dangerous operations. For example, to check for the above permission under the `permissions/level/admin/updates/general` setting, use:

```
if (!Auth::getCurrentUser()->hasPermission("admin/updates/general")) {
    header("HTTP/1.1 303 See Other");
    header("Location: /return/path.php");
    exit;
}
```

If there is a need to add a new permission for something, add it in the same way as any other configuration file entries. This is explained further in the developer documentation for configuration.

CHAPTER 23

Translating FreeField

FreeField relies on crowd-sourced translation. If you speak another language than English natively, you are very welcome to contribute your language to the project.

23.1 Getting started

FreeField uses the Crowdin localization platform. To get started on Crowdin, [sign up for an account](#) if you don't already have one, then go to the FreeField project using this URL:

<https://crowdin.com/project/freefield>

Click on your language to start translating. If your language isn't listed, please [create an issue on the issue tracker on GitHub](#) stating which language (and if applicable, which regional dialect/variant) you wish to localize the project in, and the language/locale will be created for you.

23.2 Files to translate

The file view on Crowdin lists several files to translate, labeled by the types of strings that the file contains. Some files are prioritized; these are labeled with an up arrow (↑) on the right hand side and should be localized first. Low priority files, labeled with a down arrow (↓), should be localized last.

Note: You do not have to translate everything! Translate as many strings as you want, but please start translating the high priority files first.

23.3 Questions

If you have any questions about translating FreeField, do not hesitate to raise an issue on the [issue tracker on GitHub](#) with your question. We value feedback on anything that might be unclear, and your input could help us write better

documentation for others who many have the same question as you.