L A M P I R A N

```
<?php
/**
 * Front to the WordPress
application. This file doesn't
do anything, but loads
 * wp-blog-header.php which
does and tells WordPress to
load the theme.
 *
 * @package WordPress
 */

/**
 * Tells WordPress to load the
WordPress theme and output it.
 *
 * @var bool
 */
define( 'WP_USE_THEMES', true
);

/** Loads the WordPress
Environment and Template */
require __DIR__ . '/wp-blog-
header.php';
<?php
/**
 * Confirms that the
activation key that is sent in
an email after a user signs
 * up for a new site matches
the key for that user and then
displays confirmation.
 *
 * @package WordPress
 */

define( 'WP_INSTALLING', true
);

/** Sets up the WordPress
Environment. */
require __DIR__ . '/wp-
load.php';

require __DIR__ . '/wp-blog-
header.php';

if ( ! is_multisite() ) {
    wp_redirect(
wp_registration_url() );
    die();
}

$valid_error_codes = array(
'already_active', 'blog_taken'
);

list( $activate_path ) =
explode( '?', wp_unslash(
$_SERVER['REQUEST_URI'] ) );
$activate_cookie       = 'wp-
activate-' . COOKIEHASH;

$key    = '';
$result = null;

if ( isset( $_GET['key'] ) &&
isset( $_POST['key'] ) &&
$_GET['key'] !== $_POST['key']
) {
    wp_die( __( 'A key value
mismatch has been detected.
Please follow the link
provided in your activation
email.' ), __( 'An error
occurred during the
activation' ), 400 );
} elseif ( ! empty(
$_GET['key'] ) ) {
    $key = $_GET['key'];
} elseif ( ! empty(
$_POST['key'] ) ) {
    $key = $_POST['key'];
}

if ( $key ) {
    $redirect_url =
remove_query_arg( 'key' );

    if ( remove_query_arg(
false ) !== $redirect_url ) {
        setcookie(
$activate_cookie, $key, 0,
$activate_path, COOKIE_DOMAIN,
is_ssl(), true );
        wp_safe_redirect(
$redirect_url );
        exit;
    } else {
        $result =
wpmu_activate_signup( $key );
    }
}
```

```
if ( null === $result &&
isset( $_COOKIE[
$activate_cookie ] ) ) {
    $key    = $_COOKIE[
$activate_cookie ];
    $result =
wpmu_activate_signup( $key );
    setcookie(
$activate_cookie, ' ', time()
- YEAR_IN_SECONDS,
$activate_path, COOKIE_DOMAIN,
is_ssl(), true );
}

if ( null === $result || (
is_wp_error( $result ) &&
'invalid_key' === $result-
>get_error_code() ) ) {
    status_header( 404 );
} elseif ( is_wp_error(
$result ) ) {
    $error_code = $result-
>get_error_code();

    if ( ! in_array(
$error_code,
$valid_error_codes, true ) ) {
        status_header( 400 );
    }
}

nocache_headers();

if ( is_object(
$wp_object_cache ) ) {
    $wp_object_cache-
>cache_enabled = false;
}

// Fix for page title.
$wp_query->is_404 = false;

/**
 * Fires before the Site
Activation page is loaded.
 *
 * @since 3.0.0
 */
do_action( 'activate_header'
);

/**
 * Adds an action hook
specific to this page.
 *
 * Fires on {@see 'wp_head'}.
 *
 * @since MU (3.0.0)
 */
function do_activate_header()
{
    /**
     * Fires within the
`<head>` section of the Site
Activation page.
     *
     * Fires on the {@see
'wp_head'} action.
     *
     * @since 3.0.0
     */
    do_action(
'activate_wp_head' );
}
add_action( 'wp_head',
'do_activate_header' );

/**
 * Loads styles specific to
this page.
 *
 * @since MU (3.0.0)
 */
function
wpmu_activate_stylesheet() {
    ?>
    <style type="text/css">
        .wp-activate-container
{ width: 90%; margin: 0 auto;
}
        .wp-activate-container
form { margin-top: 2em; }
        #submit, #key { width:
100%; font-size: 24px; box-
sizing: border-box; }
        #language { margin-
top: 0.5em; }
        .wp-activate-container
.error { background: #f66;
color: #333; }
        span.h3 { padding: 0
8px; font-size: 1.3em; font-
weight: 600; }
```

```
    </style>
    <?php
}
add_action( 'wp_head',
'wpmu_activate_stylesheet' );
add_action( 'wp_head',
'wp_strict_cross_origin_referr
er' );
add_filter( 'wp_robots',
'wp_robots_sensitive_page' );

get_header( 'wp-activate' );

$blog_details = get_site();
?>

<div id="signup-content"
class="widecolumn">
    <div class="wp-activate-
container">
    <?php if ( ! $key ) { ?>

        <h2><?php _e(
'Activation Key Required' );
?></h2>
        <form
name="activateform"
id="activateform"
method="post" action="<?php
echo esc_url(
network_site_url(
$blog_details->path . 'wp-
activate.php' ) ); ?>">
            <p>
                <label
for="key"><?php _e(
'Activation Key:' );
?></label>
                <br /><input
type="text" name="key"
id="key" value="" size="50"
autofocus="autofocus" />
            </p>
            <p class="submit">
                <input
id="submit" type="submit"
name="Submit" class="submit"
value="<?php esc_attr_e(
'Activate' ); ?>" />
            </p>
        </form>
        <?php
    } else {
        if ( is_wp_error(
$result ) && in_array(
$result->get_error_code(),
$valid_error_codes, true ) ) {
            $signup = $result-
>get_error_data();
            ?>
            <h2><?php _e(
'Your account is now active!'
); ?></h2>
            <?php
            echo '<p
class="lead-in">';
            if ( '' ===
$signup->domain . $signup-
>path ) {
                printf(
                    /*
translators: 1: Login URL, 2:
Username, 3: User email
address, 4: Lost password URL.
*/
                    __( 'Your
account has been activated.
You may now <a href="%1$s">log
in</a> to the site using your
chosen username of
&#8220;%2$s&#8221;. Please
check your email inbox at %3$s
for your password and login
instructions. If you do not
receive an email, please check
your junk or spam folder. If
you still do not receive an
email within an hour, you can
<a href="%4$s">reset your
password</a>.' ),
                    esc_url(
network_site_url(
$blog_details->path . 'wp-
login.php', 'login' ) ),
                    esc_html(
$signup->user_login ),
                    esc_html(
$signup->user_email ),
                    esc_url(
wp_lostpassword_url() )
                );
            } else {
                printf(
```

```
                    /*
translators: 1: Site URL, 2:
Username, 3: User email
address, 4: Lost password URL.
*/
                    __( 'Your
site at %1$s is active. You
may now log in to your site
using your chosen username of
&#8220;%2$s&#8221;. Please
check your email inbox at %3$s
for your password and login
instructions. If you do not
receive an email, please check
your junk or spam folder. If
you still do not receive an
email within an hour, you can
<a href="%4$s">reset your
password</a>.' ),
                    sprintf(
'<a
href="http://%1$s">%1$s</a>',
esc_url( $signup->domain .
$blog_details->path ) ),
                    esc_html(
$signup->user_login ),
                    esc_html(
$signup->user_email ),
                    esc_url(
wp_lostpassword_url() )
                );
            }
            echo '</p>';
        } elseif ( null ===
$result || is_wp_error(
$result ) ) {
            ?>
            <h2><?php _e( 'An
error occurred during the
activation' ); ?></h2>
            <?php if (
is_wp_error( $result ) ) : ?>
                <p><?php echo
esc_html( $result-
>get_error_message() ); ?></p>
            <?php endif; ?>
            <?php
        } else {
            $url  = isset(
$result['blog_id'] ) ?
esc_url( get_home_url( (int)
$result['blog_id'] ) ) : '';
            $user =
get_userdata( (int)
$result['user_id'] );
            ?>
            <h2><?php _e(
'Your account is now active!'
); ?></h2>

            <div id="signup-
welcome">
            <p><span
class="h3"><?php _e(
'Username:' ); ?></span> <?php
echo esc_html( $user-
>user_login ); ?></p>
            <p><span
class="h3"><?php _e(
'Password:' ); ?></span> <?php
echo esc_html(
$result['password'] ); ?></p>
            </div>

            <?php
            if ( $url &&
network_home_url( '', 'http' )
!== $url ) :

switch_to_blog( (int)
$result['blog_id'] );
                $login_url =
wp_login_url();

restore_current_blog();
                ?>
                <p
class="view">
                <?php
                    /*
translators: 1: Site URL, 2:
Login URL. */
                    printf(
__( 'Your account is now
activated. <a href="%1$s">View
your site</a> or <a
href="%2$s">Log in</a>' ),
esc_url( $url ), esc_url(
$login_url ) );
                ?>
                </p>
            <?php else : ?>
                <p
class="view">
```

```
                <?php
                    printf(
                        /*
translators: 1: Login URL, 2:
Network home URL. */
                        __(
'Your account is now
activated. <a href="%1$s">Log
in</a> or go back to the <a
href="%2$s">homepage</a>.' ),

esc_url( network_site_url(
$blog_details->path . 'wp-
login.php', 'login' ) ),

esc_url( network_home_url(
$blog_details->path ) )
                        );
                ?>
                </p>
                <?php
                endif;
        }
    }
    ?>
    </div>
</div>
<?php
get_footer( 'wp-activate' );

<?php
/**
 * Handles Comment Post to
WordPress and prevents
duplicate comment posting.
 *
 * @package WordPress
 */

if ( 'POST' !==
$_SERVER['REQUEST_METHOD'] ) {
    $protocol =
$_SERVER['SERVER_PROTOCOL'];
    if ( ! in_array(
$protocol, array( 'HTTP/1.1',
'HTTP/2', 'HTTP/2.0', 'HTTP/3'
), true ) ) {
        $protocol =
'HTTP/1.0';
    }

    header( 'Allow: POST' );
    header( "$protocol 405
Method Not Allowed" );
    header( 'Content-Type:
text/plain' );
    exit;
}

/** Sets up the WordPress
Environment. */
require __DIR__ . '/wp-
load.php';

nocache_headers();

$comment =
wp_handle_comment_submission(
wp_unslash( $_POST ) );
if ( is_wp_error( $comment ) )
{
    $data = (int) $comment-
>get_error_data();
    if ( ! empty( $data ) ) {
        wp_die(
            '<p>' . $comment-
>get_error_message() . '</p>',
            __( 'Comment
Submission Failure' ),
            array(
                'response'  =>
$data,
                'back_link' =>
true,
            )
        );
    } else {
        exit;
    }
}

$user            =
wp_get_current_user();
$cookies_consent = ( isset(
$_POST['wp-comment-cookies-
consent'] ) );

/**
 * Fires after comment cookies
are set.
 *
 * @since 3.4.0
```

```
 * @since 4.9.6 The
`$cookies_consent` parameter
was added.
 *
 * @param WP_Comment $comment
       Comment object.
 * @param WP_User    $user
      Comment author's user
object. The user may not
exist.
 * @param bool
$cookies_consent Comment
author's consent to store
cookies.
 */
do_action(
'set_comment_cookies',
$comment, $user,
$cookies_consent );

$location = empty(
$_POST['redirect_to'] ) ?
get_comment_link( $comment ) :
$_POST['redirect_to'] .
'#comment-' . $comment-
>comment_ID;

// If user didn't consent to
cookies, add specific query
arguments to display the
awaiting moderation message.
if ( ! $cookies_consent &&
'unapproved' ===
wp_get_comment_status(
$comment ) && ! empty(
$comment->comment_author_email
) ) {
    $location = add_query_arg(
        array(
            'unapproved'
 => $comment->comment_ID,
            'moderation-hash'
=> wp_hash( $comment-
>comment_date_gmt ),
        ),
        $location
    );
}

/**
 * Filters the location URI to
send the commenter after
posting.
 *
 * @since 2.0.5
 *
 * @param string     $location
The 'redirect_to' URI sent via
$_POST.
 * @param WP_Comment $comment
 Comment object.
 */
$location = apply_filters(
'comment_post_redirect',
$location, $comment );

wp_safe_redirect( $location );
exit;

<?php
/**
 * The base configuration for
WordPress
 *
 * The wp-config.php creation
script uses this file during
the installation.
 * You don't have to use the
web site, you can copy this
file to "wp-config.php"
 * and fill in the values.
 *
 * This file contains the
following configurations:
 *
 * * Database settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link
https://wordpress.org/document
ation/article/editing-wp-
config-php/
 *
 * @package WordPress
 */

// ** Database settings - You
can get this info from your
web host ** //
```

```
/** The name of the database
for WordPress */
define( 'DB_NAME', 'hardiman'
);

/** Database username */
define( 'DB_USER',
'adminadmin' );

/** Database password */
define( 'DB_PASSWORD',
'abcd12345' );

/** Database hostname */
define( 'DB_HOST', 'localhost'
);

/** Database charset to use in
creating database tables. */
define( 'DB_CHARSET',
'utf8mb4' );

/** The database collate type.
Don't change this if in doubt.
*/
define( 'DB_COLLATE', '' );

define( 'WP_MEMORY_LIMIT',
'256M' );

define(
'FLUENTMAIL_SPARKPOST_API_KEY'
,
'c72053e6c16fb2b7cb4cabb86274e
4c0255fe2ee' );
/**#@+
 * Authentication unique keys
and salts.
 *
 * Change these to different
unique phrases! You can
generate these using
 * the {@link
https://api.wordpress.org/secr
et-key/1.1/salt/ WordPress.org
secret-key service}.
 *
 * You can change these at any
point in time to invalidate
all existing cookies.
 * This will force all users
to have to log in again.
 *
 * @since 2.6.0
 */
define( 'AUTH_KEY',
'5UV!ip:.oeK-NA&pW-
>Wr?U^/_IV7_kt3#q]Hz^lXK9@{H!e
!8y#tas[j;YH4*D7' );
define( 'SECURE_AUTH_KEY',
 ';W1]NmiO=p^SS`%N{ZR;WCI+7/ra
,V)@W@ap=/<Eg>y:D:RM%j*)Zb^F$E
$}}3@D' );
define( 'LOGGED_IN_KEY',    'm
fUu{K84uEy>q-
k{/>ZyK/q)Hkq8q+M?W~vl6)2ZNZz8
YN{9&0`7*?W & -x[}Q' );
define( 'NONCE_KEY',
 't}dL#X]2eS$(-{D q
&E>K:LFYWgl&[(Td%QusY3XxOI,In)
aDHfZ*4s>Jw,(#lI' );
define( 'AUTH_SALT',
 '|K+r@!^r[)C9yz@.>^/!Y+]3Hs.F
@$!.yZ]noX&/YGKXM!&=3>LfJx>x$B
12G/a+' );
define( 'SECURE_AUTH_SALT',
'4.xnk7,hG,u%D--
^f>#(@/9bsbbaWz7x7xb4#n0;f5.!.
Js2%M|R+a&^3;waR^.k' );
define( 'LOGGED_IN_SALT',
')@KWck;%lf,VK~tv6$GtB#~S-
Tb&a1gSe9&Jh/PnH$`fE6RvtTA++iM
AHT%;]5L:' );
define( 'NONCE_SALT',
'S.0*&3:E](J<gYhCi3&,=uth8uf]S
8gG@?zp0S!M[BDD[sA<zDhR?`$Iy8S
3h7qm' );

/**#@-*/

/**
 * WordPress database table
prefix.
 *
 * You can have multiple
installations in one database
if you give each
 * a unique prefix. Only
numbers, letters, and
underscores please!
 */
$table_prefix = 'wp_';

/**
```

```
 * For developers: WordPress
debugging mode.
 *
 * Change this to true to
enable the display of notices
during development.
 * It is strongly recommended
that plugin and theme
developers use WP_DEBUG
 * in their development
environments.
 *
 * For information on other
constants that can be used for
debugging,
 * visit the documentation.
 *
 * @link
https://wordpress.org/document
ation/article/debugging-in-
wordpress/
 */
define( 'WP_DEBUG', false );

/* Add any custom values
between this line and the
"stop editing" line. */


/* That's all, stop editing!
Happy publishing. */

/** Absolute path to the
WordPress directory. */
if ( ! defined( 'ABSPATH' ) )
{
    define( 'ABSPATH', __DIR__
. '/' );
}

/** Sets up WordPress vars and
included files. */
require_once ABSPATH . 'wp-
settings.php';

<?php
/**
 * A pseudo-cron daemon for
scheduling WordPress tasks.
 *
 * WP-Cron is triggered when
the site receives a visit. In
the scenario
 * where a site may not
receive enough visits to
execute scheduled tasks
 * in a timely manner, this
file can be called directly or
via a server
 * cron daemon for X number of
times.
 *
 * Defining DISABLE_WP_CRON as
true and calling this file
directly are
 * mutually exclusive and the
latter does not rely on the
former to work.
 *
 * The HTTP request to this
file will not slow down the
visitor who happens to
 * visit when a scheduled cron
event runs.
 *
 * @package WordPress
 */

ignore_user_abort( true );

if ( ! headers_sent() ) {
    header( 'Expires: Wed, 11
Jan 1984 05:00:00 GMT' );
    header( 'Cache-Control:
no-cache, must-revalidate,
max-age=0' );
}

// Don't run cron until the
request finishes, if possible.
if ( PHP_VERSION_ID >= 70016
&& function_exists(
'fastcgi_finish_request' ) ) {
    fastcgi_finish_request();
} elseif ( function_exists(
'litespeed_finish_request' ) )
{

litespeed_finish_request();
}
```

```
if ( ! empty( $_POST ) ||
defined( 'DOING_AJAX' ) ||
defined( 'DOING_CRON' ) ) {
    die();
}

/**
 * Tell WordPress the cron
task is running.
 *
 * @var bool
 */
define( 'DOING_CRON', true );

if ( ! defined( 'ABSPATH' ) )
{
    /** Set up WordPress
environment */
    require_once __DIR__ .
'/wp-load.php';
}

// Attempt to raise the PHP
memory limit for cron event
processing.
wp_raise_memory_limit( 'cron'
);

/**
 * Retrieves the cron lock.
 *
 * Returns the uncached
`doing_cron` transient.
 *
 * @ignore
 * @since 3.3.0
 *
 * @global wpdb $wpdb
WordPress database abstraction
object.
 *
 * @return string|int|false
Value of the `doing_cron`
transient, 0|false otherwise.
 */
function _get_cron_lock() {
    global $wpdb;

    $value = 0;
    if (
wp_using_ext_object_cache() )
{
        /*
         * Skip local cache
and force re-fetch of
doing_cron transient
         * in case another
process updated the cache.
         */
        $value = wp_cache_get(
'doing_cron', 'transient',
true );
    } else {
        $row = $wpdb->get_row(
$wpdb->prepare( "SELECT
option_value FROM $wpdb-
>options WHERE option_name =
%s LIMIT 1",
'_transient_doing_cron' ) );
        if ( is_object( $row )
) {
            $value = $row-
>option_value;
        }
    }

    return $value;
}

$crons =
wp_get_ready_cron_jobs();
if ( empty( $crons ) ) {
    die();
}

$gmt_time = microtime( true );

// The cron lock: a unix
timestamp from when the cron
was spawned.
$doing_cron_transient =
get_transient( 'doing_cron' );

// Use global $doing_wp_cron
lock, otherwise use the GET
lock. If no lock, try to grab
a new lock.
if ( empty( $doing_wp_cron ) )
{
    if ( empty(
$_GET['doing_wp_cron'] ) ) {
        // Called from
external script/job. Try
setting a lock.
```

```
        if (
$doing_cron_transient && (
$doing_cron_transient +
WP_CRON_LOCK_TIMEOUT >
$gmt_time ) ) {
            return;
        }
        $doing_wp_cron
 = sprintf( '%.22F',
microtime( true ) );
        $doing_cron_transient
= $doing_wp_cron;
        set_transient(
'doing_cron', $doing_wp_cron
);
    } else {
        $doing_wp_cron =
$_GET['doing_wp_cron'];
    }
}

/*
 * The cron lock (a unix
timestamp set when the cron
was spawned),
 * must match $doing_wp_cron
(the "key").
 */
if ( $doing_cron_transient !==
$doing_wp_cron ) {
    return;
}

foreach ( $crons as $timestamp
=> $cronhooks ) {
    if ( $timestamp >
$gmt_time ) {
        break;
    }

    foreach ( $cronhooks as
$hook => $keys ) {

        foreach ( $keys as $k
=> $v ) {

            $schedule =
$v['schedule'];

            if ( $schedule ) {
                $result =
wp_reschedule_event(
$timestamp, $schedule, $hook,
$v['args'], true );

                if (
is_wp_error( $result ) ) {
                    error_log(

sprintf(
                            /*
translators: 1: Hook name, 2:
Error code, 3: Error message,
4: Event data. */

__( 'Cron reschedule event
error for hook: %1$s, Error
code: %2$s, Error message:
%3$s, Data: %4$s' ),

$hook,

$result->get_error_code(),

$result->get_error_message(),

wp_json_encode( $v )
                        )
                    );

                    /**
                     * Fires
when an error happens
rescheduling a cron event.
                     *
                     * @since
6.1.0
                     *
                     * @param
WP_Error $result The WP_Error
object.
                     * @param
string   $hook   Action hook
to execute when the event is
run.
                     * @param
array    $v      Event data.
                     */
                    do_action(
'cron_reschedule_event_error',
$result, $hook, $v );
                }
            }
```

```
            $result =
wp_unschedule_event(
$timestamp, $hook, $v['args'],
true );

            if ( is_wp_error(
$result ) ) {
                error_log(
                    sprintf(
                        /*
translators: 1: Hook name, 2:
Error code, 3: Error message,
4: Event data. */
                        __(
'Cron unschedule event error
for hook: %1$s, Error code:
%2$s, Error message: %3$s,
Data: %4$s' ),
                        $hook,

$result->get_error_code(),

$result->get_error_message(),

wp_json_encode( $v )
                    )
                );

                /**
                 * Fires when
an error happens unscheduling
a cron event.
                 *
                 * @since
6.1.0
                 *
                 * @param
WP_Error $result The WP_Error
object.
                 * @param
string   $hook   Action hook
to execute when the event is
run.
                 * @param
array    $v      Event data.
                 */
                do_action(
'cron_unschedule_event_error',
$result, $hook, $v );
            }

            /**
             * Fires scheduled
events.
             *
             * @ignore
             * @since 2.1.0
             *
             * @param string
$hook Name of the hook that
was scheduled to be fired.
             * @param array
 $args The arguments to be
passed to the hook.
             */

do_action_ref_array( $hook,
$v['args'] );

            // If the hook ran
too long and another cron
process stole the lock, quit.
            if (
_get_cron_lock() !==
$doing_wp_cron ) {
                return;
            }
        }
    }
}

if ( _get_cron_lock() ===
$doing_wp_cron ) {
    delete_transient(
'doing_cron' );
}

die();

<?php
/**
 * Outputs the OPML XML format
for getting the links defined
in the link
 * administration. This can be
used to export links from one
blog over to
 * another. Links aren't
exported by the WordPress
export, so this file handles
 * that.
 *
```

```
 * This file is not added by
default to WordPress theme
pages when outputting
 * feed links. It will have to
be added manually for browsers
and users to pick
 * up that this file exists.
 *
 * @package WordPress
 */

require_once __DIR__ . '/wp-
load.php';

header( 'Content-Type:
text/xml; charset=' .
get_option( 'blog_charset' ),
true );
$link_cat = '';
if ( ! empty(
$_GET['link_cat'] ) ) {
    $link_cat =
$_GET['link_cat'];
    if ( ! in_array(
$link_cat, array( 'all', '0'
), true ) ) {
        $link_cat = absint(
(string) urldecode( $link_cat
) );
    }
}

echo '<?xml version="1.0"?' .
">\n";
?>
<opml version="1.0">
    <head>
        <title>
        <?php
            /* translators:
%s: Site title. */
            printf( __( 'Links
for %s' ), esc_attr(
get_bloginfo( 'name',
'display' ) ) );
        ?>
        </title>
        <dateCreated><?php
echo gmdate( 'D, d M Y H:i:s'
); ?> GMT</dateCreated>
        <?php
        /**
         * Fires in the OPML
header.
         *
         * @since 3.0.0
         */
        do_action( 'opml_head'
);
        ?>
    </head>
    <body>
<?php
if ( empty( $link_cat ) ) {
    $cats = get_categories(
        array(
            'taxonomy'     =>
'link_category',
            'hierarchical' =>
0,
        )
    );
} else {
    $cats = get_categories(
        array(
            'taxonomy'     =>
'link_category',
            'hierarchical' =>
0,
            'include'      =>
$link_cat,
        )
    );
}

foreach ( (array) $cats as
$cat ) :
    /** This filter is
documented in wp-
includes/bookmark-template.php
*/
    $catname = apply_filters(
'link_category', $cat->name );

    ?>
<outline type="category"
title="<?php echo esc_attr(
$catname ); ?>">
    <?php
    $bookmarks =
get_bookmarks( array(
'category' => $cat->term_id )
);
```

```
    foreach ( (array)
$bookmarks as $bookmark ) :
        /**
         * Filters the OPML
outline link title text.
         *
         * @since 2.2.0
         *
         * @param string
$title The OPML outline title
text.
         */
        $title =
apply_filters( 'link_title',
$bookmark->link_name );
        ?>
<outline text="<?php echo
esc_attr( $title ); ?>"
type="link" xmlUrl="<?php echo
esc_url( $bookmark->link_rss
); ?>" htmlUrl="<?php echo
esc_url( $bookmark->link_url
); ?>" updated="

<?php
                            if
( '0000-00-00 00:00:00' !==
$bookmark->link_updated ) {

  echo $bookmark-
>link_updated;
                            }
                            ?>
" />
        <?php
    endforeach; // $bookmarks
    ?>
</outline>
    <?php
endforeach; // $cats
?>
</body>
</opml>

<?php
/**
 * Bootstrap file for setting
the ABSPATH constant
 * and loading the wp-
config.php file. The wp-
config.php
 * file will then load the wp-
settings.php file, which
 * will then set up the
WordPress environment.
 *
 * If the wp-config.php file
is not found then an error
 * will be displayed asking
the visitor to set up the
 * wp-config.php file.
 *
 * Will also search for wp-
config.php in WordPress'
parent
 * directory to allow the
WordPress directory to remain
 * untouched.
 *
 * @package WordPress
 */

/** Define ABSPATH as this
file's directory */
if ( ! defined( 'ABSPATH' ) )
{
    define( 'ABSPATH', __DIR__
. '/' );
}

/*
 * The error_reporting()
function can be disabled in
php.ini. On systems where that
is the case,
 * it's best to add a dummy
function to the wp-config.php
file, but as this call to the
function
 * is run prior to wp-
config.php loading, it is
wrapped in a function_exists()
check.
 */
if ( function_exists(
'error_reporting' ) ) {
    /*
     * Initialize error
reporting to a known set of
levels.
     *
     * This will be adapted in
wp_debug_mode() located in wp-
```

```
includes/load.php based on
WP_DEBUG.
     * @see
https://www.php.net/manual/en/
errorfunc.constants.php List
of known error levels.
     */
    error_reporting(
E_CORE_ERROR | E_CORE_WARNING
| E_COMPILE_ERROR | E_ERROR |
E_WARNING | E_PARSE |
E_USER_ERROR | E_USER_WARNING
| E_RECOVERABLE_ERROR );
}

/*
 * If wp-config.php exists in
the WordPress root, or if it
exists in the root and wp-
settings.php
 * doesn't, load wp-
config.php. The secondary
check for wp-settings.php has
the added benefit
 * of avoiding cases where the
current directory is a nested
installation, e.g. / is
WordPress(a)
 * and /blog/ is WordPress(b).
 *
 * If neither set of
conditions is true, initiate
loading the setup process.
 */
if ( file_exists( ABSPATH .
'wp-config.php' ) ) {

    /** The config file
resides in ABSPATH */
    require_once ABSPATH .
'wp-config.php';

} elseif ( @file_exists(
dirname( ABSPATH ) . '/wp-
config.php' ) && !
@file_exists( dirname( ABSPATH
) . '/wp-settings.php' ) ) {

    /** The config file
resides one level above
ABSPATH but is not part of
another installation */
    require_once dirname(
ABSPATH ) . '/wp-config.php';

} else {

    // A config file doesn't
exist.

    define( 'WPINC', 'wp-
includes' );
    require_once ABSPATH .
WPINC . '/version.php';
    require_once ABSPATH .
WPINC . '/compat.php';
    require_once ABSPATH .
WPINC . '/load.php';

    // Check for the required
PHP version and for the MySQL
extension or a database drop-
in.

wp_check_php_mysql_versions();

    // Standardize $_SERVER
variables across setups.
    wp_fix_server_vars();

    define( 'WP_CONTENT_DIR',
ABSPATH . 'wp-content' );
    require_once ABSPATH .
WPINC . '/functions.php';

    $path = wp_guess_url() .
'/wp-admin/setup-config.php';

    // Redirect to setup-
config.php.
    if ( ! str_contains(
$_SERVER['REQUEST_URI'],
'setup-config' ) ) {
        header( 'Location: ' .
$path );
        exit;
    }


wp_load_translations_early();

    // Die with an error
message.
    $die = '<p>' . sprintf(
```

```
        /* translators: %s:
wp-config.php */
        __( "There doesn't
seem to be a %s file. It is
needed before the installation
can continue." ),
        '<code>wp-
config.php</code>'
    ) . '</p>';
    $die .= '<p>' . sprintf(
        /* translators: 1:
Documentation URL, 2: wp-
config.php */
        __( 'Need more help?
<a href="%1$s">Read the
support article on %2$s</a>.'
),
        __(
'https://developer.wordpress.o
rg/advanced-
administration/wordpress/wp-
config/' ),
        '<code>wp-
config.php</code>'
    ) . '</p>';
    $die .= '<p>' . sprintf(
        /* translators: %s:
wp-config.php */
        __( "You can create a
%s file through a web
interface, but this doesn't
work for all server setups.
The safest way is to manually
create the file." ),
        '<code>wp-
config.php</code>'
    ) . '</p>';
    $die .= '<p><a href="' .
$path . '" class="button
button-large">' . __( 'Create
a Configuration File' ) .
'</a></p>';

    wp_die( $die, __(
'WordPress &rsaquo; Error' )
);
}

<?php
/**
 * WordPress User Page
 *
 * Handles authentication,
registering, resetting
passwords, forgot password,
 * and other user handling.
 *
 * @package WordPress
 */

/** Make sure that the
WordPress bootstrap has run
before continuing. */
require __DIR__ . '/wp-
load.php';

// Redirect to HTTPS login if
forced to use SSL.
if ( force_ssl_admin() &&
is_ssl() ) {
    if ( str_starts_with(
$_SERVER['REQUEST_URI'],
'http' ) ) {
        wp_safe_redirect(
set_url_scheme(
$_SERVER['REQUEST_URI'],
'https' ) );
        exit;
    } else {
        wp_safe_redirect(
'https://' .
$_SERVER['HTTP_HOST'] .
$_SERVER['REQUEST_URI'] );
        exit;
    }
}

/**
 * Outputs the login page
header.
 *
 * @since 2.1.0
 *
 * @global string      $error
        Login error message
set by deprecated pluggable
wp_login() function
 *
       or plugins replacing
it.
 * @global bool|string
$interim_login Whether interim
login modal is being
displayed. String 'success'
```

```
 *
       upon successful login.
 * @global string      $action
       The action that brought
the visitor to the login page.
 *
 * @param string|null   $title
   Optional. WordPress login
page title to display in the
`<title>` element.
 *
   Defaults to 'Log In'.
 * @param string
 $message  Optional. Message
to display in header. Default
empty.
 * @param WP_Error|null
$wp_error Optional. The error
to pass. Defaults to a
WP_Error instance.
 */
function login_header( $title
= null, $message = '',
$wp_error = null ) {
    global $error,
$interim_login, $action;

    if ( null === $title ) {
        $title = __( 'Log In'
);
    }

    // Don't index any of
these forms.
    add_filter( 'wp_robots',
'wp_robots_sensitive_page' );
    add_action( 'login_head',
'wp_strict_cross_origin_referr
er' );

    add_action( 'login_head',
'wp_login_viewport_meta' );

    if ( ! is_wp_error(
$wp_error ) ) {
        $wp_error = new
WP_Error();
    }

    // Shake it!
    $shake_error_codes =
array( 'empty_password',
'empty_email',
'invalid_email',
'invalidcombo',
'empty_username',
'invalid_username',
'incorrect_password',
'retrieve_password_email_failu
re' );
    /**
     * Filters the error codes
array for shaking the login
form.
     *
     * @since 3.0.0
     *
     * @param string[]
$shake_error_codes Error codes
that shake the login form.
     */
    $shake_error_codes =
apply_filters(
'shake_error_codes',
$shake_error_codes );

    if ( $shake_error_codes &&
$wp_error->has_errors() &&
in_array( $wp_error-
>get_error_code(),
$shake_error_codes, true ) ) {
        add_action(
'login_footer', 'wp_shake_js',
12 );
    }

    $login_title =
get_bloginfo( 'name',
'display' );

    /* translators: Login
screen title. 1: Login screen
name, 2: Network or site name.
*/
    $login_title = sprintf(
__( '%1$s &lsaquo; %2$s
&#8212; WordPress' ), $title,
$login_title );

    if ( wp_is_recovery_mode()
) {
        /* translators: %s:
Login screen title. */
```

```
        $login_title =
sprintf( __( 'Recovery Mode
&#8212; %s' ), $login_title );
    }

    /**
     * Filters the title tag
content for login page.
     *
     * @since 4.9.0
     *
     * @param string
$login_title The page title,
with extra context added.
     * @param string $title
    The original page title.
     */
    $login_title =
apply_filters( 'login_title',
$login_title, $title );

    ?><!DOCTYPE html>
    <html <?php
language_attributes(); ?>>
    <head>
    <meta http-equiv="Content-
Type" content="<?php bloginfo(
'html_type' ); ?>;
charset=<?php bloginfo(
'charset' ); ?>" />
    <title><?php echo
$login_title; ?></title>
    <?php

    wp_enqueue_style( 'login'
);

    /*
     * Remove all stored post
data on logging out.
     * This could be added by
add_action('login_head'...)
like wp_shake_js(),
     * but maybe better if
it's not removable by plugins.
     */
    if ( 'loggedout' ===
$wp_error->get_error_code() )
{
        ob_start();
        ?>
<script>if("sessionStorage" in
window){try{for(var key in
sessionStorage){if(key.indexOf
("wp-autosave-")!=-
1){sessionStorage.removeItem(k
ey)}}}catch(e){}};</script>
        <?php

wp_print_inline_script_tag(
wp_remove_surrounding_empty_sc
ript_tags( ob_get_clean() ) );
    }

    /**
     * Enqueues scripts and
styles for the login page.
     *
     * @since 3.1.0
     */
    do_action(
'login_enqueue_scripts' );

    /**
     * Fires in the login page
header after scripts are
enqueued.
     *
     * @since 2.1.0
     */
    do_action( 'login_head' );

    $login_header_url = __(
'https://wordpress.org/' );

    /**
     * Filters link URL of the
header logo above login form.
     *
     * @since 2.1.0
     *
     * @param string
$login_header_url Login header
logo URL.
     */
    $login_header_url =
apply_filters(
'login_headerurl',
$login_header_url );

    $login_header_title = '';
```

```
    /**
     * Filters the title
attribute of the header logo
above login form.
     *
     * @since 2.1.0
     * @deprecated 5.2.0 Use
{@see 'login_headertext'}
instead.
     *
     * @param string
$login_header_title Login
header logo title attribute.
     */
    $login_header_title =
apply_filters_deprecated(
        'login_headertitle',
        array(
$login_header_title ),
        '5.2.0',
        'login_headertext',
        __( 'Usage of the
title attribute on the login
logo is not recommended for
accessibility reasons. Use the
link text instead.' )
    );

    $login_header_text =
empty( $login_header_title ) ?
__( 'Powered by WordPress' ) :
$login_header_title;

    /**
     * Filters the link text
of the header logo above the
login form.
     *
     * @since 5.2.0
     *
     * @param string
$login_header_text The login
header logo link text.
     */
    $login_header_text =
apply_filters(
'login_headertext',
$login_header_text );

    $classes = array( 'login-
action-' . $action, 'wp-core-
ui' );

    if ( is_rtl() ) {
        $classes[] = 'rtl';
    }

    if ( $interim_login ) {
        $classes[] = 'interim-
login';

        ?>
        <style
type="text/css">html{backgroun
d-color: transparent;}</style>
        <?php

        if ( 'success' ===
$interim_login ) {
            $classes[] =
'interim-login-success';
        }
    }

    $classes[] = ' locale-' .
sanitize_html_class(
strtolower( str_replace( '_',
'-', get_locale() ) ) );

    /**
     * Filters the login page
body classes.
     *
     * @since 3.5.0
     *
     * @param string[]
$classes An array of body
classes.
     * @param string   $action
 The action that brought the
visitor to the login page.
     */
    $classes = apply_filters(
'login_body_class', $classes,
$action );

    ?>
    </head>
    <body class="login no-js
<?php echo esc_attr( implode(
' ', $classes ) ); ?>">
    <?php

wp_print_inline_script_tag(
```

```
"document.body.className =
document.body.className.replac
e('no-js','js');" );
    ?>

    <?php
    /**
     * Fires in the login page
header after the body tag is
opened.
     *
     * @since 4.6.0
     */
    do_action( 'login_header'
);

    ?>
    <div id="login">
        <h1><a href="<?php
echo esc_url(
$login_header_url ); ?>"><?php
echo $login_header_text;
?></a></h1>
    <?php
    /**
     * Filters the message to
display above the login form.
     *
     * @since 2.1.0
     *
     * @param string $message
Login message text.
     */
    $message = apply_filters(
'login_message', $message );

    if ( ! empty( $message ) )
{
        echo $message . "\n";
    }

    // In case a plugin uses
$error rather than the
$wp_errors object.
    if ( ! empty( $error ) ) {
        $wp_error->add(
'error', $error );
        unset( $error );
    }

    if ( $wp_error-
>has_errors() ) {
        $error_list = array();
        $messages   = '';

        foreach ( $wp_error-
>get_error_codes() as $code )
{
            $severity =
$wp_error->get_error_data(
$code );
            foreach (
$wp_error->get_error_messages(
$code ) as $error_message ) {
                if ( 'message'
=== $severity ) {
                    $messages
.= '<p>' . $error_message .
'</p>';
                } else {

$error_list[] =
$error_message;
                }
            }
        }

        if ( ! empty(
$error_list ) ) {
            $errors = '';

            if ( count(
$error_list ) > 1 ) {
                $errors .=
'<ul class="login-error-
list">';

                foreach (
$error_list as $item ) {
                    $errors .=
'<li>' . $item . '</li>';
                }

                $errors .=
'</ul>';
            } else {
                $errors .=
'<p>' . $error_list[0] .
'</p>';
            }

            /**
```

```
             * Filters the
error messages displayed above
the login form.
             *
             * @since 2.1.0
             *
             * @param string
$errors Login error messages.
             */
            $errors =
apply_filters( 'login_errors',
$errors );

            wp_admin_notice(
                $errors,
                array(
                    'type'
      => 'error',
                    'id'
      => 'login_error',

'paragraph_wrap' => false,
                )
            );
        }

        if ( ! empty(
$messages ) ) {
            /**
             * Filters
instructional messages
displayed above the login
form.
             *
             * @since 2.5.0
             *
             * @param string
$messages Login messages.
             */
            $messages =
apply_filters(
'login_messages', $messages );

            wp_admin_notice(
                $messages,
                array(
                    'type'
          => 'info',
                    'id'
          => 'login-message',
'additional_classes' => array(
'message' ),

'paragraph_wrap'     => false,
                )
            );
        }
    }
} // End of login_header().

/**
 * Outputs the footer for the
login page.
 *
 * @since 3.1.0
 *
 * @global bool|string
$interim_login Whether interim
login modal is being
displayed. String 'success'
 *
       upon successful login.
 *
 * @param string $input_id
Which input to auto-focus.
 */
function login_footer(
$input_id = '' ) {
    global $interim_login;

    // Don't allow interim
logins to navigate away from
the page.
    if ( ! $interim_login ) {
        ?>
        <p id="backtoblog">
            <?php
            $html_link =
sprintf(
                '<a
href="%s">%s</a>',
                esc_url(
home_url( '/' ) ),
                sprintf(
                    /*
translators: %s: Site title.
*/
                    _x(
'&larr; Go to %s', 'site' ),
```

```
get_bloginfo( 'title',
'display' )
                )
            );
            /**
             * Filters the "Go
to site" link displayed in the
login page footer.
             *
             * @since 5.7.0
             *
             * @param string
$link HTML link to the home
URL of the current site.
             */
            echo
apply_filters(
'login_site_html_link',
$html_link );
            ?>
        </p>
        <?php


the_privacy_policy_link( '<div
class="privacy-policy-page-
link">', '</div>' );
    }

    ?>
    </div><?php // End of <div
id="login">. ?>

    <?php
    if (
        ! $interim_login &&
        /**
         * Filters whether to
display the Language selector
on the login screen.
         *
         * @since 5.9.0
         *
         * @param bool
$display Whether to display
the Language selector on the
login screen.
         */
        apply_filters(
'login_display_language_dropdo
wn', true )
    ) {
        $languages =
get_available_languages();

        if ( ! empty(
$languages ) ) {
            ?>
            <div
class="language-switcher">
                <form
id="language-switcher"
method="get">

                    <label
for="language-switcher-
locales">
                        <span
class="dashicons dashicons-
translation" aria-
hidden="true"></span>
                        <span
class="screen-reader-text">

<?php
                            /*
translators: Hidden
accessibility text. */

_e( 'Language' );
                            ?>

</span>
                    </label>

                    <?php
                    $args =
array(
                        'id'
                       =>
'language-switcher-locales',
                        'name'
                       =>
'wp_lang',

'selected'
 => determine_locale(),

'show_available_translations'
=> false,

'explicit_option_en_us'
=> true,
```

```
'languages'
=> $languages,
                    );

                    /**
                     * Filters
default arguments for the
Languages select input on the
login screen.
                     *
                     * The
arguments get passed to the
wp_dropdown_languages()
function.
                     *
                     * @since
5.9.0
                     *
                     * @param
array $args Arguments for the
Languages select input on the
login screen.
                     */

wp_dropdown_languages(
apply_filters(
'login_language_dropdown_args'
, $args ) );
                    ?>

                    <?php if (
$interim_login ) { ?>
                        <input
type="hidden" name="interim-
login" value="1" />
                    <?php } ?>

                    <?php if (
isset( $_GET['redirect_to'] )
&& '' !== $_GET['redirect_to']
) { ?>
                        <input
type="hidden"
name="redirect_to"
value="<?php echo
sanitize_url(
$_GET['redirect_to'] ); ?>" />
                    <?php } ?>

                    <?php if (
isset( $_GET['action'] ) && ''
!== $_GET['action'] ) { ?>
                        <input
type="hidden" name="action"
value="<?php echo esc_attr(
$_GET['action'] ); ?>" />
                    <?php } ?>

                        <input
type="submit" class="button"
value="<?php esc_attr_e(
'Change' ); ?>">

                    </form>
                </div>
        <?php } ?>
    <?php } ?>
    <?php

    if ( ! empty( $input_id )
) {
        ob_start();
        ?>
        <script>

try{document.getElementById('<
?php echo $input_id;
?>').focus();}catch(e){}
        if(typeof
wpOnload=='function')wpOnload
();
        </script>
        <?php

wp_print_inline_script_tag(
wp_remove_surrounding_empty_sc
ript_tags( ob_get_clean() ) );
    }

    /**
     * Fires in the login page
footer.
     *
     * @since 3.1.0
     */
    do_action( 'login_footer'
);

    ?>
    </body>
    </html>
```

```
    <?php
}

/**
 * Outputs the JavaScript to
handle the form shaking on the
login page.
 *
 * @since 3.0.0
 */
function wp_shake_js() {

wp_print_inline_script_tag(
"document.querySelector('form'
).classList.add('shake');" );
}

/**
 * Outputs the viewport meta
tag for the login page.
 *
 * @since 3.7.0
 */
function
wp_login_viewport_meta() {
    ?>
    <meta name="viewport"
content="width=device-width"
/>
    <?php
}

/*
 * Main part.
 *
 * Check the request and
redirect or display a form
based on the current action.
 */

$action = isset(
$_REQUEST['action'] ) ?
$_REQUEST['action'] : 'login';
$errors = new WP_Error();

if ( isset( $_GET['key'] ) ) {
    $action = 'resetpass';
}

if ( isset(
$_GET['checkemail'] ) ) {
    $action = 'checkemail';
}

$default_actions = array(
    'confirm_admin_email',
    'postpass',
    'logout',
    'lostpassword',
    'retrievepassword',
    'resetpass',
    'rp',
    'register',
    'checkemail',
    'confirmaction',
    'login',

WP_Recovery_Mode_Link_Service:
:LOGIN_ACTION_ENTERED,
);

// Validate action so as to
default to the login screen.
if ( ! in_array( $action,
$default_actions, true ) &&
false === has_filter(
'login_form_' . $action ) ) {
    $action = 'login';
}

nocache_headers();

header( 'Content-Type: ' .
get_bloginfo( 'html_type' ) .
'; charset=' . get_bloginfo(
'charset' ) );

if ( defined( 'RELOCATE' ) &&
RELOCATE ) { // Move flag is
set.
    if ( isset(
$_SERVER['PATH_INFO'] ) && (
$_SERVER['PATH_INFO'] !==
$_SERVER['PHP_SELF'] ) ) {
        $_SERVER['PHP_SELF'] =
str_replace(
$_SERVER['PATH_INFO'], '',
$_SERVER['PHP_SELF'] );
    }

    $url = dirname(
set_url_scheme( 'http://' .
$_SERVER['HTTP_HOST'] .
$_SERVER['PHP_SELF'] ) );
```

```
    if ( get_option( 'siteurl'
) !== $url ) {
        update_option(
'siteurl', $url );
    }
}

// Set a cookie now to see if
they are supported by the
browser.
$secure = ( 'https' ===
parse_url( wp_login_url(),
PHP_URL_SCHEME ) );
setcookie( TEST_COOKIE, 'WP
Cookie check', 0, COOKIEPATH,
COOKIE_DOMAIN, $secure );

if ( SITECOOKIEPATH !==
COOKIEPATH ) {
    setcookie( TEST_COOKIE,
'WP Cookie check', 0,
SITECOOKIEPATH, COOKIE_DOMAIN,
$secure );
}

if ( isset( $_GET['wp_lang'] )
) {
    setcookie( 'wp_lang',
sanitize_text_field(
$_GET['wp_lang'] ), 0,
COOKIEPATH, COOKIE_DOMAIN,
$secure );
}

/**
 * Fires when the login form
is initialized.
 *
 * @since 3.2.0
 */
do_action( 'login_init' );

/**
 * Fires before a specified
login form action.
 *
 * The dynamic portion of the
hook name, `$action`, refers
to the action
 * that brought the visitor to
the login form.
 *
 * Possible hook names
include:
 *
 *  - `login_form_checkemail`
 *  -
`login_form_confirm_admin_emai
l`
 *  -
`login_form_confirmaction`
 *  -
`login_form_entered_recovery_m
ode`
 *  - `login_form_login`
 *  - `login_form_logout`
 *  -
`login_form_lostpassword`
 *  - `login_form_postpass`
 *  - `login_form_register`
 *  - `login_form_resetpass`
 *  -
`login_form_retrievepassword`
 *  - `login_form_rp`
 *
 * @since 2.8.0
 */
do_action(
"login_form_{$action}" );

$http_post     = ( 'POST' ===
$_SERVER['REQUEST_METHOD'] );
$interim_login = isset(
$_REQUEST['interim-login'] );

/**
 * Filters the separator used
between login form navigation
links.
 *
 * @since 4.9.0
 *
 * @param string
$login_link_separator The
separator used between login
form navigation links.
 */
$login_link_separator =
apply_filters(
'login_link_separator', ' | '
);

switch ( $action ) {
```

```
    case
'confirm_admin_email':
        /*
         * Note that
`is_user_logged_in()` will
return false immediately after
logging in
         * as the current user
is not set, see wp-
includes/pluggable.php.
         * However this action
runs on a redirect after
logging in.
         */
        if ( !
is_user_logged_in() ) {
            wp_safe_redirect(
wp_login_url() );
            exit;
        }

        if ( ! empty(
$_REQUEST['redirect_to'] ) ) {
            $redirect_to =
$_REQUEST['redirect_to'];
        } else {
            $redirect_to =
admin_url();
        }

        if ( current_user_can(
'manage_options' ) ) {
            $admin_email =
get_option( 'admin_email' );
        } else {
            wp_safe_redirect(
$redirect_to );
            exit;
        }

        /**
         * Filters the
interval for dismissing the
admin email confirmation
screen.
         *
         * If `0` (zero) is
returned, the "Remind me
later" link will not be
displayed.
         *
         * @since 5.3.1
         *
         * @param int
$interval Interval time (in
seconds). Default is 3 days.
         */
        $remind_interval =
(int) apply_filters(
'admin_email_remind_interval',
3 * DAY_IN_SECONDS );

        if ( ! empty(
$_GET['remind_me_later'] ) ) {
            if ( !
wp_verify_nonce(
$_GET['remind_me_later'],
'remind_me_later_nonce' ) ) {

wp_safe_redirect(
wp_login_url() );
                exit;
            }

            if (
$remind_interval > 0 ) {
                update_option(
'admin_email_lifespan', time()
+ $remind_interval );
            }

            $redirect_to =
add_query_arg(
'admin_email_remind_later', 1,
$redirect_to );
            wp_safe_redirect(
$redirect_to );
            exit;
        }

        if ( ! empty(
$_POST['correct-admin-email']
) ) {
            if ( !
check_admin_referer(
'confirm_admin_email',
'confirm_admin_email_nonce' )
) {

wp_safe_redirect(
wp_login_url() );
                exit;
            }
```

```
            /**
             * Filters the
interval for redirecting the
user to the admin email
confirmation screen.
             *
             * If `0` (zero)
is returned, the user will not
be redirected.
             *
             * @since 5.3.0
             *
             * @param int
$interval Interval time (in
seconds). Default is 6 months.
             */

$admin_email_check_interval =
(int) apply_filters(
'admin_email_check_interval',
6 * MONTH_IN_SECONDS );

            if (
$admin_email_check_interval >
0 ) {
                update_option(
'admin_email_lifespan', time()
+ $admin_email_check_interval
);
            }

            wp_safe_redirect(
$redirect_to );
            exit;
        }

        login_header( __(
'Confirm your administration
email' ), '', $errors );

        /**
         * Fires before the
admin email confirm form.
         *
         * @since 5.3.0
         *
         * @param WP_Error
$errors A `WP_Error` object
containing any errors
generated by using invalid
         *
    credentials. Note that the
error object may not contain
any errors.
         */
        do_action(
'admin_email_confirm', $errors
);

        ?>

        <form class="admin-
email-confirm-form"
name="admin-email-confirm-
form" action="<?php echo
esc_url( site_url( 'wp-
login.php?action=confirm_admin
_email', 'login_post' ) ); ?>"
method="post">
            <?php
            /**
             * Fires inside
the admin-email-confirm-form
form tags, before the hidden
fields.
             *
             * @since 5.3.0
             */
            do_action(
'admin_email_confirm_form' );

            wp_nonce_field(
'confirm_admin_email',
'confirm_admin_email_nonce' );

            ?>
            <input
type="hidden"
name="redirect_to"
value="<?php echo esc_attr(
$redirect_to ); ?>" />

            <h1 class="admin-
email__heading">
                <?php _e(
'Administration email
verification' ); ?>
            </h1>
            <p class="admin-
email__details">
                <?php _e(
'Please verify that the
```

```
<strong>administration
email</strong> for this
website is still correct.' );
?>
                <?php

                /*
translators: URL to the
WordPress help section about
admin email. */

$admin_email_help_url = __(
'https://wordpress.org/documen
tation/article/settings-
general-screen/#email-address'
);


$accessibility_text = sprintf(
                    '<span
class="screen-reader-text">
%s</span>',
                    /*
translators: Hidden
accessibility text. */
                    __(
'(opens in a new tab)' )
                );

                printf(
                    '<a
href="%s" rel="noopener"
target="_blank">%s%s</a>',
                    esc_url(
$admin_email_help_url ),
                    __( 'Why
is this important?' ),

$accessibility_text
                );

                ?>
            </p>
            <p class="admin-
email__details">
                <?php

                printf(
                    /*
translators: %s: Admin email
address. */
                    __(
'Current administration email:
%s' ),
                    '<strong>'
. esc_html( $admin_email ) .
'</strong>'
                );

                ?>
            </p>
            <p class="admin-
email__details">
                <?php _e(
'This email may be different
from your personal email
address.' ); ?>
            </p>

            <div class="admin-
email__actions">
                <div
class="admin-email__actions-
primary">
                    <?php


$change_link = admin_url(
'options-general.php' );

$change_link = add_query_arg(
'highlight',
'confirm_admin_email',
$change_link );

                    ?>
                    <a
class="button button-large"
href="<?php echo esc_url(
$change_link ); ?>"><?php _e(
'Update' ); ?></a>
                    <input
type="submit" name="correct-
admin-email" id="correct-
admin-email" class="button
button-primary button-large"
value="<?php esc_attr_e( 'The
email is correct' ); ?>" />
                </div>
                <?php if (
$remind_interval > 0 ) : ?>
```

```
                    <div
class="admin-email__actions-
secondary">
                        <?php


$remind_me_link =
wp_login_url( $redirect_to );

$remind_me_link =
add_query_arg(

array(

  'action'          =>
'confirm_admin_email',

  'remind_me_later' =>
wp_create_nonce(
'remind_me_later_nonce' ),
                            ),

$remind_me_link
                        );

                        ?>
                        <a
href="<?php echo esc_url(
$remind_me_link ); ?>"><?php
_e( 'Remind me later' );
?></a>
                    </div>
                <?php endif;
?>
            </div>
        </form>

        <?php

        login_footer();
        break;

    case 'postpass':
        if ( ! isset(
$_POST['post_password'] ) || !
is_string(
$_POST['post_password'] ) ) {
            wp_safe_redirect(
wp_get_referer() );
            exit;
        }

        require_once ABSPATH .
WPINC . '/class-phpass.php';
        $hasher = new
PasswordHash( 8, true );

        /**
         * Filters the life
span of the post password
cookie.
         *
         * By default, the
cookie expires 10 days from
creation. To turn this
         * into a session
cookie, return 0.
         *
         * @since 3.7.0
         *
         * @param int $expires
The expiry time, as passed to
setcookie().
         */
        $expire  =
apply_filters(
'post_password_expires',
time() + 10 * DAY_IN_SECONDS
);
        $referer =
wp_get_referer();

        if ( $referer ) {
            $secure = (
'https' === parse_url(
$referer, PHP_URL_SCHEME ) );
        } else {
            $secure = false;
        }

        setcookie( 'wp-
postpass_' . COOKIEHASH,
$hasher->HashPassword(
wp_unslash(
$_POST['post_password'] ) ),
$expire, COOKIEPATH,
COOKIE_DOMAIN, $secure );

        wp_safe_redirect(
wp_get_referer() );
        exit;

    case 'logout':
```

```
        check_admin_referer(
'log-out' );

        $user =
wp_get_current_user();

        wp_logout();

        if ( ! empty(
$_REQUEST['redirect_to'] ) &&
is_string(
$_REQUEST['redirect_to'] ) ) {
            $redirect_to
    =
$_REQUEST['redirect_to'];

$requested_redirect_to =
$redirect_to;
        } else {
            $redirect_to =
add_query_arg(
                array(

'loggedout' => 'true',
                    'wp_lang'
  => get_user_locale( $user ),
                ),
                wp_login_url()
            );


$requested_redirect_to = '';
        }

        /**
         * Filters the log out
redirect URL.
         *
         * @since 4.2.0
         *
         * @param string
 $redirect_to           The
redirect destination URL.
         * @param string
 $requested_redirect_to The
requested redirect destination
URL passed as a parameter.
         * @param WP_User
$user                  The
WP_User object for the user
that's logging out.
         */
        $redirect_to =
apply_filters(
'logout_redirect',
$redirect_to,
$requested_redirect_to, $user
);

        wp_safe_redirect(
$redirect_to );
        exit;

    case 'lostpassword':
    case 'retrievepassword':
        if ( $http_post ) {
            $errors =
retrieve_password();

            if ( !
is_wp_error( $errors ) ) {
                $redirect_to =
! empty(
$_REQUEST['redirect_to'] ) ?
$_REQUEST['redirect_to'] :
'wp-
login.php?checkemail=confirm';

wp_safe_redirect( $redirect_to
);
                exit;
            }
        }

        if ( isset(
$_GET['error'] ) ) {
            if ( 'invalidkey'
=== $_GET['error'] ) {
                $errors->add(
'invalidkey', __(
'<strong>Error:</strong> Your
password reset link appears to
be invalid. Please request a
new link below.' ) );
            } elseif (
'expiredkey' ===
$_GET['error'] ) {
                $errors->add(
'expiredkey', __(
'<strong>Error:</strong> Your
password reset link has
expired. Please request a new
link below.' ) );
            }
```

```
        }

        $lostpassword_redirect
= ! empty(
$_REQUEST['redirect_to'] ) ?
$_REQUEST['redirect_to'] : '';
        /**
         * Filters the URL
redirected to after submitting
the
lostpassword/retrievepassword
form.
         *
         * @since 3.0.0
         *
         * @param string
$lostpassword_redirect The
redirect destination URL.
         */
        $redirect_to =
apply_filters(
'lostpassword_redirect',
$lostpassword_redirect );

        /**
         * Fires before the
lost password form.
         *
         * @since 1.5.1
         * @since 5.1.0 Added
the `$errors` parameter.
         *
         * @param WP_Error
$errors A `WP_Error` object
containing any errors
generated by using invalid
         *
    credentials. Note that the
error object may not contain
any errors.
         */
        do_action(
'lost_password', $errors );

        login_header(
            __( 'Lost
Password' ),

wp_get_admin_notice(
                __( 'Please
enter your username or email
address. You will receive an
email message with
instructions on how to reset
your password.' ),
                array(
                    'type'
          => 'info',

'additional_classes' => array(
'message' ),
                )
            ),
            $errors
        );

        $user_login = '';

        if ( isset(
$_POST['user_login'] ) &&
is_string(
$_POST['user_login'] ) ) {
            $user_login =
wp_unslash(
$_POST['user_login'] );
        }

        ?>

        <form
name="lostpasswordform"
id="lostpasswordform"
action="<?php echo esc_url(
network_site_url( 'wp-
login.php?action=lostpassword'
, 'login_post' ) ); ?>"
method="post">
            <p>
                <label
for="user_login"><?php _e(
'Username or Email Address' );
?></label>
                <input
type="text" name="user_login"
id="user_login" class="input"
value="<?php echo esc_attr(
$user_login ); ?>" size="20"
autocapitalize="off"
autocomplete="username"
required="required" />
            </p>
            <?php

            /**
```

```
             * Fires inside
the lostpassword form tags,
before the hidden fields.
             *
             * @since 2.1.0
             */
            do_action(
'lostpassword_form' );

            ?>
            <input
type="hidden"
name="redirect_to"
value="<?php echo esc_attr(
$redirect_to ); ?>" />
            <p class="submit">
                <input
type="submit" name="wp-submit"
id="wp-submit" class="button
button-primary button-large"
value="<?php esc_attr_e( 'Get
New Password' ); ?>" />
            </p>
        </form>

        <p id="nav">
            <a class="wp-
login-log-in" href="<?php echo
esc_url( wp_login_url() );
?>"><?php _e( 'Log in' );
?></a>
            <?php

            if ( get_option(
'users_can_register' ) ) {

$registration_url = sprintf(
'<a class="wp-login-register"
href="%s">%s</a>', esc_url(
wp_registration_url() ), __(
'Register' ) );

                echo esc_html(
$login_link_separator );

                /** This
filter is documented in wp-
includes/general-template.php
*/
                echo
apply_filters( 'register',
$registration_url );
                }

                ?>
            </p>
            <?php

        login_footer(
'user_login' );
        break;

    case 'resetpass':
    case 'rp':
        list( $rp_path ) =
explode( '?', wp_unslash(
$_SERVER['REQUEST_URI'] ) );
        $rp_cookie       =
'wp-resetpass-' . COOKIEHASH;

        if ( isset(
$_GET['key'] ) && isset(
$_GET['login'] ) ) {
            $value = sprintf(
'%s:%s', wp_unslash(
$_GET['login'] ), wp_unslash(
$_GET['key'] ) );
            setcookie(
$rp_cookie, $value, 0,
$rp_path, COOKIE_DOMAIN,
is_ssl(), true );

            wp_safe_redirect(
remove_query_arg( array(
'key', 'login' ) ) );
            exit;
        }

        if ( isset( $_COOKIE[
$rp_cookie ] ) && 0 < strpos(
$_COOKIE[ $rp_cookie ], ':' )
) {
            list( $rp_login,
$rp_key ) = explode( ':',
wp_unslash( $_COOKIE[
$rp_cookie ] ), 2 );

            $user =
check_password_reset_key(
$rp_key, $rp_login );

            if ( isset(
$_POST['pass1'] ) && !
```

```
hash_equals( $rp_key,
$_POST['rp_key'] ) ) {
                $user = false;
            }
        } else {
            $user = false;
        }

        if ( ! $user ||
is_wp_error( $user ) ) {
            setcookie(
$rp_cookie, ' ', time() -
YEAR_IN_SECONDS, $rp_path,
COOKIE_DOMAIN, is_ssl(), true
);

            if ( $user &&
$user->get_error_code() ===
'expired_key' ) {
                wp_redirect(
site_url( 'wp-
login.php?action=lostpassword&
error=expiredkey' ) );
            } else {
                wp_redirect(
site_url( 'wp-
login.php?action=lostpassword&
error=invalidkey' ) );
            }

            exit;
        }

        $errors = new
WP_Error();

        // Check if password
is one or all empty spaces.
        if ( ! empty(
$_POST['pass1'] ) ) {
            $_POST['pass1'] =
trim( $_POST['pass1'] );

            if ( empty(
$_POST['pass1'] ) ) {
                $errors->add(
'password_reset_empty_space',
__( 'The password cannot be a
space or all spaces.' ) );
            }
        }

        // Check if password
fields do not match.
        if ( ! empty(
$_POST['pass1'] ) && trim(
$_POST['pass2'] ) !==
$_POST['pass1'] ) {
            $errors->add(
'password_reset_mismatch', __(
'<strong>Error:</strong> The
passwords do not match.' ) );
        }

        /**
         * Fires before the
password reset procedure is
validated.
         *
         * @since 3.5.0
         *
         * @param WP_Error
    $errors WP Error object.
         * @param
WP_User|WP_Error $user
WP_User object if the login
and reset key match. WP_Error
object otherwise.
         */
        do_action(
'validate_password_reset',
$errors, $user );

        if ( ( ! $errors-
>has_errors() ) && isset(
$_POST['pass1'] ) && ! empty(
$_POST['pass1'] ) ) {
            reset_password(
$user, $_POST['pass1'] );
            setcookie(
$rp_cookie, ' ', time() -
YEAR_IN_SECONDS, $rp_path,
COOKIE_DOMAIN, is_ssl(), true
);
            login_header(
                __( 'Password
Reset' ),

wp_get_admin_notice(
                    __( 'Your
password has been reset.' ) .
' <a href="' . esc_url(
wp_login_url() ) . '">' . __(
'Log in' ) . '</a>',
```

```
                    array(
                        'type'
              => 'info',

'additional_classes' => array(
'message', 'reset-pass' ),
                    )
                )
            );
            login_footer();
            exit;
        }

        wp_enqueue_script(
'utils' );
        wp_enqueue_script(
'user-profile' );

        login_header(
            __( 'Reset
Password' ),

wp_get_admin_notice(
                __( 'Enter
your new password below or
generate one.' ),
                array(
                    'type'
          => 'info',

'additional_classes' => array(
'message', 'reset-pass' ),
                )
            ),
            $errors
        );

        ?>
        <form
name="resetpassform"
id="resetpassform"
action="<?php echo esc_url(
network_site_url( 'wp-
login.php?action=resetpass',
'login_post' ) ); ?>"
method="post"
autocomplete="off">
            <input
type="hidden" id="user_login"
value="<?php echo esc_attr(
$rp_login ); ?>"
autocomplete="off" />

            <div class="user-
pass1-wrap">
                <p>
                    <label
for="pass1"><?php _e( 'New
password' ); ?></label>
                </p>

                <div
class="wp-pwd">
                    <input
type="password" name="pass1"
id="pass1" class="input
password-input" size="24"
value="" autocomplete="new-
password" spellcheck="false"
data-reveal="1" data-pw="<?php
echo esc_attr(
wp_generate_password( 16 ) );
?>" aria-describedby="pass-
strength-result" />

                    <button
type="button" class="button
button-secondary wp-hide-pw
hide-if-no-js" data-toggle="0"
aria-label="<?php esc_attr_e(
'Hide password' ); ?>">
                        <span
class="dashicons dashicons-
hidden" aria-
hidden="true"></span>
                    </button>
                    <div
id="pass-strength-result"
class="hide-if-no-js" aria-
live="polite"><?php _e(
'Strength indicator' );
?></div>
                </div>
                <div
class="pw-weak">
                    <input
type="checkbox" name="pw_weak"
id="pw-weak" class="pw-
checkbox" />
                    <label
for="pw-weak"><?php _e(
'Confirm use of weak password'
); ?></label>
                </div>
```

```
            </div>

            <p class="user-
pass2-wrap">
                <label
for="pass2"><?php _e( 'Confirm
new password' ); ?></label>
                <input
type="password" name="pass2"
id="pass2" class="input"
size="20" value=""
autocomplete="new-password"
spellcheck="false" />
            </p>

            <p
class="description indicator-
hint"><?php echo
wp_get_password_hint(); ?></p>

            <?php

            /**
             * Fires following
the 'Strength indicator' meter
in the user password reset
form.
             *
             * @since 3.9.0
             *
             * @param WP_User
$user User object of the user
whose password is being reset.
             */
            do_action(
'resetpass_form', $user );

            ?>
            <input
type="hidden" name="rp_key"
value="<?php echo esc_attr(
$rp_key ); ?>" />
            <p class="submit
reset-pass-submit">
                <button
type="button" class="button
wp-generate-pw hide-if-no-js
skip-aria-expanded"><?php _e(
'Generate Password' );
?></button>
                <input
type="submit" name="wp-submit"
id="wp-submit" class="button
button-primary button-large"
value="<?php esc_attr_e( 'Save
Password' ); ?>" />
            </p>
        </form>

        <p id="nav">
            <a class="wp-
login-log-in" href="<?php echo
esc_url( wp_login_url() );
?>"><?php _e( 'Log in' );
?></a>
            <?php

            if ( get_option(
'users_can_register' ) ) {

$registration_url = sprintf(
'<a class="wp-login-register"
href="%s">%s</a>', esc_url(
wp_registration_url() ), __(
'Register' ) );

                echo esc_html(
$login_link_separator );

                /** This
filter is documented in wp-
includes/general-template.php
*/
                echo
apply_filters( 'register',
$registration_url );
            }

            ?>
        </p>
        <?php

        login_footer( 'pass1'
);
        break;

    case 'register':
        if ( is_multisite() )
{
            /**
             * Filters the
Multisite sign up URL.
             *
             * @since 3.0.0
```

```
             *
             * @param string
$sign_up_url The sign up URL.
             */
            wp_redirect(
apply_filters(
'wp_signup_location',
network_site_url( 'wp-
signup.php' ) ) );
            exit;
        }

        if ( ! get_option(
'users_can_register' ) ) {
            wp_redirect(
site_url( 'wp-
login.php?registration=disable
d' ) );
            exit;
        }

        $user_login = '';
        $user_email = '';

        if ( $http_post ) {
            if ( isset(
$_POST['user_login'] ) &&
is_string(
$_POST['user_login'] ) ) {
                $user_login =
wp_unslash(
$_POST['user_login'] );
            }

            if ( isset(
$_POST['user_email'] ) &&
is_string(
$_POST['user_email'] ) ) {
                $user_email =
wp_unslash(
$_POST['user_email'] );
            }

            $errors =
register_new_user(
$user_login, $user_email );

            if ( !
is_wp_error( $errors ) ) {
                $redirect_to =
! empty( $_POST['redirect_to']
) ? $_POST['redirect_to'] :
'wp-
login.php?checkemail=registere
d';

wp_safe_redirect( $redirect_to
);
                exit;
            }
        }

        $registration_redirect
= ! empty(
$_REQUEST['redirect_to'] ) ?
$_REQUEST['redirect_to'] : '';

        /**
         * Filters the
registration redirect URL.
         *
         * @since 3.0.0
         * @since 5.9.0 Added
the `$errors` parameter.
         *
         * @param string
$registration_redirect The
redirect destination URL.
         * @param int|WP_Error
$errors                User id
if registration was
successful,
         *

 WP_Error object otherwise.
         */
        $redirect_to =
apply_filters(
'registration_redirect',
$registration_redirect,
$errors );

        login_header(
            __( 'Registration
Form' ),

wp_get_admin_notice(
                __( 'Register
For This Site' ),
                array(
                    'type'
          => 'info',
```

```
'additional_classes' => array(
'message', 'register' ),
                )
            ),
            $errors
        );

        ?>
        <form
name="registerform"
id="registerform"
action="<?php echo esc_url(
site_url( 'wp-
login.php?action=register',
'login_post' ) ); ?>"
method="post"
novalidate="novalidate">
            <p>
                <label
for="user_login"><?php _e(
'Username' ); ?></label>
                <input
type="text" name="user_login"
id="user_login" class="input"
value="<?php echo esc_attr(
wp_unslash( $user_login ) );
?>" size="20"
autocapitalize="off"
autocomplete="username"
required="required" />
            </p>
            <p>
                <label
for="user_email"><?php _e(
'Email' ); ?></label>
                <input
type="email" name="user_email"
id="user_email" class="input"
value="<?php echo esc_attr(
wp_unslash( $user_email ) );
?>" size="25"
autocomplete="email"
required="required" />
            </p>
            <?php

            /**
             * Fires following
the 'Email' field in the user
registration form.
             *
             * @since 2.1.0
             */
            do_action(
'register_form' );

            ?>
            <p
id="reg_passmail">
                <?php _e(
'Registration confirmation
will be emailed to you.' ); ?>
            </p>
            <input
type="hidden"
name="redirect_to"
value="<?php echo esc_attr(
$redirect_to ); ?>" />
            <p class="submit">
                <input
type="submit" name="wp-submit"
id="wp-submit" class="button
button-primary button-large"
value="<?php esc_attr_e(
'Register' ); ?>" />
            </p>
        </form>

        <p id="nav">
            <a class="wp-
login-log-in" href="<?php echo
esc_url( wp_login_url() );
?>"><?php _e( 'Log in' );
?></a>
            <?php

            echo esc_html(
$login_link_separator );

            $html_link =
sprintf( '<a class="wp-login-
lost-password"
href="%s">%s</a>', esc_url(
wp_lostpassword_url() ), __(
'Lost your password?' ) );

            /** This filter is
documented in wp-login.php */
            echo
apply_filters(
'lost_password_html_link',
$html_link );
```

```
            ?>
        </p>
        <?php

        login_footer(
'user_login' );
        break;

    case 'checkemail':
        $redirect_to =
admin_url();
        $errors      = new
WP_Error();

        if ( 'confirm' ===
$_GET['checkemail'] ) {
            $errors->add(
                'confirm',
                sprintf(
                    /*
translators: %s: Link to the
login page. */
                    __( 'Check
your email for the
confirmation link, then visit
the <a href="%s">login
page</a>.' ),

wp_login_url()
                ),
                'message'
            );
        } elseif (
'registered' ===
$_GET['checkemail'] ) {
            $errors->add(
                'registered',
                sprintf(
                    /*
translators: %s: Link to the
login page. */
                    __(
'Registration complete. Please
check your email, then visit
the <a href="%s">login
page</a>.' ),

wp_login_url()
                ),
                'message'
            );
        }

        /** This action is
documented in wp-login.php */
        $errors =
apply_filters(
'wp_login_errors', $errors,
$redirect_to );

        login_header( __(
'Check your email' ), '',
$errors );
        login_footer();
        break;

    case 'confirmaction':
        if ( ! isset(
$_GET['request_id'] ) ) {
            wp_die( __(
'Missing request ID.' ) );
        }

        if ( ! isset(
$_GET['confirm_key'] ) ) {
            wp_die( __(
'Missing confirm key.' ) );
        }

        $request_id = (int)
$_GET['request_id'];
        $key        =
sanitize_text_field(
wp_unslash(
$_GET['confirm_key'] ) );
        $result     =
wp_validate_user_request_key(
$request_id, $key );

        if ( is_wp_error(
$result ) ) {
            wp_die( $result );
        }

        /**
         * Fires an action
hook when the account action
has been confirmed by the
user.
         *
         * Using this you can
assume the user has agreed to
perform the action by
```

```
         * clicking on the
link in the confirmation
email.
         *
         * After firing this
action hook the page will
redirect to wp-login a
callback
         * redirects or exits
first.
         *
         * @since 4.9.6
         *
         * @param int
$request_id Request ID.
         */
        do_action(
'user_request_action_confirmed
', $request_id );

        $message =
_wp_privacy_account_request_co
nfirmed_message( $request_id
);

        login_header( __(
'User action confirmed.' ),
$message );
        login_footer();
        exit;

    case 'login':
    default:
        $secure_cookie   = '';
        $customize_login =
isset( $_REQUEST['customize-
login'] );

        if ( $customize_login
) {
            wp_enqueue_script(
'customize-base' );
        }

        // If the user wants
SSL but the session is not
SSL, force a secure cookie.
        if ( ! empty(
$_POST['log'] ) && !
force_ssl_admin() ) {
            $user_name =
sanitize_user( wp_unslash(
$_POST['log'] ) );
            $user      =
get_user_by( 'login',
$user_name );

            if ( ! $user &&
strpos( $user_name, '@' ) ) {
                $user =
get_user_by( 'email',
$user_name );
            }

            if ( $user ) {
                if (
get_user_option( 'use_ssl',
$user->ID ) ) {

$secure_cookie = true;

force_ssl_admin( true );
                }
            }
        }

        if ( isset(
$_REQUEST['redirect_to'] ) &&
is_string(
$_REQUEST['redirect_to'] ) ) {
            $redirect_to =
$_REQUEST['redirect_to'];
            // Redirect to
HTTPS if user wants SSL.
            if (
$secure_cookie &&
str_contains( $redirect_to,
'wp-admin' ) ) {
                $redirect_to =
preg_replace( '|^http://|',
'https://', $redirect_to );
            }
        } else {
            $redirect_to =
admin_url();
        }

        $reauth = empty(
$_REQUEST['reauth'] ) ? false
: true;
```

```
        $user = wp_signon(
array(), $secure_cookie );

        if ( empty( $_COOKIE[
LOGGED_IN_COOKIE ] ) ) {
            if (
headers_sent() ) {
                $user = new
WP_Error(

'test_cookie',
                    sprintf(
                        /*
translators: 1: Browser cookie
documentation URL, 2: Support
forums URL. */
                        __(
'<strong>Error:</strong>
Cookies are blocked due to
unexpected output. For help,
please see <a href="%1$s">this
documentation</a> or try the
<a href="%2$s">support
forums</a>.' ),
                        __(
'https://developer.wordpress.o
rg/advanced-
administration/wordpress/cooki
es/' ),
                        __(
'https://wordpress.org/support
/forums/' )
                    )
                );
            } elseif ( isset(
$_POST['testcookie'] ) &&
empty( $_COOKIE[ TEST_COOKIE ]
) ) {
                // If cookies
are disabled, the user can't
log in even with a valid
username and password.
                $user = new
WP_Error(

'test_cookie',
                    sprintf(
                        /*
translators: %s: Browser
cookie documentation URL. */
                        __(
'<strong>Error:</strong>
Cookies are blocked or not
supported by your browser. You
must <a href="%s">enable
cookies</a> to use WordPress.'
),
                        __(
'https://developer.wordpress.o
rg/advanced-
administration/wordpress/cooki
es/#enable-cookies-in-your-
browser' )
                    )
                );
            }
        }

        $requested_redirect_to
= isset(
$_REQUEST['redirect_to'] ) &&
is_string(
$_REQUEST['redirect_to'] ) ?
$_REQUEST['redirect_to'] : '';

        /**
         * Filters the login
redirect URL.
         *
         * @since 3.0.0
         *
         * @param string
    $redirect_to           The
redirect destination URL.
         * @param string
    $requested_redirect_to The
requested redirect destination
URL passed as a parameter.
         * @param
WP_User|WP_Error $user
         WP_User object if
login was successful, WP_Error
object otherwise.
         */
        $redirect_to =
apply_filters(
'login_redirect',
$redirect_to,
$requested_redirect_to, $user
);

        if ( ! is_wp_error(
$user ) && ! $reauth ) {
```

```
            if (
$interim_login ) {
                $message
= '<p class="message">' . __(
'You have logged in
successfully.' ) . '</p>';
                $interim_login
= 'success';
                login_header(
'', $message );

                ?>
                </div>
                <?php

                /** This
action is documented in wp-
login.php */
                do_action(
'login_footer' );

                if (
$customize_login ) {

ob_start();
                    ?>

<script>setTimeout(
function(){ new
wp.customize.Messenger({ url:
'<?php echo
wp_customize_url(); ?>',
channel: 'login'
}).send('login') }, 1000
);</script>
                    <?php

wp_print_inline_script_tag(
wp_remove_surrounding_empty_sc
ript_tags( ob_get_clean() ) );
                }

                ?>
                </body></html>
                <?php

                exit;
            }

            // Check if it is
time to add a redirect to the
admin email confirmation
screen.
            if ( $user
instanceof WP_User && $user-
>exists() && $user->has_cap(
'manage_options' ) ) {

$admin_email_lifespan = (int)
get_option(
'admin_email_lifespan' );

                /*
                 * If `0` (or
anything "falsey" as it is
cast to int) is returned, the
user will not be redirected
                 * to the
admin email confirmation
screen.
                 */
                /** This
filter is documented in wp-
login.php */

$admin_email_check_interval =
(int) apply_filters(
'admin_email_check_interval',
6 * MONTH_IN_SECONDS );

                if (
$admin_email_check_interval >
0 && time() >
$admin_email_lifespan ) {

$redirect_to = add_query_arg(
                        array(

'action'  =>
'confirm_admin_email',

'wp_lang' => get_user_locale(
$user ),
                        ),

wp_login_url( $redirect_to )
                    );
                }
            }

            if ( ( empty(
$redirect_to ) || 'wp-admin/'
=== $redirect_to ||
```

```
admin_url() === $redirect_to )
) {
                // If the user
doesn't belong to a blog, send
them to user admin. If the
user can't edit posts, send
them to their profile.
                if (
is_multisite() && !
get_active_blog_for_user(
$user->ID ) && !
is_super_admin( $user->ID ) )
{

$redirect_to =
user_admin_url();
                } elseif (
is_multisite() && ! $user-
>has_cap( 'read' ) ) {

$redirect_to =
get_dashboard_url( $user->ID
);
                } elseif ( !
$user->has_cap( 'edit_posts' )
) {

$redirect_to = $user->has_cap(
'read' ) ? admin_url(
'profile.php' ) : home_url();
                }

                wp_redirect(
$redirect_to );
                exit;
            }

            wp_safe_redirect(
$redirect_to );
            exit;
        }

        $errors = $user;
        // Clear errors if
loggedout is set.
        if ( ! empty(
$_GET['loggedout'] ) ||
$reauth ) {
            $errors = new
WP_Error();
        }

        if ( empty( $_POST )
&& $errors->get_error_codes()
=== array( 'empty_username',
'empty_password' ) ) {
            $errors = new
WP_Error( '', '' );
        }

        if ( $interim_login )
{
            if ( ! $errors-
>has_errors() ) {
                $errors->add(
'expired', __( 'Your session
has expired. Please log in to
continue where you left off.'
), 'message' );
            }
        } else {
            // Some parts of
this script use the main login
form to display a message.
            if ( isset(
$_GET['loggedout'] ) &&
$_GET['loggedout'] ) {
                $errors->add(
'loggedout', __( 'You are now
logged out.' ), 'message' );
            } elseif ( isset(
$_GET['registration'] ) &&
'disabled' ===
$_GET['registration'] ) {
                $errors->add(
'registerdisabled', __(
'<strong>Error:</strong> User
registration is currently not
allowed.' ) );
            } elseif (
str_contains( $redirect_to,
'about.php?updated' ) ) {
                $errors->add(
'updated', __( '<strong>You
have successfully updated
WordPress!</strong> Please log
back in to see what&#8217;s
new.' ), 'message' );
            } elseif (
WP_Recovery_Mode_Link_Service:
:LOGIN_ACTION_ENTERED ===
$action ) {
                $errors->add(
'enter_recovery_mode', __(
```

```
'Recovery Mode Initialized.
Please log in to continue.' ),
'message' );
            } elseif ( isset(
$_GET['redirect_to'] ) &&
is_string(
$_GET['redirect_to'] )
                &&
str_contains(
$_GET['redirect_to'], 'wp-
admin/authorize-
application.php' )
            ) {

$query_component =
wp_parse_url(
$_GET['redirect_to'],
PHP_URL_QUERY );
                $query
  = array();
                if (
$query_component ) {
                    parse_str(
$query_component, $query );
                }

                if ( ! empty(
$query['app_name'] ) ) {
                    /*
translators: 1: Website name,
2: Application name. */
                    $message =
sprintf( 'Please log in to
%1$s to authorize %2$s to
connect to your account.',
get_bloginfo( 'name',
'display' ), '<strong>' .
esc_html( $query['app_name'] )
. '</strong>' );
                } else {
                    /*
translators: %s: Website name.
*/
                    $message =
sprintf( 'Please log in to %s
to proceed with
authorization.', get_bloginfo(
'name', 'display' ) );
                }
                $errors->add(
'authorize_application',
$message, 'message' );
            }
        }

        /**
         * Filters the login
page errors.
         *
         * @since 3.6.0
         *
         * @param WP_Error
$errors      WP Error object.
         * @param string
$redirect_to Redirect
destination URL.
         */
        $errors =
apply_filters(
'wp_login_errors', $errors,
$redirect_to );

        // Clear any stale
cookies.
        if ( $reauth ) {

wp_clear_auth_cookie();
        }

        login_header( __( 'Log
In' ), '', $errors );

        if ( isset(
$_POST['log'] ) ) {
            $user_login = (
'incorrect_password' ===
$errors->get_error_code() ||
'empty_password' === $errors-
>get_error_code() ) ?
esc_attr( wp_unslash(
$_POST['log'] ) ) : '';
        }

        $rememberme = ! empty(
$_POST['rememberme'] );

        $aria_describedby =
'';
        $has_errors       =
$errors->has_errors();
```

```
        if ( $has_errors ) {
            $aria_describedby
= ' aria-
describedby="login_error"';
        }

        if ( $has_errors &&
'message' === $errors-
>get_error_data() ) {
            $aria_describedby
= ' aria-describedby="login-
message"';
        }

        wp_enqueue_script(
'user-profile' );
        ?>

        <form name="loginform"
id="loginform" action="<?php
echo esc_url( site_url( 'wp-
login.php', 'login_post' ) );
?>" method="post">
            <p>
                <label
for="user_login"><?php _e(
'Username or Email Address' );
?></label>
                <input
type="text" name="log"
id="user_login"<?php echo
$aria_describedby; ?>
class="input" value="<?php
echo esc_attr( $user_login );
?>" size="20"
autocapitalize="off"
autocomplete="username"
required="required" />
            </p>

            <div class="user-
pass-wrap">
                <label
for="user_pass"><?php _e(
'Password' ); ?></label>
                <div
class="wp-pwd">
                    <input
type="password" name="pwd"
id="user_pass"<?php echo
$aria_describedby; ?>
class="input password-input"
value="" size="20"
autocomplete="current-
password" spellcheck="false"
required="required" />
                    <button
type="button" class="button
button-secondary wp-hide-pw
hide-if-no-js" data-toggle="0"
aria-label="<?php esc_attr_e(
'Show password' ); ?>">
                        <span
class="dashicons dashicons-
visibility" aria-
hidden="true"></span>
                    </button>
                </div>
            </div>
            <?php

            /**
             * Fires following
the 'Password' field in the
login form.
             *
             * @since 2.1.0
             */
            do_action(
'login_form' );

            ?>
            <p
class="forgetmenot"><input
name="rememberme"
type="checkbox"
id="rememberme"
value="forever" <?php checked(
$rememberme ); ?> /> <label
for="rememberme"><?php
esc_html_e( 'Remember Me' );
?></label></p>
            <p class="submit">
                <input
type="submit" name="wp-submit"
id="wp-submit" class="button
button-primary button-large"
value="<?php esc_attr_e( 'Log
In' ); ?>" />
                <?php

                if (
$interim_login ) {
                    ?>
```

```
                    <input
type="hidden" name="interim-
login" value="1" />
                    <?php
                } else {
                    ?>
                    <input
type="hidden"
name="redirect_to"
value="<?php echo esc_attr(
$redirect_to ); ?>" />
                    <?php
                }

                if (
$customize_login ) {
                    ?>
                    <input
type="hidden" name="customize-
login" value="1" />
                    <?php
                }

                ?>
                <input
type="hidden"
name="testcookie" value="1" />
            </p>
        </form>

        <?php

        if ( ! $interim_login
) {
            ?>
            <p id="nav">
                <?php

                if (
get_option(
'users_can_register' ) ) {

$registration_url = sprintf(
'<a class="wp-login-register"
href="%s">%s</a>', esc_url(
wp_registration_url() ), __(
'Register' ) );

                    /** This
filter is documented in wp-
includes/general-template.php
*/
                    echo
apply_filters( 'register',
$registration_url );

                    echo
esc_html(
$login_link_separator );
                }

                $html_link =
sprintf( '<a class="wp-login-
lost-password"
href="%s">%s</a>', esc_url(
wp_lostpassword_url() ), __(
'Lost your password?' ) );

                /**
                 * Filters the
link that allows the user to
reset the lost password.
                 *
                 * @since
6.1.0
                 *
                 * @param
string $html_link HTML link to
the lost password form.
                 */
                echo
apply_filters(
'lost_password_html_link',
$html_link );

                ?>
            </p>
            <?php
        }

        $login_script  =
'function wp_attempt_focus()
{';
        $login_script .=
'setTimeout( function() {';
        $login_script .= 'try
{';

        if ( $user_login ) {
            $login_script .=
'd = document.getElementById(
"user_pass" ); d.value = "";';
        } else {
```

```
            $login_script .=
'd = document.getElementById(
"user_login" );';

            if ( $errors-
>get_error_code() ===
'invalid_username' ) {
                $login_script
.= 'd.value = "";';
            }
        }

        $login_script .=
'd.focus(); d.select();';
        $login_script .= '}
catch( er ) {}';
        $login_script .= '},
200);';
        $login_script .=
"}\n"; // End of
wp_attempt_focus().

        /**
         * Filters whether to
print the call to
`wp_attempt_focus()` on the
login screen.
         *
         * @since 4.8.0
         *
         * @param bool $print
Whether to print the function
call. Default true.
         */
        if ( apply_filters(
'enable_login_autofocus', true
) && ! $error ) {
            $login_script .=
"wp_attempt_focus();\n";
        }

        // Run `wpOnload()` if
defined.
        $login_script .= "if (
typeof wpOnload === 'function'
) { wpOnload() }";


wp_print_inline_script_tag(
$login_script );
```

```
        if ( $interim_login )
{
            ob_start();
            ?>
            <script>
            ( function() {
                try {
                    var i,
links =
document.getElementsByTagName(
'a' );
                    for ( i in
links ) {
                        if (
links[i].href ) {

links[i].target = '_blank';

links[i].rel = 'noopener';
                        }
                    }
                } catch( er )
{}
            }());
            </script>
            <?php

wp_print_inline_script_tag(
wp_remove_surrounding_empty_sc
ript_tags( ob_get_clean() ) );
        }

        login_footer();
        break;
} // End action switch.

<?php
/**
 * Gets the email message from
the user's mailbox to add as
 * a WordPress post. Mailbox
connection information must be
 * configured under Settings >
Writing
 *
 * @package WordPress
 */

/** Make sure that the
WordPress bootstrap has run
before continuing. */
```

```
require __DIR__ . '/wp-
load.php';

/** This filter is documented
in wp-admin/options.php */
if ( ! apply_filters(
'enable_post_by_email_configur
ation', true ) ) {
    wp_die( __( 'This action
has been disabled by the
administrator.' ), 403 );
}

$mailserver_url = get_option(
'mailserver_url' );

if ( 'mail.example.com' ===
$mailserver_url || empty(
$mailserver_url ) ) {
    wp_die( __( 'This action
has been disabled by the
administrator.' ), 403 );
}

/**
 * Fires to allow a plugin to
do a complete takeover of Post
by Email.
 *
 * @since 2.9.0
 */
do_action( 'wp-mail.php' ); //
phpcs:ignore
WordPress.NamingConventions.Va
lidHookName.UseUnderscores

/** Get the POP3 class with
which to access the mailbox.
*/
require_once ABSPATH . WPINC .
'/class-pop3.php';

/** Only check at this
interval for new messages. */
if ( ! defined(
'WP_MAIL_INTERVAL' ) ) {
    define(
'WP_MAIL_INTERVAL', 5 *
MINUTE_IN_SECONDS );
}

$last_checked = get_transient(
'mailserver_last_checked' );

if ( $last_checked ) {
    wp_die( __( 'Slow down
cowboy, no need to check for
new mails so often!' ) );
}

set_transient(
'mailserver_last_checked',
true, WP_MAIL_INTERVAL );

$time_difference = get_option(
'gmt_offset' ) *
HOUR_IN_SECONDS;

$phone_delim = '::';

$pop3 = new POP3();

if ( ! $pop3->connect(
get_option( 'mailserver_url'
), get_option(
'mailserver_port' ) ) || !
$pop3->user( get_option(
'mailserver_login' ) ) ) {
    wp_die( esc_html( $pop3-
>ERROR ) );
}

$count = $pop3->pass(
get_option( 'mailserver_pass'
) );

if ( false === $count ) {
    wp_die( esc_html( $pop3-
>ERROR ) );
}

if ( 0 === $count ) {
    $pop3->quit();
    wp_die( __( 'There does
not seem to be any new mail.'
) );
}

// Always run as an
unauthenticated user.
wp_set_current_user( 0 );
```

```
for ( $i = 1; $i <= $count;
$i++ ) {

    $message = $pop3->get( $i
);

    $bodysignal
 = false;
    $boundary
 = '';
    $charset
= '';
    $content
= '';
    $content_type
 = '';
    $content_transfer_encoding
= '';
    $post_author
= 1;
    $author_found
 = false;
    $post_date
= null;
    $post_date_gmt
= null;

    foreach ( $message as
$line ) {
        // Body signal.
        if ( strlen( $line ) <
3 ) {
            $bodysignal =
true;
        }
        if ( $bodysignal ) {
            $content .= $line;
        } else {
            if ( preg_match(
'/Content-Type: /i', $line ) )
{
                $content_type
= trim( $line );
                $content_type
= substr( $content_type, 14,
strlen( $content_type ) - 14
);
                $content_type
= explode( ';', $content_type
);
                if ( ! empty(
$content_type[1] ) ) {
                    $charset =
explode( '=', $content_type[1]
);
                    $charset =
( ! empty( $charset[1] ) ) ?
trim( $charset[1] ) : '';
                }
                $content_type
= $content_type[0];
            }
            if ( preg_match(
'/Content-Transfer-Encoding:
/i', $line ) ) {

$content_transfer_encoding =
trim( $line );

$content_transfer_encoding =
substr(
$content_transfer_encoding,
27, strlen(
$content_transfer_encoding ) -
27 );

$content_transfer_encoding =
explode( ';',
$content_transfer_encoding );

$content_transfer_encoding =
$content_transfer_encoding[0];
            }
            if (
'multipart/alternative' ===
$content_type && str_contains(
$line, 'boundary="' ) && ''
=== $boundary ) {
                $boundary =
trim( $line );
                $boundary =
explode( '"', $boundary );
                $boundary =
$boundary[1];
            }
            if ( preg_match(
'/Subject: /i', $line ) ) {
                $subject =
trim( $line );
                $subject =
substr( $subject, 9, strlen(
$subject ) - 9 );
```

```
                // Captures
any text in the subject before
$phone_delim as the subject.
                if (
function_exists(
'iconv_mime_decode' ) ) {
                    $subject =
iconv_mime_decode( $subject,
2, get_option( 'blog_charset'
) );
                } else {
                    $subject =
wp_iso_descrambler( $subject
);
                }
                $subject =
explode( $phone_delim,
$subject );
                $subject =
$subject[0];
            }

            /*
             * Set the author
using the email address (From
or Reply-To, the last used)
             * otherwise use
the site admin.
             */
            if ( !
$author_found && preg_match(
'/^(From|Reply-To): /', $line
) ) {
                if (
preg_match( '|[a-z0-9_.-]+@[a-
z0-9_.-]+(?!.*<)|i', $line,
$matches ) ) {
                    $author =
$matches[0];
                } else {
                    $author =
trim( $line );
                }
                $author =
sanitize_email( $author );
                if ( is_email(
$author ) ) {
                    $userdata
= get_user_by( 'email',
$author );
                    if ( !
empty( $userdata ) ) {
$post_author  = $userdata->ID;

$author_found = true;
                    }
                }
            }

            if ( preg_match(
'/Date: /i', $line ) ) { // Of
the form '20 Mar 2002 20:32:37
+0100'.
                $ddate =
str_replace( 'Date: ', '',
trim( $line ) );
                // Remove
parenthesized timezone string
if it exists, as this confuses
strtotime().
                $ddate
  = preg_replace(
'!\s*\(.+\)\s*$!', '', $ddate
);

$ddate_timestamp = strtotime(
$ddate );
                $post_date
  = gmdate( 'Y-m-d H:i:s',
$ddate_timestamp +
$time_difference );
                $post_date_gmt
  = gmdate( 'Y-m-d H:i:s',
$ddate_timestamp );
            }
        }
    }

    // Set $post_status based
on $author_found and on
author's publish_posts
capability.
    if ( $author_found ) {
        $user        = new
WP_User( $post_author );
        $post_status = (
$user->has_cap(
'publish_posts' ) ) ?
'publish' : 'pending';
    } else {
        // Author not found in
DB, set status to pending.
Author already set to admin.
```

```
        $post_status =
'pending';
    }

    $subject = trim( $subject
);

    if (
'multipart/alternative' ===
$content_type ) {
        $content = explode( '-
-' . $boundary, $content );
        $content =
$content[2];

        // Match case-
insensitive Content-Transfer-
Encoding.
        if ( preg_match(
'/Content-Transfer-Encoding:
quoted-printable/i', $content,
$delim ) ) {
            $content =
explode( $delim[0], $content
);
            $content =
$content[1];
        }
        $content = strip_tags(
$content,
'<img><p><br><i><b><u><em><str
ong><strike><font><span><div>'
);
    }
    $content = trim( $content
);

    /**
     * Filters the original
content of the email.
     *
     * Give Post-By-Email
extending plugins full access
to the content, either
     * the raw content, or the
content of the last quoted-
printable section.
     *
     * @since 2.8.0
     *
     * @param string $content
The original email content.
     */
    $content = apply_filters(
'wp_mail_original_content',
$content );

    if ( false !== stripos(
$content_transfer_encoding,
'quoted-printable' ) ) {
        $content =
quoted_printable_decode(
$content );
    }

    if ( function_exists(
'iconv' ) && ! empty( $charset
) ) {
        $content = iconv(
$charset, get_option(
'blog_charset' ), $content );
    }

    // Captures any text in
the body after $phone_delim as
the body.
    $content = explode(
$phone_delim, $content );
    $content = empty(
$content[1] ) ? $content[0] :
$content[1];

    $content = trim( $content
);

    /**
     * Filters the content of
the post submitted by email
before saving.
     *
     * @since 1.2.0
     *
     * @param string $content
The email content.
     */
    $post_content =
apply_filters(
'phone_content', $content );

    $post_title =
xmlrpc_getposttitle( $content
);
```

```
    if ( '' === trim(
$post_title ) ) {
        $post_title =
$subject;
    }

    $post_category = array(
get_option(
'default_email_category' ) );

    $post_data = compact(
'post_content', 'post_title',
'post_date', 'post_date_gmt',
'post_author',
'post_category', 'post_status'
);
    $post_data = wp_slash(
$post_data );

    $post_ID = wp_insert_post(
$post_data );
    if ( is_wp_error( $post_ID
) ) {
        echo "\n" . $post_ID-
>get_error_message();
    }

    // The post wasn't
inserted or updated, for
whatever reason. Better move
forward to the next email.
    if ( empty( $post_ID ) ) {
        continue;
    }

    /**
     * Fires after a post
submitted by email is
published.
     *
     * @since 1.2.0
     *
     * @param int $post_ID The
post ID.
     */
    do_action(
'publish_phone', $post_ID );

    echo "\n<p><strong>" . __(
'Author:' ) . '</strong> ' .
esc_html( $post_author ) .
'</p>';

    echo "\n<p><strong>" . __(
'Posted title:' ) . '</strong>
' . esc_html( $post_title ) .
'</p>';

    if ( ! $pop3->delete( $i )
) {
        echo '<p>' . sprintf(
            /* translators:
%s: POP3 error. */
            __( 'Oops: %s' ),
            esc_html( $pop3-
>ERROR )
        ) . '</p>';
        $pop3->reset();
        exit;
    } else {
        echo '<p>' . sprintf(
            /* translators:
%s: The message ID. */
            __( 'Mission
complete. Message %s deleted.'
),
            '<strong>' . $i .
'</strong>'
        ) . '</p>';
    }
}

$pop3->quit();
```

```
<?php
/**
 * Used to set up and fix
common variables and include
 * the WordPress procedural
and class library.
 *
 * Allows for some
configuration in wp-config.php
(see default-constants.php)
 *
 * @package WordPress
 */

/**
 * Stores the location of the
WordPress directory of
functions, classes, and core
content.
 *
 * @since 1.0.0
```

```
 */
define( 'WPINC', 'wp-includes'
);

/**
 * Version information for the
current WordPress release.
 *
 * These can't be directly
globalized in version.php.
When updating,
 * include version.php from
another installation and don't
override
 * these values if already
set.
 *
 * @global string $wp_version
           The WordPress
version string.
 * @global int
 $wp_db_version
 WordPress database version.
 * @global string
$tinymce_version
 TinyMCE version.
 * @global string
$required_php_version   The
required PHP version string.
 * @global string
$required_mysql_version The
required MySQL version string.
 * @global string
$wp_local_package       Locale
code of the package.
 */
global $wp_version,
$wp_db_version,
$tinymce_version,
$required_php_version,
$required_mysql_version,
$wp_local_package;
require ABSPATH . WPINC .
'/version.php';
require ABSPATH . WPINC .
'/compat.php';
require ABSPATH . WPINC .
'/load.php';

// Check for the required PHP
version and for the MySQL
extension or a database drop-
in.
wp_check_php_mysql_versions();

// Include files required for
initialization.
require ABSPATH . WPINC .
'/class-wp-paused-extensions-
storage.php';
require ABSPATH . WPINC .
'/class-wp-fatal-error-
handler.php';
require ABSPATH . WPINC .
'/class-wp-recovery-mode-
cookie-service.php';
require ABSPATH . WPINC .
'/class-wp-recovery-mode-key-
service.php';
require ABSPATH . WPINC .
'/class-wp-recovery-mode-link-
service.php';
require ABSPATH . WPINC .
'/class-wp-recovery-mode-
email-service.php';
require ABSPATH . WPINC .
'/class-wp-recovery-mode.php';
require ABSPATH . WPINC .
'/error-protection.php';
require ABSPATH . WPINC .
'/default-constants.php';
require_once ABSPATH . WPINC .
'/plugin.php';

/**
 * If not already configured,
`$blog_id` will default to 1
in a single site
 * configuration. In
multisite, it will be
overridden by default in ms-
settings.php.
 *
 * @since 2.0.0
 *
 * @global int $blog_id
 */
global $blog_id;

// Set initial default
constants including
WP_MEMORY_LIMIT,
WP_MAX_MEMORY_LIMIT, WP_DEBUG,
```

```
SCRIPT_DEBUG, WP_CONTENT_DIR
and WP_CACHE.
wp_initial_constants();

// Register the shutdown
handler for fatal errors as
soon as possible.
wp_register_fatal_error_handle
r();

// WordPress calculates
offsets from UTC.
// phpcs:ignore
WordPress.DateTime.RestrictedF
unctions.timezone_change_date_
default_timezone_set
date_default_timezone_set(
'UTC' );

// Standardize $_SERVER
variables across setups.
wp_fix_server_vars();

// Check if the site is in
maintenance mode.
wp_maintenance();

// Start loading timer.
timer_start();

// Check if WP_DEBUG mode is
enabled.
wp_debug_mode();

/**
 * Filters whether to enable
loading of the advanced-
cache.php drop-in.
 *
 * This filter runs before it
can be used by plugins. It is
designed for non-web
 * run-times. If false is
returned, advanced-cache.php
will never be loaded.
 *
 * @since 4.6.0
 *
 * @param bool
$enable_advanced_cache Whether
to enable loading advanced-
cache.php (if present).
 *
       Default true.
 */
if ( WP_CACHE &&
apply_filters(
'enable_loading_advanced_cache
_dropin', true ) &&
file_exists( WP_CONTENT_DIR .
'/advanced-cache.php' ) ) {
    // For an advanced caching
plugin to use. Uses a static
drop-in because you would only
want one.
    include WP_CONTENT_DIR .
'/advanced-cache.php';

    // Re-initialize any hooks
added manually by advanced-
cache.php.
    if ( $wp_filter ) {
        $wp_filter =
WP_Hook::build_preinitialized_
hooks( $wp_filter );
    }
}

// Define WP_LANG_DIR if not
set.
wp_set_lang_dir();

// Load early WordPress files.
require ABSPATH . WPINC .
'/class-wp-list-util.php';
require ABSPATH . WPINC .
'/class-wp-token-map.php';
require ABSPATH . WPINC .
'/formatting.php';
require ABSPATH . WPINC .
'/meta.php';
require ABSPATH . WPINC .
'/functions.php';
require ABSPATH . WPINC .
'/class-wp-meta-query.php';
require ABSPATH . WPINC .
'/class-wp-
matchesmapregex.php';
require ABSPATH . WPINC .
'/class-wp.php';
require ABSPATH . WPINC .
'/class-wp-error.php';
require ABSPATH . WPINC .
'/pomo/mo.php';
```

```
require ABSPATH . WPINC .
'/l10n/class-wp-translation-
controller.php';
require ABSPATH . WPINC .
'/l10n/class-wp-
translations.php';
require ABSPATH . WPINC .
'/l10n/class-wp-translation-
file.php';
require ABSPATH . WPINC .
'/l10n/class-wp-translation-
file-mo.php';
require ABSPATH . WPINC .
'/l10n/class-wp-translation-
file-php.php';

/**
 * @since 0.71
 *
 * @global wpdb $wpdb
WordPress database abstraction
object.
 */
global $wpdb;
// Include the wpdb class and,
if present, a db.php database
drop-in.
require_wp_db();

/**
 * @since 3.3.0
 *
 * @global string
$table_prefix The database
table prefix.
 */
$GLOBALS['table_prefix'] =
$table_prefix;

// Set the database table
prefix and the format
specifiers for database table
columns.
wp_set_wpdb_vars();

// Start the WordPress object
cache, or an external object
cache if the drop-in is
present.
wp_start_object_cache();

// Attach the default filters.
require ABSPATH . WPINC .
'/default-filters.php';

// Initialize multisite if
enabled.
if ( is_multisite() ) {
    require ABSPATH . WPINC .
'/class-wp-site-query.php';
    require ABSPATH . WPINC .
'/class-wp-network-query.php';
    require ABSPATH . WPINC .
'/ms-blogs.php';
    require ABSPATH . WPINC .
'/ms-settings.php';
} elseif ( ! defined(
'MULTISITE' ) ) {
    define( 'MULTISITE', false
);
}

register_shutdown_function(
'shutdown_action_hook' );

// Stop most of WordPress from
being loaded if SHORTINIT is
enabled.
if ( SHORTINIT ) {
    return false;
}

// Load the L10n library.
require_once ABSPATH . WPINC .
'/l10n.php';
require_once ABSPATH . WPINC .
'/class-wp-textdomain-
registry.php';
require_once ABSPATH . WPINC .
'/class-wp-locale.php';
require_once ABSPATH . WPINC .
'/class-wp-locale-
switcher.php';

// Run the installer if
WordPress is not installed.
wp_not_installed();

// Load most of WordPress.
require ABSPATH . WPINC .
'/class-wp-walker.php';
require ABSPATH . WPINC .
'/class-wp-ajax-response.php';
```

```
require ABSPATH . WPINC .
'/capabilities.php';
require ABSPATH . WPINC .
'/class-wp-roles.php';
require ABSPATH . WPINC .
'/class-wp-role.php';
require ABSPATH . WPINC .
'/class-wp-user.php';
require ABSPATH . WPINC .
'/class-wp-query.php';
require ABSPATH . WPINC .
'/query.php';
require ABSPATH . WPINC .
'/class-wp-date-query.php';
require ABSPATH . WPINC .
'/theme.php';
require ABSPATH . WPINC .
'/class-wp-theme.php';
require ABSPATH . WPINC .
'/class-wp-theme-json-
schema.php';
require ABSPATH . WPINC .
'/class-wp-theme-json-
data.php';
require ABSPATH . WPINC .
'/class-wp-theme-json.php';
require ABSPATH . WPINC .
'/class-wp-theme-json-
resolver.php';
require ABSPATH . WPINC .
'/class-wp-duotone.php';
require ABSPATH . WPINC .
'/global-styles-and-
settings.php';
require ABSPATH . WPINC .
'/class-wp-block-
template.php';
require ABSPATH . WPINC .
'/block-template-utils.php';
require ABSPATH . WPINC .
'/block-template.php';
require ABSPATH . WPINC .
'/theme-templates.php';
require ABSPATH . WPINC .
'/theme-previews.php';
require ABSPATH . WPINC .
'/template.php';
require ABSPATH . WPINC .
'/https-detection.php';
require ABSPATH . WPINC .
'/https-migration.php';
require ABSPATH . WPINC .
'/class-wp-user-request.php';
require ABSPATH . WPINC .
'/user.php';
require ABSPATH . WPINC .
'/class-wp-user-query.php';
require ABSPATH . WPINC .
'/class-wp-session-
tokens.php';
require ABSPATH . WPINC .
'/class-wp-user-meta-session-
tokens.php';
require ABSPATH . WPINC .
'/general-template.php';
require ABSPATH . WPINC .
'/link-template.php';
require ABSPATH . WPINC .
'/author-template.php';
require ABSPATH . WPINC .
'/robots-template.php';
require ABSPATH . WPINC .
'/post.php';
require ABSPATH . WPINC .
'/class-walker-page.php';
require ABSPATH . WPINC .
'/class-walker-page-
dropdown.php';
require ABSPATH . WPINC .
'/class-wp-post-type.php';
require ABSPATH . WPINC .
'/class-wp-post.php';
require ABSPATH . WPINC .
'/post-template.php';
require ABSPATH . WPINC .
'/revision.php';
require ABSPATH . WPINC .
'/post-formats.php';
require ABSPATH . WPINC .
'/post-thumbnail-
template.php';
require ABSPATH . WPINC .
'/category.php';
require ABSPATH . WPINC .
'/class-walker-category.php';
require ABSPATH . WPINC .
'/class-walker-category-
dropdown.php';
require ABSPATH . WPINC .
'/category-template.php';
require ABSPATH . WPINC .
'/comment.php';
```

```
require ABSPATH . WPINC .
'/class-wp-comment.php';
require ABSPATH . WPINC .
'/class-wp-comment-query.php';
require ABSPATH . WPINC .
'/class-walker-comment.php';
require ABSPATH . WPINC .
'/comment-template.php';
require ABSPATH . WPINC .
'/rewrite.php';
require ABSPATH . WPINC .
'/class-wp-rewrite.php';
require ABSPATH . WPINC .
'/feed.php';
require ABSPATH . WPINC .
'/bookmark.php';
require ABSPATH . WPINC .
'/bookmark-template.php';
require ABSPATH . WPINC .
'/kses.php';
require ABSPATH . WPINC .
'/cron.php';
require ABSPATH . WPINC .
'/deprecated.php';
require ABSPATH . WPINC .
'/script-loader.php';
require ABSPATH . WPINC .
'/taxonomy.php';
require ABSPATH . WPINC .
'/class-wp-taxonomy.php';
require ABSPATH . WPINC .
'/class-wp-term.php';
require ABSPATH . WPINC .
'/class-wp-term-query.php';
require ABSPATH . WPINC .
'/class-wp-tax-query.php';
require ABSPATH . WPINC .
'/update.php';
require ABSPATH . WPINC .
'/canonical.php';
require ABSPATH . WPINC .
'/shortcodes.php';
require ABSPATH . WPINC .
'/embed.php';
require ABSPATH . WPINC .
'/class-wp-embed.php';
require ABSPATH . WPINC .
'/class-wp-oembed.php';
require ABSPATH . WPINC .
'/class-wp-oembed-
controller.php';
require ABSPATH . WPINC .
'/media.php';
require ABSPATH . WPINC .
'/http.php';
require ABSPATH . WPINC .
'/html-api/html5-named-
character-references.php';
require ABSPATH . WPINC .
'/html-api/class-wp-html-
attribute-token.php';
require ABSPATH . WPINC .
'/html-api/class-wp-html-
span.php';
require ABSPATH . WPINC .
'/html-api/class-wp-html-text-
replacement.php';
require ABSPATH . WPINC .
'/html-api/class-wp-html-
decoder.php';
require ABSPATH . WPINC .
'/html-api/class-wp-html-tag-
processor.php';
require ABSPATH . WPINC .
'/html-api/class-wp-html-
unsupported-exception.php';
require ABSPATH . WPINC .
'/html-api/class-wp-html-
active-formatting-
elements.php';
require ABSPATH . WPINC .
'/html-api/class-wp-html-open-
elements.php';
require ABSPATH . WPINC .
'/html-api/class-wp-html-
token.php';
require ABSPATH . WPINC .
'/html-api/class-wp-html-
stack-event.php';
require ABSPATH . WPINC .
'/html-api/class-wp-html-
processor-state.php';
require ABSPATH . WPINC .
'/html-api/class-wp-html-
processor.php';
require ABSPATH . WPINC .
'/class-wp-http.php';
require ABSPATH . WPINC .
'/class-wp-http-streams.php';
require ABSPATH . WPINC .
'/class-wp-http-curl.php';
require ABSPATH . WPINC .
'/class-wp-http-proxy.php';
```

```
require ABSPATH . WPINC .
'/class-wp-http-cookie.php';
require ABSPATH . WPINC .
'/class-wp-http-encoding.php';
require ABSPATH . WPINC .
'/class-wp-http-response.php';
require ABSPATH . WPINC .
'/class-wp-http-requests-
response.php';
require ABSPATH . WPINC .
'/class-wp-http-requests-
hooks.php';
require ABSPATH . WPINC .
'/widgets.php';
require ABSPATH . WPINC .
'/class-wp-widget.php';
require ABSPATH . WPINC .
'/class-wp-widget-
factory.php';
require ABSPATH . WPINC .
'/nav-menu-template.php';
require ABSPATH . WPINC .
'/nav-menu.php';
require ABSPATH . WPINC .
'/admin-bar.php';
require ABSPATH . WPINC .
'/class-wp-application-
passwords.php';
require ABSPATH . WPINC .
'/rest-api.php';
require ABSPATH . WPINC .
'/rest-api/class-wp-rest-
server.php';
require ABSPATH . WPINC .
'/rest-api/class-wp-rest-
response.php';
require ABSPATH . WPINC .
'/rest-api/class-wp-rest-
request.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-posts-controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-attachments-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-global-styles-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-post-types-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-post-statuses-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-revisions-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-global-styles-revisions-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-template-revisions-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-autosaves-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-template-autosaves-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-taxonomies-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-terms-controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-menu-items-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-menus-controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-menu-locations-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-users-controller.php';
```

```
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-comments-controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-search-controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-blocks-controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-block-types-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-block-renderer-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-settings-controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-themes-controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-plugins-controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-block-directory-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-edit-site-export-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-pattern-directory-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-block-patterns-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-block-pattern-categories-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-application-passwords-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-site-health-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-sidebars-controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-widget-types-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-widgets-controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-templates-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-url-details-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-navigation-fallback-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-font-families-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-font-faces-
controller.php';
require ABSPATH . WPINC .
'/rest-api/endpoints/class-wp-
rest-font-collections-
controller.php';
require ABSPATH . WPINC .
'/rest-api/fields/class-wp-
rest-meta-fields.php';
require ABSPATH . WPINC .
'/rest-api/fields/class-wp-
rest-comment-meta-fields.php';
require ABSPATH . WPINC .
'/rest-api/fields/class-wp-
rest-post-meta-fields.php';
require ABSPATH . WPINC .
'/rest-api/fields/class-wp-
rest-term-meta-fields.php';
require ABSPATH . WPINC .
'/rest-api/fields/class-wp-
rest-user-meta-fields.php';
```

```
require ABSPATH . WPINC .
'/rest-api/search/class-wp-
rest-search-handler.php';
require ABSPATH . WPINC .
'/rest-api/search/class-wp-
rest-post-search-handler.php';
require ABSPATH . WPINC .
'/rest-api/search/class-wp-
rest-term-search-handler.php';
require ABSPATH . WPINC .
'/rest-api/search/class-wp-
rest-post-format-search-
handler.php';
require ABSPATH . WPINC .
'/sitemaps.php';
require ABSPATH . WPINC .
'/sitemaps/class-wp-
sitemaps.php';
require ABSPATH . WPINC .
'/sitemaps/class-wp-sitemaps-
index.php';
require ABSPATH . WPINC .
'/sitemaps/class-wp-sitemaps-
provider.php';
require ABSPATH . WPINC .
'/sitemaps/class-wp-sitemaps-
registry.php';
require ABSPATH . WPINC .
'/sitemaps/class-wp-sitemaps-
renderer.php';
require ABSPATH . WPINC .
'/sitemaps/class-wp-sitemaps-
stylesheet.php';
require ABSPATH . WPINC .
'/sitemaps/providers/class-wp-
sitemaps-posts.php';
require ABSPATH . WPINC .
'/sitemaps/providers/class-wp-
sitemaps-taxonomies.php';
require ABSPATH . WPINC .
'/sitemaps/providers/class-wp-
sitemaps-users.php';
require ABSPATH . WPINC .
'/class-wp-block-bindings-
source.php';
require ABSPATH . WPINC .
'/class-wp-block-bindings-
registry.php';
require ABSPATH . WPINC .
'/class-wp-block-editor-
context.php';
require ABSPATH . WPINC .
'/class-wp-block-type.php';
require ABSPATH . WPINC .
'/class-wp-block-pattern-
categories-registry.php';
require ABSPATH . WPINC .
'/class-wp-block-patterns-
registry.php';
require ABSPATH . WPINC .
'/class-wp-block-styles-
registry.php';
require ABSPATH . WPINC .
'/class-wp-block-type-
registry.php';
require ABSPATH . WPINC .
'/class-wp-block.php';
require ABSPATH . WPINC .
'/class-wp-block-list.php';
require ABSPATH . WPINC .
'/class-wp-block-parser-
block.php';
require ABSPATH . WPINC .
'/class-wp-block-parser-
frame.php';
require ABSPATH . WPINC .
'/class-wp-block-parser.php';
require ABSPATH . WPINC .
'/class-wp-classic-to-block-
menu-converter.php';
require ABSPATH . WPINC .
'/class-wp-navigation-
fallback.php';
require ABSPATH . WPINC .
'/block-bindings.php';
require ABSPATH . WPINC .
'/block-bindings/pattern-
overrides.php';
require ABSPATH . WPINC .
'/block-bindings/post-
meta.php';
require ABSPATH . WPINC .
'/blocks.php';
require ABSPATH . WPINC .
'/blocks/index.php';
require ABSPATH . WPINC .
'/block-editor.php';
require ABSPATH . WPINC .
'/block-patterns.php';
require ABSPATH . WPINC .
'/class-wp-block-
supports.php';
```

```
require ABSPATH . WPINC .
'/block-supports/utils.php';
require ABSPATH . WPINC .
'/block-supports/align.php';
require ABSPATH . WPINC .
'/block-supports/custom-
classname.php';
require ABSPATH . WPINC .
'/block-supports/generated-
classname.php';
require ABSPATH . WPINC .
'/block-
supports/settings.php';
require ABSPATH . WPINC .
'/block-
supports/elements.php';
require ABSPATH . WPINC .
'/block-supports/colors.php';
require ABSPATH . WPINC .
'/block-
supports/typography.php';
require ABSPATH . WPINC .
'/block-supports/border.php';
require ABSPATH . WPINC .
'/block-supports/layout.php';
require ABSPATH . WPINC .
'/block-
supports/position.php';
require ABSPATH . WPINC .
'/block-supports/spacing.php';
require ABSPATH . WPINC .
'/block-
supports/dimensions.php';
require ABSPATH . WPINC .
'/block-supports/duotone.php';
require ABSPATH . WPINC .
'/block-supports/shadow.php';
require ABSPATH . WPINC .
'/block-
supports/background.php';
require ABSPATH . WPINC .
'/block-supports/block-style-
variations.php';
require ABSPATH . WPINC .
'/style-engine.php';
require ABSPATH . WPINC .
'/style-engine/class-wp-style-
engine.php';
require ABSPATH . WPINC .
'/style-engine/class-wp-style-
engine-css-declarations.php';
require ABSPATH . WPINC .
'/style-engine/class-wp-style-
engine-css-rule.php';
require ABSPATH . WPINC .
'/style-engine/class-wp-style-
engine-css-rules-store.php';
require ABSPATH . WPINC .
'/style-engine/class-wp-style-
engine-processor.php';
require ABSPATH . WPINC .
'/fonts/class-wp-font-face-
resolver.php';
require ABSPATH . WPINC .
'/fonts/class-wp-font-
collection.php';
require ABSPATH . WPINC .
'/fonts/class-wp-font-
face.php';
require ABSPATH . WPINC .
'/fonts/class-wp-font-
library.php';
require ABSPATH . WPINC .
'/fonts/class-wp-font-
utils.php';
require ABSPATH . WPINC .
'/fonts.php';
require ABSPATH . WPINC .
'/class-wp-script-
modules.php';
require ABSPATH . WPINC .
'/script-modules.php';
require ABSPATH . WPINC .
'/interactivity-api/class-wp-
interactivity-api.php';
require ABSPATH . WPINC .
'/interactivity-api/class-wp-
interactivity-api-directives-
processor.php';
require ABSPATH . WPINC .
'/interactivity-
api/interactivity-api.php';
require ABSPATH . WPINC .
'/class-wp-plugin-
dependencies.php';

add_action(
'after_setup_theme', array(
wp_script_modules(),
'add_hooks' ) );
add_action(
'after_setup_theme', array(
```

```
wp_interactivity(),
'add_hooks' ) );

/**
 * @since 3.3.0
 *
 * @global WP_Embed $wp_embed
WordPress Embed object.
 */
$GLOBALS['wp_embed'] = new
WP_Embed();

/**
 * WordPress Textdomain
Registry object.
 *
 * Used to support just-in-
time translations for manually
loaded text domains.
 *
 * @since 6.1.0
 *
 * @global
WP_Textdomain_Registry
$wp_textdomain_registry
WordPress Textdomain Registry.
 */
$GLOBALS['wp_textdomain_regist
ry'] = new
WP_Textdomain_Registry();
$GLOBALS['wp_textdomain_regist
ry']->init();

// Load multisite-specific
files.
if ( is_multisite() ) {
    require ABSPATH . WPINC .
'/ms-functions.php';
    require ABSPATH . WPINC .
'/ms-default-filters.php';
    require ABSPATH . WPINC .
'/ms-deprecated.php';
}

// Define constants that rely
on the API to obtain the
default value.
// Define must-use plugin
directory constants, which may
be overridden in the
sunrise.php drop-in.
wp_plugin_directory_constants(
);

/**
 * @since 3.9.0
 *
 * @global array
$wp_plugin_paths
 */
$GLOBALS['wp_plugin_paths'] =
array();

// Load must-use plugins.
foreach ( wp_get_mu_plugins()
as $mu_plugin ) {
    $_wp_plugin_file =
$mu_plugin;
    include_once $mu_plugin;
    $mu_plugin =
$_wp_plugin_file; // Avoid
stomping of the $mu_plugin
variable in a plugin.

    /**
     * Fires once a single
must-use plugin has loaded.
     *
     * @since 5.1.0
     *
     * @param string
$mu_plugin Full path to the
plugin's main file.
     */
    do_action(
'mu_plugin_loaded', $mu_plugin
);
}
unset( $mu_plugin,
$_wp_plugin_file );

// Load network activated
plugins.
if ( is_multisite() ) {
    foreach (
wp_get_active_network_plugins(
) as $network_plugin ) {

wp_register_plugin_realpath(
$network_plugin );

        $_wp_plugin_file =
$network_plugin;
```

```
        include_once
$network_plugin;
        $network_plugin =
$_wp_plugin_file; // Avoid
stomping of the
$network_plugin variable in a
plugin.

        /**
         * Fires once a single
network-activated plugin has
loaded.
         *
         * @since 5.1.0
         *
         * @param string
$network_plugin Full path to
the plugin's main file.
         */
        do_action(
'network_plugin_loaded',
$network_plugin );
    }
    unset( $network_plugin,
$_wp_plugin_file );
}

/**
 * Fires once all must-use and
network-activated plugins have
loaded.
 *
 * @since 2.8.0
 */
do_action( 'muplugins_loaded'
);

if ( is_multisite() ) {
    ms_cookie_constants();
}

// Define constants after
multisite is loaded.
wp_cookie_constants();

// Define and enforce our SSL
constants.
wp_ssl_constants();

// Create common globals.
require ABSPATH . WPINC .
'/vars.php';

// Make taxonomies and posts
available to plugins and
themes.
// @plugin authors: warning:
these get registered again on
the init hook.
create_initial_taxonomies();
create_initial_post_types();

wp_start_scraping_edited_file_
errors();

// Register the default theme
directory root.
register_theme_directory(
get_theme_root() );

if ( ! is_multisite() &&
wp_is_fatal_error_handler_enab
led() ) {
    // Handle users requesting
a recovery mode link and
initiating recovery mode.
    wp_recovery_mode()-
>initialize();
}

// Load active plugins.
foreach (
wp_get_active_and_valid_plugin
s() as $plugin ) {

wp_register_plugin_realpath(
$plugin );

    $_wp_plugin_file =
$plugin;
    include_once $plugin;
    $plugin =
$_wp_plugin_file; // Avoid
stomping of the $plugin
variable in a plugin.

    /**
     * Fires once a single
activated plugin has loaded.
     *
     * @since 5.1.0
     *
```

```
     * @param string $plugin
Full path to the plugin's main
file.
     */
    do_action(
'plugin_loaded', $plugin );
}
unset( $plugin,
$_wp_plugin_file );

// Load pluggable functions.
require ABSPATH . WPINC .
'/pluggable.php';
require ABSPATH . WPINC .
'/pluggable-deprecated.php';

// Set internal encoding.
wp_set_internal_encoding();

// Run wp_cache_postload() if
object cache is enabled and
the function exists.
if ( WP_CACHE &&
function_exists(
'wp_cache_postload' ) ) {
    wp_cache_postload();
}

/**
 * Fires once activated
plugins have loaded.
 *
 * Pluggable functions are
also available at this point
in the loading order.
 *
 * @since 1.5.0
 */
do_action( 'plugins_loaded' );

// Define constants which
affect functionality if not
already defined.
wp_functionality_constants();

// Add magic quotes and set up
$_REQUEST ( $_GET + $_POST ).
wp_magic_quotes();

/**
 * Fires when comment cookies
are sanitized.
 *
 * @since 2.0.11
 */
do_action(
'sanitize_comment_cookies' );

/**
 * WordPress Query object
 *
 * @since 2.0.0
 *
 * @global WP_Query
$wp_the_query WordPress Query
object.
 */
$GLOBALS['wp_the_query'] = new
WP_Query();

/**
 * Holds the reference to
{@see $wp_the_query}.
 * Use this global for
WordPress queries
 *
 * @since 1.5.0
 *
 * @global WP_Query $wp_query
WordPress Query object.
 */
$GLOBALS['wp_query'] =
$GLOBALS['wp_the_query'];

/**
 * Holds the WordPress Rewrite
object for creating pretty
URLs
 *
 * @since 1.5.0
 *
 * @global WP_Rewrite
$wp_rewrite WordPress rewrite
component.
 */
$GLOBALS['wp_rewrite'] = new
WP_Rewrite();

/**
 * WordPress Object
 *
 * @since 2.0.0
 *
```

```
 * @global WP $wp Current
WordPress environment
instance.
 */
$GLOBALS['wp'] = new WP();

/**
 * WordPress Widget Factory
Object
 *
 * @since 2.8.0
 *
 * @global WP_Widget_Factory
$wp_widget_factory
 */
$GLOBALS['wp_widget_factory']
= new WP_Widget_Factory();

/**
 * WordPress User Roles
 *
 * @since 2.0.0
 *
 * @global WP_Roles $wp_roles
WordPress role management
object.
 */
$GLOBALS['wp_roles'] = new
WP_Roles();

/**
 * Fires before the theme is
loaded.
 *
 * @since 2.6.0
 */
do_action( 'setup_theme' );

// Define the template related
constants and globals.
wp_templating_constants();
wp_set_template_globals();

// Load the default text
localization domain.
load_default_textdomain();

$locale      = get_locale();
$locale_file = WP_LANG_DIR .
"/$locale.php";

if ( ( 0 === validate_file(
$locale ) ) && is_readable(
$locale_file ) ) {
    require $locale_file;
}
unset( $locale_file );

/**
 * WordPress Locale object for
loading locale domain date and
various strings.
 *
 * @since 2.1.0
 *
 * @global WP_Locale
$wp_locale WordPress date and
time locale object.
 */
$GLOBALS['wp_locale'] = new
WP_Locale();

/**
 * WordPress Locale Switcher
object for switching locales.
 *
 * @since 4.7.0
 *
 * @global WP_Locale_Switcher
$wp_locale_switcher WordPress
locale switcher object.
 */
$GLOBALS['wp_locale_switcher']
= new WP_Locale_Switcher();
$GLOBALS['wp_locale_switcher']
->init();

// Load the functions for the
active theme, for both parent
and child theme if applicable.
foreach (
wp_get_active_and_valid_themes
() as $theme ) {
    if ( file_exists( $theme .
'/functions.php' ) ) {
        include $theme .
'/functions.php';
    }
}
unset( $theme );

/**
```

```
 * Fires after the theme is
loaded.
 *
 * @since 3.0.0
 */
do_action( 'after_setup_theme'
);

// Create an instance of
WP_Site_Health so that Cron
events may fire.
if ( ! class_exists(
'WP_Site_Health' ) ) {
    require_once ABSPATH .
'wp-admin/includes/class-wp-
site-health.php';
}
WP_Site_Health::get_instance()
;

// Set up current user.
$GLOBALS['wp']->init();

/**
 * Fires after WordPress has
finished loading but before
any headers are sent.
 *
 * Most of WP is loaded at
this stage, and the user is
authenticated. WP continues
 * to load on the {@see
'init'} hook that follows
(e.g. widgets), and many
plugins instantiate
 * themselves on it for all
sorts of reasons (e.g. they
need a user, a taxonomy,
etc.).
 *
 * If you wish to plug an
action once WP is loaded, use
the {@see 'wp_loaded'} hook
below.
 *
 * @since 1.5.0
 */
do_action( 'init' );

// Check site status.
if ( is_multisite() ) {
    $file = ms_site_check();
    if ( true !== $file ) {
        require $file;
        die();
    }
    unset( $file );
}

/**
 * This hook is fired once WP,
all plugins, and the theme are
fully loaded and instantiated.
 *
 * Ajax requests should use
wp-admin/admin-ajax.php.
admin-ajax.php can handle
requests for
 * users not logged in.
 *
 * @link
https://developer.wordpress.or
g/plugins/javascript/ajax
 *
 * @since 3.0.0
 */
do_action( 'wp_loaded' );

<?php

/** Sets up the WordPress
Environment. */
require __DIR__ . '/wp-
load.php';

add_filter( 'wp_robots',
'wp_robots_no_robots' );

require __DIR__ . '/wp-blog-
header.php';

nocache_headers();

if ( is_array(
get_site_option(
'illegal_names' ) ) && isset(
$_GET['new'] ) && in_array(
$_GET['new'], get_site_option(
'illegal_names' ), true ) ) {
    wp_redirect(
network_home_url() );
    die();
}
```

```
/**
 * Prints signup_header via
wp_head.
 *
 * @since MU (3.0.0)
 */
function do_signup_header() {
    /**
     * Fires within the head
section of the site sign-up
screen.
     *
     * @since 3.0.0
     */
    do_action( 'signup_header'
);
}
add_action( 'wp_head',
'do_signup_header' );

if ( ! is_multisite() ) {
    wp_redirect(
wp_registration_url() );
    die();
}

if ( ! is_main_site() ) {
    wp_redirect(
network_site_url( 'wp-
signup.php' ) );
    die();
}

// Fix for page title.
$wp_query->is_404 = false;

/**
 * Fires before the Site Sign-
up page is loaded.
 *
 * @since 4.4.0
 */
do_action(
'before_signup_header' );

/**
 * Prints styles for front-end
Multisite Sign-up pages.
 *
 * @since MU (3.0.0)
 */
function
wpmu_signup_stylesheet() {
    ?>
    <style type="text/css">
        .mu_register { width:
90%; margin: 0 auto; }
        .mu_register form {
margin-top: 2em; }
        .mu_register fieldset,
            .mu_register
legend { margin: 0; padding:
0; border: none; }
        .mu_register .error {
font-weight: 600; padding:
10px; color: #333; background:
#ffebe8; border: 1px solid
#c00; }
        .mu_register
input[type="submit"],
            .mu_register
#blog_title,
            .mu_register
#user_email,
            .mu_register
#blogname,
            .mu_register
#user_name { width: 100%;
font-size: 24px; margin: 5px
0; box-sizing: border-box; }
        .mu_register #site-
language { display: block; }
        .mu_register
.prefix_address,
            .mu_register
.suffix_address { font-size:
18px; display: inline-block;
direction: ltr; }
        .mu_register label,
            .mu_register
legend,
            .mu_register
.label-heading { font-weight:
600; font-size: 15px; display:
block; margin: 10px 0; }
        .mu_register legend +
p,
            .mu_register input
+ p { margin-top: 0; }
        .mu_register
label.checkbox { display:
inline; }
```

```
        .mu_register .mu_alert
{ font-weight: 600; padding:
10px; color: #333; background:
#ffffe0; border: 1px solid
#e6db55; }
        .mu_register .mu_alert
a { color: inherit; text-
decoration: underline; }
        .mu_register .signup-
options .wp-signup-radio-
button { display: block; }
        .mu_register .privacy-
intro .wp-signup-radio-button
{ margin-right: 0.5em; }
        .rtl .mu_register .wp-
signup-blogname { direction:
ltr; text-align: right; }
    </style>
    <?php
}
add_action( 'wp_head',
'wpmu_signup_stylesheet' );

get_header( 'wp-signup' );

/**
 * Fires before the site Sign-
up form.
 *
 * @since 3.0.0
 */
do_action(
'before_signup_form' );
?>
<div id="signup-content"
class="widecolumn">
<div class="mu_register wp-
signup-container" role="main">
<?php
/**
 * Generates and displays the
Sign-up and Create Site forms.
 *
 * @since MU (3.0.0)
 *
 * @param string
 $blogname   The new site
name.
 * @param string
 $blog_title The new site
title.
 * @param WP_Error|string
$errors     A WP_Error object
containing existing errors.
Defaults to empty string.
 */
function show_blog_form(
$blogname = '', $blog_title =
'', $errors = '' ) {
    if ( ! is_wp_error(
$errors ) ) {
        $errors = new
WP_Error();
    }

    $current_network =
get_network();
    // Site name.
    if ( !
is_subdomain_install() ) {
        echo '<label
for="blogname">' . __( 'Site
Name (subdirectory only):' ) .
'</label>';
    } else {
        echo '<label
for="blogname">' . __( 'Site
Domain (subdomain only):' ) .
'</label>';
    }

    $errmsg_blogname      =
$errors->get_error_message(
'blogname' );
    $errmsg_blogname_aria =
'';
    if ( $errmsg_blogname ) {
        $errmsg_blogname_aria
= 'wp-signup-blogname-error ';
        echo '<p class="error"
id="wp-signup-blogname-
error">' . $errmsg_blogname .
'</p>';
    }

    if ( !
is_subdomain_install() ) {
        echo '<div class="wp-
signup-blogname"><span
class="prefix_address"
id="prefix-address">' .
$current_network->domain .
$current_network->path .
```

```
'</span><input name="blogname"
type="text" id="blogname"
value="' . esc_attr( $blogname
) . '" maxlength="60"
autocomplete="off"
required="required" aria-
describedby="' .
$errmsg_blogname_aria .
'prefix-address" /></div>';
    } else {
        $site_domain =
preg_replace( '|^www\.|', '',
$current_network->domain );
        echo '<div class="wp-
signup-blogname"><input
name="blogname" type="text"
id="blogname" value="' .
esc_attr( $blogname ) . '"
maxlength="60"
autocomplete="off"
required="required" aria-
describedby="' .
$errmsg_blogname_aria .
'suffix-address" /><span
class="suffix_address"
id="suffix-address">.' .
esc_html( $site_domain ) .
'</span></div>';
    }

    if ( ! is_user_logged_in()
) {
        if ( !
is_subdomain_install() ) {
            $site =
$current_network->domain .
$current_network->path . __(
'sitename' );
        } else {
            $site = __(
'domain' ) . '.' .
$site_domain .
$current_network->path;
        }

        printf(

'<p>(<strong>%s</strong>)
%s</p>',
            /* translators:
%s: Site address. */
            sprintf( __( 'Your
address will be %s.' ), $site
),
            __( 'Must be at
least 4 characters, letters
and numbers only. It cannot be
changed, so choose carefully!'
)
        );
    }

    // Site Title.
    ?>
    <label
for="blog_title"><?php _e(
'Site Title:' ); ?></label>
    <?php
    $errmsg_blog_title      =
$errors->get_error_message(
'blog_title' );
    $errmsg_blog_title_aria =
'';
    if ( $errmsg_blog_title )
{

$errmsg_blog_title_aria = '
aria-describedby="wp-signup-
blog-title-error"';
        echo '<p class="error"
id="wp-signup-blog-title-
error">' . $errmsg_blog_title
. '</p>';
    }
    echo '<input
name="blog_title" type="text"
id="blog_title" value="' .
esc_attr( $blog_title ) . '"
required="required"
autocomplete="off"' .
$errmsg_blog_title_aria . '
/>';
    ?>

    <?php
    // Site Language.
    $languages =
signup_get_available_languages
();

    if ( ! empty( $languages )
) :
        ?>
```

```
        <p>
            <label for="site-
language"><?php _e( 'Site
Language:' ); ?></label>
            <?php
            // Network
default.
            $lang =
get_site_option( 'WPLANG' );

            if ( isset(
$_POST['WPLANG'] ) ) {
                $lang =
$_POST['WPLANG'];
            }

            // Use US English
if the default isn't
available.
            if ( ! in_array(
$lang, $languages, true ) ) {
                $lang = '';
            }


wp_dropdown_languages(
                array(
                    'name'
                   =>
'WPLANG',
                    'id'
                   => 'site-
language',
                    'selected'
                   => $lang,

'languages'
=> $languages,

'show_available_translations'
=> false,
                )
            );
            ?>
        </p>
        <?php
        endif; // Languages.


$blog_public_on_checked  = '';

$blog_public_off_checked = '';

    if ( isset(
$_POST['blog_public'] ) && '0'
=== $_POST['blog_public'] ) {

$blog_public_off_checked =
'checked="checked"';
    } else {

$blog_public_on_checked =
'checked="checked"';
    }
    ?>

    <div id="privacy">
        <fieldset
class="privacy-intro">
            <legend>
                <span
class="label-heading"><?php
_e( 'Privacy:' ); ?></span>
                <?php _e(
'Allow search engines to index
this site.' ); ?>
            </legend>
            <p class="wp-
signup-radio-buttons">
                <span
class="wp-signup-radio-
button">
                    <input
type="radio"
id="blog_public_on"
name="blog_public" value="1"
<?php echo
$blog_public_on_checked; ?> />
                    <label
class="checkbox"
for="blog_public_on"><?php _e(
'Yes' ); ?></label>
                </span>
                <span
class="wp-signup-radio-
button">
                    <input
type="radio"
id="blog_public_off"
name="blog_public" value="0"
<?php echo
$blog_public_off_checked; ?>
/>
                    <label
class="checkbox"
```

```
for="blog_public_off"><?php
_e( 'No' ); ?></label>
                </span>
            </p>
        </fieldset>
    </div>

    <?php
    /**
     * Fires after the site
sign-up form.
     *
     * @since 3.0.0
     *
     * @param WP_Error $errors
A WP_Error object possibly
containing 'blogname' or
'blog_title' errors.
     */
    do_action(
'signup_blogform', $errors );
}

/**
 * Validates the new site
sign-up.
 *
 * @since MU (3.0.0)
 *
 * @return array Contains the
new site data and error
messages.
 *               See
wpmu_validate_blog_signup()
for details.
 */
function validate_blog_form()
{
    $user = '';
    if ( is_user_logged_in() )
{
        $user =
wp_get_current_user();
    }

    return
wpmu_validate_blog_signup(
$_POST['blogname'],
$_POST['blog_title'], $user );
}

/**
 * Displays the fields for the
new user account registration
form.
 *
 * @since MU (3.0.0)
 *
 * @param string
 $user_name  The entered
username.
 * @param string
 $user_email The entered email
address.
 * @param WP_Error|string
$errors     A WP_Error object
containing existing errors.
Defaults to empty string.
 */
function show_user_form(
$user_name = '', $user_email =
'', $errors = '' ) {
    if ( ! is_wp_error(
$errors ) ) {
        $errors = new
WP_Error();
    }

    // Username.
    echo '<label
for="user_name">' . __(
'Username:' ) . '</label>';
    $errmsg_username      =
$errors->get_error_message(
'user_name' );
    $errmsg_username_aria =
'';
    if ( $errmsg_username ) {
        $errmsg_username_aria
= 'wp-signup-username-error ';
        echo '<p class="error"
id="wp-signup-username-
error">' . $errmsg_username .
'</p>';
    }
    ?>
    <input name="user_name"
type="text" id="user_name"
value="<?php echo esc_attr(
$user_name ); ?>"
autocapitalize="none"
autocorrect="off"
maxlength="60"
autocomplete="username"
```

```
required="required" aria-
describedby="<?php echo
$errmsg_username_aria; ?>wp-
signup-username-description"
/>
    <p id="wp-signup-username-
description"><?php _e( '(Must
be at least 4 characters,
lowercase letters and numbers
only.)' ); ?></p>

    <?php
    // Email address.
    echo '<label
for="user_email">' . __(
'Email Address:' ) .
'</label>';
    $errmsg_email      =
$errors->get_error_message(
'user_email' );
    $errmsg_email_aria = '';
    if ( $errmsg_email ) {
        $errmsg_email_aria =
'wp-signup-email-error ';
        echo '<p class="error"
id="wp-signup-email-error">' .
$errmsg_email . '</p>';
    }
    ?>
    <input name="user_email"
type="email" id="user_email"
value="<?php echo esc_attr(
$user_email ); ?>"
maxlength="200"
autocomplete="email"
required="required" aria-
describedby="<?php echo
$errmsg_email_aria; ?>wp-
signup-email-description" />
    <p id="wp-signup-email-
description"><?php _e( 'Your
registration email is sent to
this address. (Double-check
your email address before
continuing.)' ); ?></p>

    <?php
    // Extra fields.
    $errmsg_generic = $errors-
>get_error_message( 'generic'
);
    if ( $errmsg_generic ) {
        echo '<p class="error"
id="wp-signup-generic-error">'
. $errmsg_generic . '</p>';
    }
    /**
     * Fires at the end of the
new user account registration
form.
     *
     * @since 3.0.0
     *
     * @param WP_Error $errors
A WP_Error object containing
'user_name' or 'user_email'
errors.
     */
    do_action(
'signup_extra_fields', $errors
);
}

/**
 * Validates user sign-up name
and email.
 *
 * @since MU (3.0.0)
 *
 * @return array Contains
username, email, and error
messages.
 *               See
wpmu_validate_user_signup()
for details.
 */
function validate_user_form()
{
    return
wpmu_validate_user_signup(
$_POST['user_name'],
$_POST['user_email'] );
}

/**
 * Shows a form for returning
users to sign up for another
site.
 *
 * @since MU (3.0.0)
 *
 * @param string
 $blogname   The new site name
```

```
 * @param string
 $blog_title The new site
title.
 * @param WP_Error|string
$errors     A WP_Error object
containing existing errors.
Defaults to empty string.
 */
function signup_another_blog(
$blogname = '', $blog_title =
'', $errors = '' ) {
    $current_user =
wp_get_current_user();

    if ( ! is_wp_error(
$errors ) ) {
        $errors = new
WP_Error();
    }

    $signup_defaults = array(
        'blogname'   =>
$blogname,
        'blog_title' =>
$blog_title,
        'errors'     =>
$errors,
    );

    /**
     * Filters the default
site sign-up variables.
     *
     * @since 3.0.0
     *
     * @param array
$signup_defaults {
     *     An array of default
site sign-up variables.
     *
     *     @type string
$blogname   The site blogname.
     *     @type string
$blog_title The site title.
     *     @type WP_Error
$errors     A WP_Error object
possibly containing 'blogname'
or 'blog_title' errors.
     * }
     */
    $filtered_results =
apply_filters(
'signup_another_blog_init',
$signup_defaults );

    $blogname   =
$filtered_results['blogname'];
    $blog_title =
$filtered_results['blog_title'
];
    $errors     =
$filtered_results['errors'];

    /* translators: %s:
Network title. */
    echo '<h2>' . sprintf( __(
'Get <em>another</em> %s site
in seconds' ), get_network()-
>site_name ) . '</h2>';

    if ( $errors->has_errors()
) {
        echo '<p>' . __(
'There was a problem, please
correct the form below and try
again.' ) . '</p>';
    }
    ?>
    <p>
        <?php
        printf(
            /* translators:
%s: Current user's display
name. */
            __( 'Welcome back,
%s. By filling out the form
below, you can <strong>add
another site to your
account</strong>. There is no
limit to the number of sites
you can have, so create to
your heart&#8217;s content,
but write responsibly!' ),
            $current_user-
>display_name
        );
        ?>
    </p>

    <?php
    $blogs =
get_blogs_of_user(
$current_user->ID );
    if ( ! empty( $blogs ) ) {
```

```
        ?>

            <p><?php _e(
'Sites you are already a
member of:' ); ?></p>
            <ul>
                <?php
                foreach (
$blogs as $blog ) {
                    $home_url
= get_home_url( $blog-
>userblog_id );
                    echo
'<li><a href="' . esc_url(
$home_url ) . '">' . $home_url
. '</a></li>';
                }
                ?>
            </ul>
    <?php } ?>

    <p><?php _e( 'If you are
not going to use a great site
domain, leave it for a new
user. Now have at it!' );
?></p>
    <form id="setupform"
method="post" action="wp-
signup.php">
        <input type="hidden"
name="stage"
value="gimmeanotherblog" />
        <?php
        /**
         * Fires when hidden
sign-up form fields output
when creating another site or
user.
         *
         * @since MU (3.0.0)
         *
         * @param string
$context A string describing
the steps of the sign-up
process. The value can be
         *
   'create-another-site',
'validate-user', or 'validate-
site'.
         */
        do_action(
'signup_hidden_fields',
'create-another-site' );
        ?>
        <?php show_blog_form(
$blogname, $blog_title,
$errors ); ?>
        <p
class="submit"><input
type="submit" name="submit"
class="submit" value="<?php
esc_attr_e( 'Create Site' );
?>" /></p>
    </form>
    <?php
}

/**
 * Validates a new site sign-
up for an existing user.
 *
 * @since MU (3.0.0)
 *
 * @global string   $blogname
  The new site's subdomain or
directory name.
 * @global string
$blog_title The new site's
title.
 * @global WP_Error $errors
  Existing errors in the
global scope.
 * @global string   $domain
  The new site's domain.
 * @global string   $path
  The new site's path.
 *
 * @return null|bool True if
site signup was validated,
false on error.
 *                    The
function halts all execution
if the user is not logged in.
 */
function
validate_another_blog_signup()
{
    global $blogname,
$blog_title, $errors, $domain,
$path;
    $current_user =
wp_get_current_user();
```

```
    if ( ! is_user_logged_in()
) {
        die();
    }

    $result =
validate_blog_form();

    // Extracted values
set/overwrite globals.
    $domain     =
$result['domain'];
    $path       =
$result['path'];
    $blogname   =
$result['blogname'];
    $blog_title =
$result['blog_title'];
    $errors     =
$result['errors'];

    if ( $errors->has_errors()
) {
        signup_another_blog(
$blogname, $blog_title,
$errors );
        return false;
    }

    $public = (int)
$_POST['blog_public'];

    $blog_meta_defaults =
array(
        'lang_id' => 1,
        'public'  => $public,
    );

    // Handle the language
setting for the new site.
    if ( ! empty(
$_POST['WPLANG'] ) ) {

        $languages =
signup_get_available_languages
();

        if ( in_array(
$_POST['WPLANG'], $languages,
true ) ) {
            $language =
wp_unslash(
sanitize_text_field(
$_POST['WPLANG'] ) );

            if ( $language ) {

$blog_meta_defaults['WPLANG']
= $language;
            }
        }
    }

    /**
     * Filters the new site
meta variables.
     *
     * Use the {@see
'add_signup_meta'} filter
instead.
     *
     * @since MU (3.0.0)
     * @deprecated 3.0.0 Use
the {@see 'add_signup_meta'}
filter instead.
     *
     * @param array
$blog_meta_defaults An array
of default blog meta
variables.
     */
    $meta_defaults =
apply_filters_deprecated(
'signup_create_blog_meta',
array( $blog_meta_defaults ),
'3.0.0', 'add_signup_meta' );

    /**
     * Filters the new default
site meta variables.
     *
     * @since 3.0.0
     *
     * @param array $meta {
     *     An array of default
site meta variables.
     *
     *     @type int $lang_id
    The language ID.
     *     @type int
$blog_public Whether search
engines should be discouraged
from indexing the site. 1 for
true, 0 for false.
```

```
     * }
     */
    $meta = apply_filters(
'add_signup_meta',
$meta_defaults );

    $blog_id =
wpmu_create_blog( $domain,
$path, $blog_title,
$current_user->ID, $meta,
get_current_network_id() );

    if ( is_wp_error( $blog_id
) ) {
        return false;
    }


confirm_another_blog_signup(
$domain, $path, $blog_title,
$current_user->user_login,
$current_user->user_email,
$meta, $blog_id );
    return true;
}

/**
 * Shows a message confirming
that the new site has been
created.
 *
 * @since MU (3.0.0)
 * @since 4.4.0 Added the
`$blog_id` parameter.
 *
 * @param string $domain
The domain URL.
 * @param string $path
The site root path.
 * @param string $blog_title
The site title.
 * @param string $user_name
 The username.
 * @param string $user_email
The user's email address.
 * @param array  $meta
Any additional meta from the
{@see 'add_signup_meta'}
filter in
validate_blog_signup().
 * @param int    $blog_id
 The site ID.
 */
function
confirm_another_blog_signup(
$domain, $path, $blog_title,
$user_name, $user_email = '',
$meta = array(), $blog_id = 0
) {

    if ( $blog_id ) {
        switch_to_blog(
$blog_id );
        $home_url  = home_url(
'/' );
        $login_url =
wp_login_url();

restore_current_blog();
    } else {
        $home_url  = 'http://'
. $domain . $path;
        $login_url = 'http://'
. $domain . $path . 'wp-
login.php';
    }

    $site = sprintf(
        '<a
href="%1$s">%2$s</a>',
        esc_url( $home_url ),
        $blog_title
    );

    ?>
    <h2>
    <?php
        /* translators: %s:
Site title. */
        printf( __( 'The site
%s is yours.' ), $site );
    ?>
    </h2>
    <p>
        <?php
        printf(
            /* translators: 1:
Link to new site, 2: Login
URL, 3: Username. */
            __( '%1$s is your
new site. <a href="%2$s">Log
in</a> as &#8220;%3$s&#8221;
using your existing password.'
),
```

```
            sprintf(
                '<a
href="%s">%s</a>',
                esc_url(
$home_url ),

untrailingslashit( $domain .
$path )
            ),
            esc_url(
$login_url ),
            $user_name
        );
        ?>
    </p>
    <?php
    /**
     * Fires when the site or
user sign-up process is
complete.
     *
     * @since 3.0.0
     */
    do_action(
'signup_finished' );
}

/**
 * Shows a form for a visitor
to sign up for a new user
account.
 *
 * @since MU (3.0.0)
 *
 * @global string
$active_signup String that
returns registration type. The
value can be
 *
  'all', 'none', 'blog', or
'user'.
 *
 * @param string
 $user_name  The username.
 * @param string
 $user_email The user's email.
 * @param WP_Error|string
$errors     A WP_Error object
containing existing errors.
Defaults to empty string.
 */
function signup_user(
$user_name = '', $user_email =
'', $errors = '' ) {
    global $active_signup;

    if ( ! is_wp_error(
$errors ) ) {
        $errors = new
WP_Error();
    }

    $signup_for = isset(
$_POST['signup_for'] ) ?
esc_html( $_POST['signup_for']
) : 'blog';

    $signup_user_defaults =
array(
        'user_name'  =>
$user_name,
        'user_email' =>
$user_email,
        'errors'     =>
$errors,
    );

    /**
     * Filters the default
user variables used on the
user sign-up form.
     *
     * @since 3.0.0
     *
     * @param array
$signup_user_defaults {
     *     An array of default
user variables.
     *
     *     @type string
$user_name  The user username.
     *     @type string
$user_email The user email
address.
     *     @type WP_Error
$errors     A WP_Error object
with possible errors relevant
to the sign-up user.
     * }
     */
    $filtered_results =
apply_filters(
```

```
'signup_user_init',
$signup_user_defaults );
    $user_name        =
$filtered_results['user_name']
;
    $user_email       =
$filtered_results['user_email'
];
    $errors           =
$filtered_results['errors'];

    ?>

    <h2>
    <?php
        /* translators: %s:
Name of the network. */
        printf( __( 'Get your
own %s account in seconds' ),
get_network()->site_name );
    ?>
    </h2>
    <form id="setupform"
method="post" action="wp-
signup.php"
novalidate="novalidate">
        <input type="hidden"
name="stage" value="validate-
user-signup" />
        <?php
        /** This action is
documented in wp-signup.php */
        do_action(
'signup_hidden_fields',
'validate-user' );
        ?>
        <?php show_user_form(
$user_name, $user_email,
$errors ); ?>

        <?php if ( 'blog' ===
$active_signup ) : ?>
            <input
id="signupblog" type="hidden"
name="signup_for" value="blog"
/>
        <?php elseif ( 'user'
=== $active_signup ) : ?>
            <input
id="signupblog" type="hidden"
name="signup_for" value="user"
/>
        <?php else : ?>
            <fieldset
class="signup-options">
                <legend><?php
_e( 'Create a site or only a
username:' ); ?></legend>
                <p class="wp-
signup-radio-buttons">
                    <span
class="wp-signup-radio-
button">
                        <input
id="signupblog" type="radio"
name="signup_for" value="blog"
<?php checked( $signup_for,
'blog' ); ?> />
                        <label
class="checkbox"
for="signupblog"><?php _e(
'Gimme a site!' ); ?></label>
                    </span>
                    <span
class="wp-signup-radio-
button">
                        <input
id="signupuser" type="radio"
name="signup_for" value="user"
<?php checked( $signup_for,
'user' ); ?> />
                        <label
class="checkbox"
for="signupuser"><?php _e(
'Just a username, please.' );
?></label>
                    </span>
                </p>
            </fieldset>
        <?php endif; ?>

        <p
class="submit"><input
type="submit" name="submit"
class="submit" value="<?php
esc_attr_e( 'Next' ); ?>"
/></p>
    </form>
    <?php
}

/**
 * Validates the new user
sign-up.
```

```
 *
 * @since MU (3.0.0)
 *
 * @return bool True if new
user sign-up was validated,
false on error.
 */
function
validate_user_signup() {
    $result     =
validate_user_form();
    $user_name  =
$result['user_name'];
    $user_email =
$result['user_email'];
    $errors     =
$result['errors'];

    if ( $errors->has_errors()
) {
        signup_user(
$user_name, $user_email,
$errors );
        return false;
    }

    if ( 'blog' ===
$_POST['signup_for'] ) {
        signup_blog(
$user_name, $user_email );
        return false;
    }

    /** This filter is
documented in wp-signup.php */
    wpmu_signup_user(
$user_name, $user_email,
apply_filters(
'add_signup_meta', array() )
);

    confirm_user_signup(
$user_name, $user_email );
    return true;
}

/**
 * Shows a message confirming
that the new user has been
registered and is awaiting
activation.
 *
 * @since MU (3.0.0)
 *
 * @param string $user_name
 The username.
 * @param string $user_email
The user's email address.
 */
function confirm_user_signup(
$user_name, $user_email ) {
    ?>
    <h2>
    <?php
    /* translators: %s:
Username. */
    printf( __( '%s is your
new username' ), $user_name )
    ?>
    </h2>
    <p><?php _e( 'But, before
you can start using your new
username, <strong>you must
activate it</strong>.' );
?></p>
    <p>
    <?php
    /* translators: %s: The
user email address. */
    printf( __( 'Check your
inbox at %s and click on the
given link.' ), '<strong>' .
$user_email . '</strong>' );
    ?>
    </p>
    <p><?php _e( 'If you do
not activate your username
within two days, you will have
to sign up again.' ); ?></p>
    <?php
    /** This action is
documented in wp-signup.php */
    do_action(
'signup_finished' );
}

/**
 * Shows a form for a user or
visitor to sign up for a new
site.
 *
 * @since MU (3.0.0)
 *
```

```
 * @param string
 $user_name  The username.
 * @param string
 $user_email The user's email
address.
 * @param string
 $blogname   The site name.
 * @param string
 $blog_title The site title.
 * @param WP_Error|string
$errors     A WP_Error object
containing existing errors.
Defaults to empty string.
 */
function signup_blog(
$user_name = '', $user_email =
'', $blogname = '',
$blog_title = '', $errors = ''
) {
    if ( ! is_wp_error(
$errors ) ) {
        $errors = new
WP_Error();
    }

    $signup_blog_defaults =
array(
        'user_name'  =>
$user_name,
        'user_email' =>
$user_email,
        'blogname'   =>
$blogname,
        'blog_title' =>
$blog_title,
        'errors'     =>
$errors,
    );

    /**
     * Filters the default
site creation variables for
the site sign-up form.
     *
     * @since 3.0.0
     *
     * @param array
$signup_blog_defaults {
     *     An array of default
site creation variables.
     *
     *     @type string
$user_name  The user username.
     *     @type string
$user_email The user email
address.
     *     @type string
$blogname   The blogname.
     *     @type string
$blog_title The title of the
site.
     *     @type WP_Error
$errors     A WP_Error object
with possible errors relevant
to new site creation
variables.
     * }
     */
    $filtered_results =
apply_filters(
'signup_blog_init',
$signup_blog_defaults );

    $user_name  =
$filtered_results['user_name']
;
    $user_email =
$filtered_results['user_email'
];
    $blogname   =
$filtered_results['blogname'];
    $blog_title =
$filtered_results['blog_title'
];
    $errors     =
$filtered_results['errors'];

    if ( empty( $blogname ) )
{
        $blogname =
$user_name;
    }
    ?>
    <form id="setupform"
method="post" action="wp-
signup.php">
        <input type="hidden"
name="stage" value="validate-
blog-signup" />
        <input type="hidden"
name="user_name" value="<?php
echo esc_attr( $user_name );
?>" />
```

```
        <input type="hidden"
name="user_email" value="<?php
echo esc_attr( $user_email );
?>" />
        <?php
        /** This action is
documented in wp-signup.php */
        do_action(
'signup_hidden_fields',
'validate-site' );
        ?>
        <?php show_blog_form(
$blogname, $blog_title,
$errors ); ?>
        <p
class="submit"><input
type="submit" name="submit"
class="submit" value="<?php
esc_attr_e( 'Sign up' ); ?>"
/></p>
    </form>
    <?php
}

/**
 * Validates new site signup.
 *
 * @since MU (3.0.0)
 *
 * @return bool True if the
site sign-up was validated,
false on error.
 */
function
validate_blog_signup() {
    // Re-validate user info.
    $user_result =
wpmu_validate_user_signup(
$_POST['user_name'],
$_POST['user_email'] );
    $user_name   =
$user_result['user_name'];
    $user_email  =
$user_result['user_email'];
    $user_errors =
$user_result['errors'];

    if ( $user_errors-
>has_errors() ) {
        signup_user(
$user_name, $user_email,
$user_errors );
        return false;
    }

    $result     =
wpmu_validate_blog_signup(
$_POST['blogname'],
$_POST['blog_title'] );
    $domain     =
$result['domain'];
    $path       =
$result['path'];
    $blogname   =
$result['blogname'];
    $blog_title =
$result['blog_title'];
    $errors     =
$result['errors'];

    if ( $errors->has_errors()
) {
        signup_blog(
$user_name, $user_email,
$blogname, $blog_title,
$errors );
        return false;
    }

    $public      = (int)
$_POST['blog_public'];
    $signup_meta = array(
        'lang_id' => 1,
        'public'  => $public,
    );

    // Handle the language
setting for the new site.
    if ( ! empty(
$_POST['WPLANG'] ) ) {

        $languages =
signup_get_available_languages
();

        if ( in_array(
$_POST['WPLANG'], $languages,
true ) ) {
            $language =
wp_unslash(
sanitize_text_field(
$_POST['WPLANG'] ) );

            if ( $language ) {
```

```
$signup_meta['WPLANG'] =
$language;
            }
        }
    }

    /** This filter is
documented in wp-signup.php */
    $meta = apply_filters(
'add_signup_meta',
$signup_meta );

    wpmu_signup_blog( $domain,
$path, $blog_title,
$user_name, $user_email, $meta
);
    confirm_blog_signup(
$domain, $path, $blog_title,
$user_name, $user_email, $meta
);
    return true;
}

/**
 * Shows a message confirming
that the new site has been
registered and is awaiting
activation.
 *
 * @since MU (3.0.0)
 *
 * @param string $domain
The domain or subdomain of the
site.
 * @param string $path
The path of the site.
 * @param string $blog_title
The title of the new site.
 * @param string $user_name
 The user's username.
 * @param string $user_email
The user's email address.
 * @param array  $meta
Any additional meta from the
{@see 'add_signup_meta'}
filter in
validate_blog_signup().
 */
function confirm_blog_signup(
$domain, $path, $blog_title,
$user_name = '', $user_email =
'', $meta = array() ) {
    ?>
    <h2>
    <?php
    /* translators: %s: Site
address. */
    printf( __(
'Congratulations! Your new
site, %s, is almost ready.' ),
"<a
href='http://{$domain}{$path}'
>{$blog_title}</a>" )
    ?>
    </h2>

    <p><?php _e( 'But, before
you can start using your site,
<strong>you must activate
it</strong>.' ); ?></p>
    <p>
    <?php
    /* translators: %s: The
user email address. */
    printf( __( 'Check your
inbox at %s and click on the
given link.' ), '<strong>' .
$user_email . '</strong>' );
    ?>
    </p>
    <p><?php _e( 'If you do
not activate your site within
two days, you will have to
sign up again.' ); ?></p>
    <h2><?php _e( 'Still
waiting for your email?' );
?></h2>
    <p><?php _e( 'If you have
not received your email yet,
there are a number of things
you can do:' ); ?></p>
    <ul id="noemail-tips">
        <li><p><strong><?php
_e( 'Wait a little longer.
Sometimes delivery of email
can be delayed by processes
outside of our control.' );
?></strong></p></li>
        <li><p><?php _e(
'Check the junk or spam folder
of your email client. Sometime
```

```
emails wind up there by
mistake.' ); ?></p></li>
        <li>
        <?php
            /* translators:
%s: Email address. */
            printf( __( 'Have
you entered your email
correctly? You have entered
%s, if it&#8217;s incorrect,
you will not receive your
email.' ), $user_email );
        ?>
        </li>
    </ul>
    <?php
    /** This action is
documented in wp-signup.php */
    do_action(
'signup_finished' );
}

/**
 * Retrieves languages
available during the site/user
sign-up process.
 *
 * @since 4.4.0
 *
 * @see
get_available_languages()
 *
 * @return string[] Array of
available language codes.
Language codes are formed by
 *                  stripping
the .mo extension from the
language file names.
 */
function
signup_get_available_languages
() {
    /**
     * Filters the list of
available languages for front-
end site sign-ups.
     *
     * Passing an empty array
to this hook will disable
output of the setting on the
     * sign-up form, and the
default language will be used
when creating the site.
     *
     * Languages not already
installed will be stripped.
     *
     * @since 4.4.0
     *
     * @param string[]
$languages Array of available
language codes. Language codes
are formed by
     *
   stripping the .mo extension
from the language file names.
     */
    $languages = (array)
apply_filters(
'signup_get_available_language
s', get_available_languages()
);

    /*
     * Strip any non-installed
languages and return.
     *
     * Re-call
get_available_languages() here
in case a language pack was
installed
     * in a callback hooked to
the
'signup_get_available_language
s' filter before this point.
     */
    return
array_intersect_assoc(
$languages,
get_available_languages() );
}

// Main.
$active_signup =
get_site_option(
'registration', 'none' );

/**
 * Filters the type of site
sign-up.
 *
 * @since 3.0.0
```

```
 *
 * @param string
$active_signup String that
returns registration type. The
value can be
 *
 'all', 'none', 'blog', or
'user'.
 */
$active_signup =
apply_filters(
'wpmu_active_signup',
$active_signup );

if ( current_user_can(
'manage_network' ) ) {
    echo '<div
class="mu_alert">';
    _e( 'Greetings Network
Administrator!' );
    echo ' ';

    switch ( $active_signup )
{
        case 'none':
            _e( 'The network
currently disallows
registrations.' );
            break;
        case 'blog':
            _e( 'The network
currently allows site
registrations.' );
            break;
        case 'user':
            _e( 'The network
currently allows user
registrations.' );
            break;
        default:
            _e( 'The network
currently allows both site and
user registrations.' );
            break;
    }

    echo ' ';

    /* translators: %s: URL to
Network Settings screen. */
    printf( __( 'To change or
disable registration go to
your <a href="%s">Options
page</a>.' ), esc_url(
network_admin_url(
'settings.php' ) ) );
    echo '</div>';
}

$newblogname = isset(
$_GET['new'] ) ? strtolower(
preg_replace( '/^-|-$|[^-a-zA-
Z0-9]/', '', $_GET['new'] ) )
: null;

$current_user =
wp_get_current_user();
if ( 'none' === $active_signup
) {
    _e( 'Registration has been
disabled.' );
} elseif ( 'blog' ===
$active_signup && !
is_user_logged_in() ) {
    $login_url = wp_login_url(
network_site_url( 'wp-
signup.php' ) );
    /* translators: %s: Login
URL. */
    printf( __( 'You must
first <a href="%s">log in</a>,
and then you can create a new
site.' ), $login_url );
} else {
    $stage = isset(
$_POST['stage'] ) ?
$_POST['stage'] : 'default';
    switch ( $stage ) {
        case 'validate-user-
signup':
            if ( 'all' ===
$active_signup
                || ( 'blog'
=== $_POST['signup_for'] &&
'blog' === $active_signup )
                || ( 'user'
=== $_POST['signup_for'] &&
'user' === $active_signup )
            ) {

validate_user_signup();
            } else {
```

```
                _e( 'User
registration has been
disabled.' );
            }
            break;
        case 'validate-blog-
signup':
            if ( 'all' ===
$active_signup || 'blog' ===
$active_signup ) {

validate_blog_signup();
            } else {
                _e( 'Site
registration has been
disabled.' );
            }
            break;
        case
'gimmeanotherblog':

validate_another_blog_signup()
;
            break;
        case 'default':
        default:
            $user_email =
isset( $_POST['user_email'] )
? $_POST['user_email'] : '';
            /**
             * Fires when the
site sign-up form is sent.
             *
             * @since 3.0.0
             */
            do_action(
'preprocess_signup_form' );
            if (
is_user_logged_in() && ( 'all'
=== $active_signup || 'blog'
=== $active_signup ) ) {

signup_another_blog(
$newblogname );
            } elseif ( !
is_user_logged_in() && ( 'all'
=== $active_signup || 'user'
=== $active_signup ) ) {
                signup_user(
$newblogname, $user_email );
            } elseif ( !
is_user_logged_in() && (
'blog' === $active_signup ) )
{
                _e( 'Sorry,
new registrations are not
allowed at this time.' );
            } else {
                _e( 'You are
logged in already. No need to
register again!' );
            }

            if ( $newblogname
) {
                $newblog =
get_blogaddress_by_name(
$newblogname );

                if ( 'blog'
=== $active_signup || 'all'
=== $active_signup ) {
                    printf(
                        /*
translators: %s: Site address.
*/
                        '<p>'
. __( 'The site you were
looking for, %s, does not
exist, but you can create it
now!' ) . '</p>',

'<strong>' . $newblog .
'</strong>'
                    );
                } else {
                    printf(
                        /*
translators: %s: Site address.
*/
                        '<p>'
. __( 'The site you were
looking for, %s, does not
exist.' ) . '</p>',

'<strong>' . $newblog .
'</strong>'
                    );
                }
            }
            break;
    }
}
?>
```

```
</div>
</div>
<?php
/**
 * Fires after the sign-up
forms, before wp_footer.
 *
 * @since 3.0.0
 */
do_action( 'after_signup_form'
);
?>

<?php
get_footer( 'wp-signup' );

<?php
/**
 * Handle Trackbacks and
Pingbacks Sent to WordPress
 *
 * @since 0.71
 *
 * @package WordPress
 * @subpackage Trackbacks
 */

if ( empty( $wp ) ) {
    require_once __DIR__ .
'/wp-load.php';
    wp( array( 'tb' => '1' )
);
}

// Always run as an
unauthenticated user.
wp_set_current_user( 0 );

/**
 * Response to a trackback.
 *
 * Responds with an error or
success XML message.
 *
 * @since 0.71
 *
 * @param int|bool $error
    Whether there was an
error.
 *
   Default '0'. Accepts '0' or
'1', true or false.
 * @param string
$error_message Error message
if an error occurred. Default
empty string.
 */
function trackback_response(
$error = 0, $error_message =
'' ) {
    header( 'Content-Type:
text/xml; charset=' .
get_option( 'blog_charset' )
);

    if ( $error ) {
        echo '<?xml
version="1.0" encoding="utf-
8"?' . ">\n";
        echo "<response>\n";
        echo
"<error>1</error>\n";
        echo
"<message>$error_message</mess
age>\n";
        echo '</response>';
        die();
    } else {
        echo '<?xml
version="1.0" encoding="utf-
8"?' . ">\n";
        echo "<response>\n";
        echo
"<error>0</error>\n";
        echo '</response>';
    }
}

if ( ! isset( $_GET['tb_id'] )
|| ! $_GET['tb_id'] ) {
    $post_id = explode( '/',
$_SERVER['REQUEST_URI'] );
    $post_id = (int) $post_id[
count( $post_id ) - 1 ];
}

$trackback_url = isset(
$_POST['url'] ) ?
$_POST['url'] : '';
$charset       = isset(
$_POST['charset'] ) ?
$_POST['charset'] : '';
```

```
// These three are
stripslashed here so they can
be properly escaped after
mb_convert_encoding().
$title     = isset(
$_POST['title'] ) ?
wp_unslash( $_POST['title'] )
: '';
$excerpt   = isset(
$_POST['excerpt'] ) ?
wp_unslash( $_POST['excerpt']
) : '';
$blog_name = isset(
$_POST['blog_name'] ) ?
wp_unslash(
$_POST['blog_name'] ) : '';

if ( $charset ) {
    $charset = str_replace(
array( ',', ' ' ), '',
strtoupper( trim( $charset ) )
);
} else {
    $charset = 'ASCII, UTF-8,
ISO-8859-1, JIS, EUC-JP,
SJIS';
}

// No valid uses for UTF-7.
if ( str_contains( $charset,
'UTF-7' ) ) {
    die;
}

// For international
trackbacks.
if ( function_exists(
'mb_convert_encoding' ) ) {
    $title     =
mb_convert_encoding( $title,
get_option( 'blog_charset' ),
$charset );
    $excerpt   =
mb_convert_encoding( $excerpt,
get_option( 'blog_charset' ),
$charset );
    $blog_name =
mb_convert_encoding(
$blog_name, get_option(
'blog_charset' ), $charset );
}

// Escape values to use in the
trackback.
$title     = wp_slash( $title
);
$excerpt   = wp_slash(
$excerpt );
$blog_name = wp_slash(
$blog_name );

if ( is_single() || is_page()
) {
    $post_id = $posts[0]->ID;
}

if ( ! isset( $post_id ) || !
(int) $post_id ) {
    trackback_response( 1, __(
'I really need an ID for this
to work.' ) );
}

if ( empty( $title ) && empty(
$trackback_url ) && empty(
$blog_name ) ) {
    // If it doesn't look like
a trackback at all.
    wp_redirect(
get_permalink( $post_id ) );
    exit;
}

if ( ! empty( $trackback_url )
&& ! empty( $title ) ) {
    /**
     * Fires before the
trackback is added to a post.
     *
     * @since 4.7.0
     *
     * @param int    $post_id
      Post ID related to the
trackback.
     * @param string
$trackback_url Trackback URL.
     * @param string $charset
      Character set.
     * @param string $title
      Trackback title.
     * @param string $excerpt
      Trackback excerpt.
     * @param string
$blog_name     Site name.
```

```
     */
    do_action(
'pre_trackback_post',
$post_id, $trackback_url,
$charset, $title, $excerpt,
$blog_name );

    header( 'Content-Type:
text/xml; charset=' .
get_option( 'blog_charset' )
);

    if ( ! pings_open(
$post_id ) ) {
        trackback_response( 1,
__( 'Sorry, trackbacks are
closed for this item.' ) );
    }

    $title   =
wp_html_excerpt( $title, 250,
'&#8230;' );
    $excerpt =
wp_html_excerpt( $excerpt,
252, '&#8230;' );

    $comment_post_id      =
(int) $post_id;
    $comment_author       =
$blog_name;
    $comment_author_email =
'';
    $comment_author_url   =
$trackback_url;
    $comment_content      =
"<strong>$title</strong>\n\n$e
xcerpt";
    $comment_type         =
'trackback';

    $dupe = $wpdb-
>get_results(
        $wpdb->prepare(
            "SELECT * FROM
$wpdb->comments WHERE
comment_post_ID = %d AND
comment_author_url = %s",
            $comment_post_id,

$comment_author_url
        )
    );

    if ( $dupe ) {
        trackback_response( 1,
__( 'There is already a ping
from that URL for this post.'
) );
    }

    $commentdata = array(
        'comment_post_ID' =>
$comment_post_id,
    );

    $commentdata += compact(
        'comment_author',

'comment_author_email',
        'comment_author_url',
        'comment_content',
        'comment_type'
    );

    $result = wp_new_comment(
$commentdata );

    if ( is_wp_error( $result
) ) {
        trackback_response( 1,
$result->get_error_message()
);
    }

    $trackback_id = $wpdb-
>insert_id;

    /**
     * Fires after a trackback
is added to a post.
     *
     * @since 1.2.0
     *
     * @param int
$trackback_id Trackback ID.
     */
    do_action(
'trackback_post',
$trackback_id );

    trackback_response( 0 );
}
```

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

HASIL

MAHASISWA : HARDIMAN	Pembimbing I : Hj. A. Irmayani Pawelloi, S. T., M.
NIM : 218280139	Pembimbing II : A. Wafiah, S.Kom., M.Kom
Judul Skripsi : SISTEM INFORMASI SARUNG TENUN TRADISIONAL MAMASA BERBASIS WEB	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 Buka Tombol pesan Tambahkan harga	6/8/2024 A+/	Konsultasi 1 BAB IV Tambahkan detail sistem dan disebutkan setiap halaman	A.
Konsultasi 2 ABSTRAK	A+/	Konsultasi 2 Format penulisan sesuai dgn Pedoman	A.
Konsultasi 3 Kesimpulan menjawab Rumusan Masalah	A+/	Konsultasi 3 BAB V disesuaikan dgn Pedoman	A
Konsultasi 4 ACC	15/8/24 A+/	Konsultasi 4 Lengkapi Lampiran data Penelitian	A.
Konsultasi 5		Konsultasi 5 ACC Ujian Hasil	A

Lanjut ke halaman sebelah...

Perhatian :

- Mahasiswa wajib konsultasi minimal 5 kali
- Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
- Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
- Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

Lanjutan...

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TG PARA PEMBIM
Konsultasi 6		Konsultasi 6	
Konsultasi 7		Konsultasi 7	
Konsultasi 8		Konsultasi 8	
Konsultasi 9		Konsultasi 9	
Konsultasi 10		Konsultasi 10	

Parepare, 30 Juli 2024

Mengetahui
Ketua Program Studi

Marlina, S.Kom.,M.Kom.
NBM. 1162 680

Mahasiswa

HARDIMAN
NIM. 218280139

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

SKRIPSI

AHASISWA : HARDIMAN	Pembimbing I : Hj. A. Irmayani Pawelloi, S. T., M.
IM : 218280139	Pembimbing II : A. Wafiah, S.Kom., M.Kom
udul Skripsi : SISTEM INFORMASI SARUNG TENUN TRADISIONAL MAMASA BERBASIS WEB	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
onsultasi 1 Acc ujian tutup	7/9/24	Konsultasi 1 Acc ujian Tutup	
onsultasi 2		Konsultasi 2	
onsultasi 3		Konsultasi 3	
onsultasi 4		Konsultasi 4	
nsultasi 5		Konsultasi 5	

Lanjut ke halaman sebelah...

hatian :

- Mahasiswa wajib konsultasi minimal 5 kali
- Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
- Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
- Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

Lanjutan...

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TG PARA PEMBIME
Konsultasi 6		Konsultasi 6	
Konsultasi 7		Konsultasi 7	
Konsultasi 8		Konsultasi 8	
Konsultasi 9		Konsultasi 9	
Konsultasi 10		Konsultasi 10	

Parepare, 05 September 2024

Mengetahui
Ketua Program Studi

Marlina, S.Kom.,M.Kom.
NBM. 1162 680

Mahasiswa

HARDIMAN
NIM. **218280139**

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik