GitHub - PHPMailer/PHPMailer: The classic email sending library for PHP

By Synchro

Archived: 2026-04-06 03:31:29 UTC

PHPMailer – A full-featured email creation and transfer class for PHP

Features

- Probably the world's most popular code for sending email from PHP!
- Used by many open-source projects: WordPress, Drupal, 1CRM, SugarCRM, Yii, Joomla! and many more
- Integrated SMTP support – send without a local mail server
- Send emails with multiple To, CC, BCC, and Reply-to addresses
- Multipart/alternative emails for mail clients that do not read HTML email
- Add attachments, including inline
- Support for UTF-8 content and 8bit, base64, binary, and quoted-printable encodings
- Full UTF-8 support when using servers that support `SMTPUTF8`.
- Support for iCal events in multiparts and attachments
- SMTP authentication with `LOGIN`, `PLAIN`, `CRAM-MD5`, and `XOAUTH2` mechanisms over SMTPS and SMTP+STARTTLS transports
- Validates email addresses automatically
- Protects against header injection attacks
- Error messages in over 50 languages!
- DKIM and S/MIME signing support
- Compatible with PHP 5.5 and later, including PHP 8.5

- Namespaced to prevent name clashes
- Much more!

Why you might need it

Many PHP developers need to send email from their code. The only PHP function that supports this directly is `mail()` . However, it does not provide any assistance for making use of popular features such as authentication, HTML messages, and attachments.

Formatting email correctly is surprisingly difficult. There are myriad overlapping (and conflicting) standards, requiring tight adherence to horribly complicated formatting and encoding rules – the vast majority of code that you'll find online that uses the `mail()` function directly is just plain wrong, if not unsafe!

The PHP `mail()` function usually sends via a local mail server, typically fronted by a `sendmail` binary on Linux, BSD, and macOS platforms, however, Windows usually doesn't include a local mail server; PHPMailer's integrated SMTP client allows email sending on all platforms without needing a local mail server. Be aware though, that the `mail()` function should be avoided when possible; it's both faster and safer to use SMTP to localhost.

Please don't be tempted to do it yourself – if you don't use PHPMailer, there are many other excellent libraries that you should look at before rolling your own. Try Symfony Mailer, Laminas/Mail, ZetaComponents, etc.

License

This software is distributed under the LGPL 2.1 license, along with the GPL Cooperation Commitment. Please read LICENSE for information on the software availability and distribution.

Installation & loading

PHPMailer is available on Packagist (using semantic versioning), and installation via Composer is the recommended way to install PHPMailer. Just add this line to your `composer.json` file:

```
"phpmailer/phpmailer": "^7.0.0"
```

or run

```
composer require phpmailer/phpmailer
```

Note that the `vendor` folder and the `vendor/autoload.php` script are generated by Composer; they are not part of PHPMailer.

If you want to use XOAUTH2 authentication, you will also need to add a dependency on the `league/oauth2-client` and appropriate service adapters package in your `composer.json` , or take a look at by @decomplexity's SendOauth2 wrapper, especially if you're using Microsoft services.

Alternatively, if you're not using Composer, you can download PHPMailer as a zip file, (note that docs and examples are not included in the zip file), then copy the contents of the PHPMailer folder into one of the `include_path` directories specified in your PHP configuration and load each class file manually:

```
<?php
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

require 'path/to/PHPMailer/src/Exception.php';
require 'path/to/PHPMailer/src/PHPMailer.php';
require 'path/to/PHPMailer/src/SMTP.php';
```

If you're not using the `SMTP` class explicitly (you're probably not), you don't need a `use` line for it. Even if you're not using exceptions, you do still need to load the `Exception` class as it is used internally.

Legacy versions

PHPMailer 5.2 (which is compatible with PHP 5.0 — 7.0) is no longer supported, even for security updates. You will find the latest version of 5.2 in the 5.2-stable branch. If you're using PHP 5.5 or later (which you should be), switch to the 6.x releases.

Upgrading from 5.2

The biggest changes are that source files are now in the `src/` folder, and PHPMailer now declares the namespace `PHPMailer\PHPMailer`. This has several important effects – read the upgrade guide for more details.

Minimal installation

While installing the entire package manually or with Composer is simple, convenient, and reliable, you may want to include only vital files in your project. At the very least you will need src/PHPMailer.php. If you're using SMTP, you'll need src/SMTP.php, and if you're using POP-before SMTP (*very* unlikely!), you'll need src/POP3.php. You can skip the language folder if you're not showing errors to users and can make do with English-only errors. If you're using XOAUTH2 you will need src/OAuth.php as well as the Composer dependencies for the services you wish to authenticate with. Really, it's much easier to use Composer!

A Simple Example

```
<?php
//Import PHPMailer classes into the global namespace
//These must be at the top of your script, not inside a function
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\SMTP;
use PHPMailer\PHPMailer\Exception;

//Load Composer's autoloader (created by composer, not included with PHPMailer)
```

```
require 'vendor/autoload.php';

//Create an instance; passing `true` enables exceptions
$mail = new PHPMailer(true);

try {
    //Server settings
    $mail->SMTPDebug = SMTP::DEBUG_SERVER;                      //Enable verbose debug output
    $mail->isSMTP();                                            //Send using SMTP
    $mail->Host       = 'smtp.example.com';                     //Set the SMTP server to send throug
    $mail->SMTPAuth   = true;                                   //Enable SMTP authentication
    $mail->Username   = 'user@example.com';                     //SMTP username
    $mail->Password   = 'secret';                               //SMTP password
    $mail->SMTPSecure = PHPMailer::ENCRYPTION_SMTPS;            //Enable implicit TLS encryption
    $mail->Port       = 465;                                    //TCP port to connect to; use 587 if

    //Recipients
    $mail->setFrom('from@example.com', 'Mailer');
    $mail->addAddress('joe@example.net', 'Joe User');     //Add a recipient
    $mail->addAddress('ellen@example.com');               //Name is optional
    $mail->addReplyTo('info@example.com', 'Information');
    $mail->addCC('cc@example.com');
    $mail->addBCC('bcc@example.com');

    //Attachments
    $mail->addAttachment('/var/tmp/file.tar.gz');         //Add attachments
    $mail->addAttachment('/tmp/image.jpg', 'new.jpg');    //Optional name

    //Content
    $mail->isHTML(true);                                  //Set email format to HTML
    $mail->Subject = 'Here is the subject';
    $mail->Body    = 'This is the HTML message body <b>in bold!</b>';
    $mail->AltBody = 'This is the body in plain text for non-HTML mail clients';

    $mail->send();
    echo 'Message has been sent';
} catch (Exception $e) {
    echo "Message could not be sent. Mailer Error: {$mail->ErrorInfo}";
}
```

You'll find plenty to play with in the examples folder, which covers many common scenarios including sending through Gmail, building contact forms, sending to mailing lists, and more.

If you are re-using the instance (e.g. when sending to a mailing list), you may need to clear the recipient list to avoid sending duplicate messages. See the mailing list example for further guidance.

That's it. You should now be ready to use PHPMailer!

Localization

PHPMailer defaults to English, but in the language folder, you'll find many translations for PHPMailer error messages that you may encounter. Their filenames contain ISO 639-1 language code for the translations, for example `fr` for French. To specify a language, you need to tell PHPMailer which one to use, like this:

```
//To load the French version
$mail->setLanguage('fr', '/optional/path/to/language/directory/');
```

We welcome corrections and new languages – if you're looking for corrections, run the Language/TranslationCompletenessTest.php script in the tests folder and it will show any missing translations.

Documentation

Start reading at the GitHub wiki. If you're having trouble, head for the troubleshooting guide as it's frequently updated.

Examples of how to use PHPMailer for common scenarios can be found in the examples folder. If you're looking for a good starting point, we recommend you start with the Gmail example.

To reduce PHPMailer's deployed code footprint, examples are not included if you load PHPMailer via Composer or via GitHub's zip file download, so you'll need to either clone the git repository or use the above links to get to the examples directly.

Complete generated API documentation is available online.

You can generate complete API-level documentation by running `phpdoc` in the top-level folder, and documentation will appear in the `docs` folder, though you'll need to have PHPDocumentor installed. You may find the unit tests a good reference for how to do various operations such as encryption.

If the documentation doesn't cover what you need, search the many questions on Stack Overflow, and before you ask a question about "SMTP Error: Could not connect to SMTP host.", read the troubleshooting guide.

Tests

PHPMailer tests use PHPUnit 9, with a polyfill to let 9-style tests run on older PHPUnit and PHP versions.

If this isn't passing, is there something you can do to help?

Security

Please disclose any vulnerabilities found responsibly – report security issues to the maintainers privately.

See SECURITY and PHPMailer's security advisories on GitHub.

Contributing

Please submit bug reports, suggestions, and pull requests to the GitHub issue tracker.

We're particularly interested in fixing edge cases, expanding test coverage, and updating translations.

If you found a mistake in the docs, or want to add something, go ahead and amend the wiki – anyone can edit it.

If you have git clones from prior to the move to the PHPMailer GitHub organisation, you'll need to update any remote URLs referencing the old GitHub location with a command like this from within your clone:

```
git remote set-url upstream https://github.com/PHPMailer/PHPMailer.git
```

Please *don't* use the SourceForge or Google Code projects any more; they are obsolete and no longer maintained.

Sponsorship

Development time and resources for PHPMailer are provided by Smartmessages.net, the world's only privacy-first email marketing system.

Donations are very welcome, whether in beer 🍺, T-shirts 👕, or cold, hard cash 💰. Sponsorship through GitHub is a simple and convenient way to say "thank you" to PHPMailer's maintainers and contributors – just click the "Sponsor" button on the project page. If your company uses PHPMailer, consider taking part in Tidelift's enterprise support programme.

PHPMailer For Enterprise

Available as part of the Tidelift Subscription.

The maintainers of PHPMailer and thousands of other packages are working with Tidelift to deliver commercial support and maintenance for the open-source packages you use to build your applications. Save time, reduce risk, and improve code health, while paying the maintainers of the exact packages you use. Learn more.

Changelog

See changelog.

History

- PHPMailer was originally written in 2001 by Brent R. Matzelle as a SourceForge project.
- Marcus Bointon (`coolbru` on SF) and Andy Prevost (`codeworxtech`) took over the project in 2004.
- Became an Apache incubator project on Google Code in 2010, managed by Jim Jagielski.

- Marcus created his fork on GitHub in 2008.
- Jim and Marcus decide to join forces and use GitHub as the canonical and official repo for PHPMailer in 2013.
- PHPMailer moves to the PHPMailer organisation on GitHub in 2013.

What's changed since moving from SourceForge?

- Official successor to the SourceForge and Google Code projects.
- Test suite.
- Continuous integration with GitHub Actions.
- Composer support.
- Public development.
- Additional languages and language strings.
- CRAM-MD5 authentication support.
- Preserves full repo history of authors, commits, and branches from the original SourceForge project.
