

Privacy

- [Privacy](#)
- [Adding the Personal Data Eraser to Your Plugin](#)
- [Adding the Personal Data Exporter to Your Plugin](#)
- [Privacy Related Options, Hooks and Capabilities](#)
- [index](#)

Privacy

Are you writing a plugin that handles personal data – things like names, addresses, and other things that can be used to identify a person? You'll want to take care with that data and protect the privacy of your users and visitors.

What is Privacy?

WordPress.org made several enhancements ahead of Europe's General Data Protection Regulation. Following the launch of this work, we have made Privacy a permanent focus in core trac development, which will allow us to continue making enhancements on privacy and data protection outside specific legislation.

But what kind of issues might fall under the definition of "privacy", and how do we define it? Although privacy requirements vary widely across countries, cultures, and legal systems, there are several general principles applicable across any situation:

- **Consent and choice:** giving users (and site visitors) choices and options over the uses of their data, and requiring clear, specific, and informed opt-in;
- **Purpose legitimacy and specification:** only collect and use the personal data for the purpose it was intended for, and for which the user was clearly informed of in advance;
- **Collection limitation:** only collect the user data which is needed; don't make extra copies of data or combine your data with data from other plugins if you can avoid it;
- **Data minimization:** restrict the processing of data, as well as the number of people who have access to it, to the minimum uses and people necessary;
- **Use, retention and disclosure limitation:** delete data which is no longer needed, both in active use and in archives, by both the recipient and any third parties;
- **Accuracy and quality:** ensure that the data collected and used is correct, relevant, and up-to-date, especially if inaccurate or poor data could adversely impact the user;
- **Openness, transparency and notice:** inform users how their data is being collected, used, and shared, as well as any rights they have over those uses;
- **Individual participation and access:** give users a means to access or download their data;
- **Accountability:** documenting the uses of data, protecting it in transit and in use by third parties, and preventing misuse and breaches as much as is possible;
- **Information security:** protecting data through appropriate technical and security measures;
- **Privacy compliance:** ensuring that the work meets the privacy regulations of the location where it will be used to collect and process people's data.

(Source: [ISO 29100/Privacy Framework standard](#))

While not all of these principles will be applicable across all situations and uses, using them in the development process can help to ensure user trust.

Privacy By Design

Many of these principles are espoused in the Privacy by Design framework, which states that:

- Privacy should be proactive, not reactive, and must anticipate privacy issues before they reach the user. Privacy must also be preventative, not remedial.
- Privacy should be the default setting. The user should not have to take actions to secure their privacy, and consent for data sharing should not be assumed.
- Privacy should be built into design as a core function, not an add-on.
- Privacy should be positive sum: there should be no trade-off between privacy and security, privacy and safety, or privacy and service provision.
- Privacy should offer end-to-end lifecycle protection through data minimization, minimal data retention, and regular deletion of data which is no longer required.
- The privacy standards used on your plugin (and service, if applicable) should be visible, transparent, open, documented and independently verifiable.
- Privacy should be user-centric. People should be given options such as granular privacy choices, maximized privacy defaults, detailed privacy information notices, user-friendly options, and clear notification of changes.

Food for Thought for Your Plugin

To help your plugin be ready, we recommend going through the following list of questions for every plugin that you make:

1. How does your plugin handle personal data? Use `wp_add_privacy_policy_content()` to disclose to your users any of the following:
 - Does the plugin share personal data with third parties (e.g. to outside APIs/servers). If so, what data does it share with which third parties and do they have a published privacy policy you can provide a link to?
 - Does the plugin collect personal data? If so, what data and where is it stored? Think about places like user data/meta, options, post meta, custom tables, files, etc.
 - Does the plugin use personal data collected by others? If so, what data? Does the plugin pass personal data to a SDK? What does that SDK do with the data?
 - Does the plugin collect telemetry data, directly or indirectly? Loading an image from a third-party source on every install, for example, could indirectly log and track the usage data of all of your plugin installs.

- Does the plugin enqueue Javascript, tracking pixels or embed iframes from a third party (third party JS, tracking pixels and iframes can collect visitor's data/actions, leave cookies, etc.)?
 - Does the plugin store things in the browser? If so, where and what? Think about things like cookies, local storage, etc.
2. If your plugin collects personal data...
 - Does it provide a personal data exporter?
 - Does it provide a personal data eraser callback?
 - For what reasons (if any) does the plugin refuse to erase personal data? (e.g. order not yet completed, etc) – those should be disclosed as well.
 3. Does the plugin use error logging? Does it avoid logging personal data if possible? Could you use things like `wp_privacy_anonymize_data()` to minimize the personal data logged? How long are log entries kept? Who has access to them?
 4. In wp-admin, what role/capabilities are required to access/see personal data? Are they sufficient?
 5. What personal data is exposed on the front end of the site by the plugin? Does it appear to logged-in and logged-out users? Should it?
 6. What personal data is exposed in REST API endpoints by the plugin? Does it appear to logged-in and logged-out users? What roles/capabilities are required to see it? Are those appropriate?
 7. Does the plugin properly remove/clean-up data, including especially personal data:
 - During uninstall of the plugin?
 - When a related item is deleted (e.g. from the post meta or any post-referencing rows in another table)?
 - When a user is deleted (e.g. from any user referencing rows in a table)?
 8. Does the plugin provide controls to reduce the amount of personal data required?
 9. Does the plugin share personal data with SDKs or APIs only when the SDK or API requires it, or is the plugin also sharing personal data that is optional?
 10. Does the amount of personal data collected or shared by this plugin change when certain other plugins are also installed?

External Resources

- [Privacy Blog](#)
- [WordPress.org Privacy Policy](#)

Adding the Personal Data Eraser to Your Plugin

In WordPress 4.9.6, new tools were added to make compliance easier with laws like the European Union's General Data Protection Regulation, or GDPR for short. Among the tools added is a Personal Data Removal tool which supports erasing/anonymizing personal data for a given user. It does NOT delete registered user accounts – that is still a separate step the admin can choose whether or not to do.

In addition to the personal data stored in things like WordPress comments, plugins can also hook into the eraser feature to erase the personal data they collect, whether it be in something like postmeta or even an entirely new Custom Post Type (CPT).

Like the exporters, the "key" for all the erasers is the user's email address – this was chosen because it supports erasing personal data for both full-fledged registered users and also unregistered users (e.g. like a logged out commenter).

However, since performing a personal data erase is a destructive process, we don't want to just do it without confirming the request, so the admin-facing user interface starts all requests by having the admin enter the username or email address making the request and then sends them a link to click to confirm their request. Once a request has been confirmed, the admin can kick off personal data erasure for the user, or force one if the need arises.

The way the personal data export is erased is similar to how the personal data exporters – and relies on hooking "eraser" callbacks to do the dirty work of erasing the data. When the admin clicks on the remove personal data link, an AJAX loop begins that iterates over all the erasers registered in the system, one at a time. In addition to erasers built into core, plugins can register their own eraser callbacks.

The eraser callback interface is designed to be as simple as possible. An eraser callback receives the email address we are working with, and a page parameter as well. The page parameter (which starts at 1) is used to avoid plugins potentially causing timeouts by attempting to erase all the personal data they've collected at once. A well behaved plugin will limit the amount of data it attempts to erase per page (e.g. 100 posts, 200 comments, etc.)

The eraser callback replies whether items containing personal data were erased, whether any items containing personal data were retained, an array of messages to present to the admin (explaining why items that were retained were retained) and whether it is done or not. If an eraser callback reports that it is not done, it will be called again (in a separate request) with the page parameter incremented by 1.

When all the exporters have been called to completion, the admin user interface is updated to show whether or not all personal data found was erased, and any messages explaining why personal data was retained.

Let's work on a hypothetical plugin which adds commenter location data to comments. Let's assume the plugin has used `add_comment_meta` to add location data using `meta_key`'s of `latitude` and `longitude`.

The first thing the plugin needs to do is to create an eraser function that accepts an email address and a page, e.g.:

```
/**
 * Removes any stored location data from a user's comment meta for the supplied email address.
 *
 * @param string $email_address  email address to manipulate
 * @param int    $page           pagination
 *
 * @return array
 */
function wporg_remove_location_meta_from_comments_for_email( $email_address, $page = 1 ) {
    $number = 500; // Limit us to avoid timing out
    $page    = (int) $page;

    $comments = get_comments(
        array(
            'author_email' => $email_address,
            'number'       => $number,
            'paged'        => $page,
            'order_by'     => 'comment_ID',
            'order'        => 'ASC',
        )
    );

    $items_removed = false;

    foreach ( (array) $comments as $comment ) {
        $latitude = get_comment_meta( $comment->comment_ID, 'latitude', true );
        $longitude = get_comment_meta( $comment->comment_ID, 'longitude', true );

        if ( ! empty( $latitude ) ) {
            delete_comment_meta( $comment->comment_ID, 'latitude' );
        }
    }
}
```

```

        $items_removed = true;
    }

    if ( ! empty( $longitude ) ) {
        delete_comment_meta( $comment->comment_ID, 'longitude' );
        $items_removed = true;
    }
}

// Tell core if we have more comments to work on still
$done = count( $comments ) < $number;
return array(
    'items_removed' => $items_removed,
    'items_retained' => false, // always false in this example
    'messages'      => array(), // no messages in this example
    'done'          => $done,
);
}

```

The next thing the plugin needs to do is to register the callback by filtering the eraser array using the `wp_privacy_personal_data_erasers` filter.

When registering you provide a friendly name for the eraser (to aid in debugging – this friendly name is not shown to anyone at this time) and the callback, e.g.

```

/**
 * Registers all data erasers.
 *
 * @param array $exporters
 *
 * @return mixed
 */
function wporg_register_privacy_erasers( $erasers ) {
    $erasers['my-plugin-slug'] = array(
        'eraser_friendly_name' => __( 'Comment Location Plugin', 'text-domain' ),
        'callback'             => 'wporg_remove_location_meta_from_comments_for_email',
    );
    return $erasers;
}

add_filter( 'wp_privacy_personal_data_erasers', 'wporg_register_privacy_erasers' );

```

And that's all there is to it! Your plugin will now clean up its personal data!

Adding the Personal Data Exporter to Your Plugin

In WordPress 4.9.6, new tools were added to make compliance easier with laws like the European Union's General Data Protection Regulation, or GDPR for short. Among the tools added is a Personal Data Export tool which supports exporting all the personal data for a given user in a ZIP file. In addition to the personal data stored in things like WordPress comments, plugins can also hook into the exporter feature to export the personal data they collect, whether it be in something like postmeta or even an entirely new Custom Post Type (CPT).

The "key" for all the exports is the user's email address – this was chosen because it supports exporting personal data for both full-fledged registered users and also unregistered users (e.g. like a logged out commenter).

However, since assembling a personal data export could be an intensive process and will likely contain sensitive data, we don't want to just generate it and email it to the requestor without confirming the request, so the admin-facing user interface starts all requests by having the admin enter the username or email address making the request and then sends them a link to click to confirm their request.

Once a request has been confirmed, the admin can generate and download or directly email the personal data export ZIP file for the user, or do the export anyways if the need arises. Inside the ZIP file the user receives, they will find a "mini website" with an index HTML page containing their personal data organized in groups (e.g. a group for comments, etc.)

Whether the admin downloads the personal data export ZIP file or sends it directly to the requestor, the way the personal data export is assembled is identical – and relies on hooking "exporter" callbacks to do the dirty work of collecting all the data for the export. When the admin clicks on the download or email link, an AJAX loop begins that iterates over all the exporters registered in the system, one at a time. In addition to exporters built into core, plugins can register their own exporter callbacks.

The exporter callback interface is designed to be as simple as possible. A exporter callback receives the email address we are working with and a page parameter as well. The page parameter (which starts at 1) is used to avoid plugins potentially causing timeouts by attempting to export all the personal data they've collected at once. A well behaved plugin will limit the amount of data it attempts to export per page (e.g. 100 posts, 200 comments, etc.)

The exporter callback replies with whatever data it has for that email address and page and whether it is done or not. If a exporter callback reports that it is not done, it will be called again (in

a separate request) with the page parameter incremented by 1. Exporter callbacks are expected to return an array of items for the export. Each item contains an a group identifier for the group of which the item is a part (e.g. comments, posts, orders, etc.), an optional group label (translated), an item identifier (e.g. comment-133) and then an array of name, value pairs containing the data to be exported for that item.

It is noteworthy that the value could be a media path, in which case a link to the media file will be added to the index HTML page in the export.

When all the exporters have been called to completion, WordPress first assembles an "index" HTML document that serves as the heart of the export report. If a plugin reports additional data for an item that WordPress or another plugin has already added, all the data for that item will be presented together.

Exports are cached on the server for 3 days and then deleted.

A plugin can register one or more exporters, but most plugins will only need one. Let's work on a hypothetical plugin which adds location data for the commenter to comments.

First, let's assume the plugin has used `add_comment_meta` to add location data using `meta_key`'s of `latitude` and `longitude`.

The first thing the plugin needs to do is to create an exporter function that accepts an email address and a page, e.g.:

```
/**
 * Export user meta for a user using the supplied email.
 *
 * @param string $email_address email address to manipulate
 * @param int    $page          pagination
 *
 * @return array
 */
function wporg_export_user_data_by_email( $email_address, $page = 1 ) {
    $number = 500; // Limit us to avoid timing out
    $page    = (int) $page;

    $export_items = array();

    $comments = get_comments(
        array(
            'author_email' => $email_address,
            'number'       => $number,
```

```

        'paged'          => $page,
        'order_by'       => 'comment_ID',
        'order'          => 'ASC',
    )
);

foreach ( (array) $comments as $comment ) {
    $latitude = get_comment_meta( $comment->comment_ID, 'latitude', true );
    $longitude = get_comment_meta( $comment->comment_ID, 'longitude', true );

    // Only add location data to the export if it is not empty.
    if ( ! empty( $latitude ) ) {
        // Most item IDs should look like postType-postID. If you don't have a post, comment or
other ID to work with,
        // use a unique value to avoid having this item's export combined in the final report
with other items
        // of the same id.
        $item_id = "comment-{$comment->comment_ID}";

        // Core group IDs include 'comments', 'posts', etc. But you can add your own group IDs
as needed
        $group_id = 'comments';

        // Optional group label. Core provides these for core groups. If you define your own
group, the first
        // exporter to include a label will be used as the group label in the final exported
report.
        $group_label = __( 'Comments', 'text-domain' );

        // Plugins can add as many items in the item data array as they want.
        $data = array(
            array(
                'name' => __( 'Commenter Latitude', 'text-domain' ),
                'value' => $latitude,
            ),
            array(
                'name' => __( 'Commenter Longitude', 'text-domain' ),
                'value' => $longitude,
            ),
        );
    }
};

```

```

        $export_items[] = array(
            'group_id'    => $group_id,
            'group_label' => $group_label,
            'item_id'     => $item_id,
            'data'        => $data,
        );
    }
}

// Tell core if we have more comments to work on still.
$done = count( $comments ) > $number;
return array(
    'data' => $export_items,
    'done' => $done,
);
}

```

The next thing the plugin needs to do is to register the callback by filtering the exporter array using the `wp_privacy_personal_data_exporters` filter.

When registering you provide a friendly name for the export (to aid in debugging – this friendly name is not shown to anyone at this time) and the callback, e.g.

```

/**
 * Registers all data exporters.
 *
 * @param array $exporters
 *
 * @return mixed
 */
function wporg_register_user_data_exporters( $exporters ) {
    $exporters['my-plugin-slug'] = array(
        'exporter_friendly_name' => __( 'Comment Location Plugin', 'text-domain' ),
        'callback'               => 'my_plugin_exporter',
    );
    return $exporters;
}

add_filter( 'wp_privacy_personal_data_exporters', 'wporg_register_user_data_exporters' );

```

And that's all there is to it! Your plugin will now provide data for the export!

Privacy Related Options, Hooks and Capabilities

The privacy tools were originally introduced in WordPress 4.9.6. These tools are designed to allow (and encourage) developers to use them as part of the Privacy Exporter, Privacy Eraser and the Privacy Policy Guide.

Since then, several newer hooks have been introduced to expand on the available capabilities. These hooks allow developers to include additional personal data in export and erasure requests, and introduce suggested content for the privacy policy guide.

Along with the ability to control these tools, there are several new filters for use with the request and confirmation emails, enabling finer-grained controls over these notifications.

Options

- `wp_page_for_privacy_policy` – contains the page ID of a site's privacy page.

Actions

- `user_request_action_confirmed` – fired when a user confirms a privacy request.
- `wp_privacy_delete_old_export_files` – a scheduled action used to prune old exports from the personal data exports folder.
- `wp_privacy_personal_data_erased` – fired after the last page of the last eraser is complete.
- `wp_privacy_personal_data_export_file` – used to create a personal data export file as part of the export flow.
- `wp_privacy_personal_data_export_file_created` – fires after a personal data export file has been created.

Filters

- `privacy_policy_url` – filters the URL of the privacy policy page.
- `the_privacy_policy_link` – filters the privacy policy page link HTML.

- `wp_get_default_privacy_policy_content` - filters the default content suggested for inclusion through the privacy policy guide.
- `user_request_action_confirmed_message` - allows modifying the action confirmation message displayed to the user.
- `user_request_action_description` - filters the user action description.
- `user_request_action_email_content` - filters the text of the email sent when an account action is attempted.
- `user_request_action_email_headers` - filters the headers of the email sent when an account action is attempted.
- `user_request_action_email_subject` - filters the subject of the email sent when an account action is attempted.
- `user_request_confirmed_email_content` - filters the body of the user request confirmation email.
- `user_request_confirmed_email_headers` - filters the headers of the user request confirmation email.
- `user_request_confirmed_email_subject` - filters the subject of the user request confirmation email.
- `user_request_confirmed_email_to` - filters the recipient of the data request confirmation notification.
- `user_request_key_expiration` - filters the expiration time of confirmation keys for user requests.
- `wp_privacy_additional_user_profile_data` - filter to extend the user's profile data for the privacy exporter.
- `wp_privacy_export_expiration` - controls how old export files are allowed to get, default is 3 days.
- `wp_privacy_personal_data_email_content` - allows modifying the email message send to users with their personal data export file link.
- `wp_privacy_personal_data_email_headers` - filters the headers of the email sent with a personal data export file.
- `wp_privacy_personal_data_email_subject` - filters the subject of the email sent when an export request is completed.
- `wp_privacy_personal_data_email_to` - filters the recipient of the personal data export email notification.

[info] `wp_privacy_personal_data_email_to` should be used with great caution to avoid sending the data export link to the wrong recipient email address(es).[/info]

- `wp_privacy_personal_data_erasers` - supports registration of core and plugin personal data erasers.
- `wp_privacy_personal_data_erasure_page` - Filters a page of personal data eraser data. Allows the erasure response to be consumed by destinations in addition to AJAX.
- `wp_privacy_personal_data_exporters` - supports registration of core and plugin personal data exporters.
- `wp_privacy_personal_data_export_page` - filters a page of personal data exporter data. Used to build the export report. Allows the export response to be consumed by destinations in

addition to AJAX.

- `wp_privacy_anonymize_data` – filters the anonymous data for each type.
- `wp_privacy_exports_dir` – filters the directory used to store personal data export files.
- `wp_privacy_exports_url` – filters the URL of the directory used to store personal data export files.
- `user_confirmed_action_email_content` – Filters the body of the user request confirmation email. The email is sent to an administrator when an user request is confirmed.
- `user_erasure_fulfillment_email_to` – Filters the recipient of the data erasure fulfillment notification.
- `user_erasure_complete_email_subject` – Filters the subject of the email sent when an erasure request is completed.
- `user_confirmed_action_email_content` – Filters the body of the data erasure fulfillment notification. The email is sent to a user when a their data erasure request is fulfilled by an administrator.
- `user_erasure_complete_email_headers` – Filters the headers of the data erasure fulfillment notification.

Capabilities

Access to the privacy tools is controlled by a few new capabilities. Administrators (on non-multisite installations) have these capabilities by default. These capabilities are:

- `erase_others_personal_data` – determines if the Erase Personal Data sub-menu is available under Tools.
- `export_others_personal_data` – determines if the Export Personal Data sub-menu is available under Tools.
- `manage_privacy_options` – determines if the Privacy sub-menu is available under Settings.

index

Suggesting text for the site privacy policy

Every plugin that collects, uses, or stores user data, or passes it to an external source or third party, should add a section of suggested text to the privacy policy postbox. This is best done with `wp_add_privacy_policy_content( $plugin_name, $policy_text )`. This will allow site administrators to pull that information into their site's privacy policy.

To make this simpler for the users, the text should address the questions provided in the default privacy policy:

- What personal data we collect and why we collect it
 - Their own manually input information
 - WP: Contact forms
 - WP: Comments
 - WP: Cookies
 - WP: Third party embeds
 - Analytics
- Who we share your data with
- How long we retain your data
- What rights you have over your data
- Where we send your data
- Your contact information
- How we protect your data
- What data breach procedures we have in place
- What third parties we receive data from
- What automated decision making and/or profiling we do with user data
- Any industry regulatory disclosure requirements

While not all of these questions will be applicable to all plugins, we recommend taking care with the sections on data sharing.

Code Example

[info]It is recommended to call `wp_add_privacy_policy_content()` during the `admin_init` action. Calling it outside of an action hook can lead to problems, see ticket [#44142](#) for details.[/info]

[info]Supplemental information can be provided through the use of the specialized `.privacy-policy-tutorial` CSS class. Any content contained within HTML elements that have this CSS class applied will be omitted from the clipboard when the section content is copied.[/info]

```
/**
 * Adds a privacy policy statement.
 */
function wporg_add_privacy_policy_content() {
    if ( ! function_exists( 'wp_add_privacy_policy_content' ) ) {
        return;
    }
    $content = '<p class="privacy-policy-tutorial">' . __( 'Some introductory content for the
suggested text.', 'text-domain' ) . '</p>'
    . '<strong class="privacy-policy-tutorial">' . __( 'Suggested Text:', 'my_plugin_textdomain'
) . '</strong> '
    . sprintf(
        __( 'When you leave a comment on this site, we send your name, email address, IP address
and comment text to example.com. Example.com does not retain your personal data. The
example.com privacy policy is <a href="%1$s" target="_blank">here</a>.', 'text-domain' ),
        'https://example.com/privacy-policy'
    );
    wp_add_privacy_policy_content( 'Example Plugin', wp_kses_post( wpautop( $content, false ) )
);
}

add_action( 'admin_init', 'wporg_add_privacy_policy_content' );
```