

Sr No	Name
1.	Prerequisites
2.	PHPMailer Workflow Overview
3.	Installation Quick Reference
4.	Core Architecture
5.	Minimal Working Example
6.	Transport Methods
7.	Basic Configuration Concepts
8.	Key Method Summary
9.	Core Feature Categories
10.	Feature Integration Flow
11.	Key Properties and Methods Reference
12.	Feature Support Matrix
13.	Core Architecture
14.	Authentication and Security
15.	Advanced Features

1. Prerequisites

PHPMailer requires the following minimum environment:

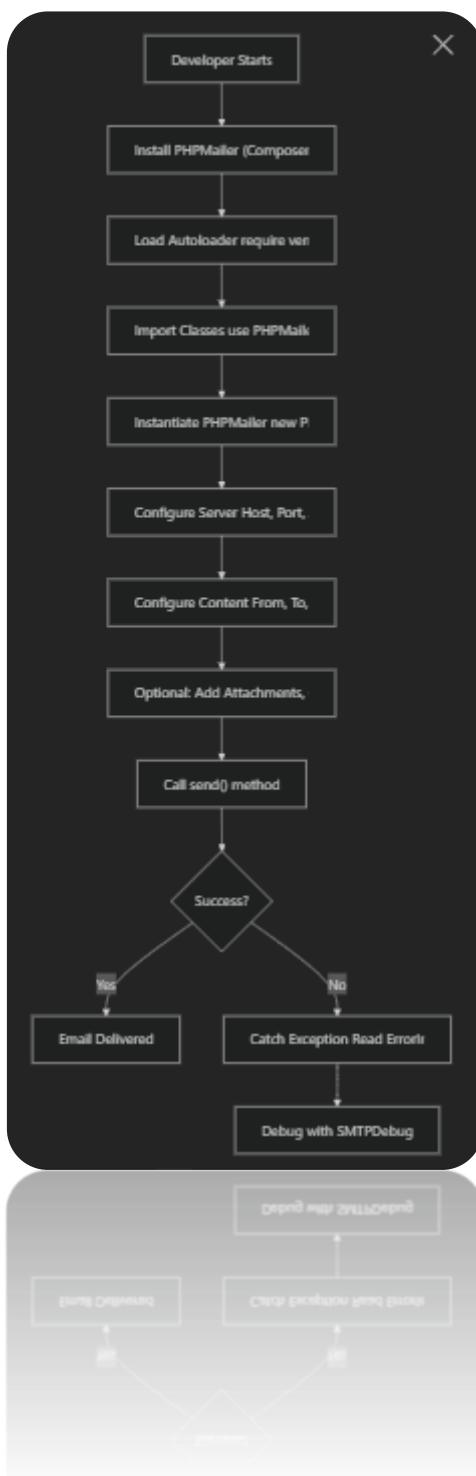
<i>Requirement</i>	<i>Minimum Version</i>	<i>Purpose</i>
<i>PHP</i>	5.5.0	Core runtime
<i>ext-ctype</i>	*	Character type checking
<i>ext-filter</i>	*	Input validation
<i>ext-hash</i>	*	Cryptographic operations

Recommended extensions:

- ext-openssl - Required for secure SMTP (TLS/SSL) and DKIM signing
- ext-mbstring - Required for multibyte character encoding (UTF-8)
- ext-imap - Required for advanced email address parsing per RFC822

2.PHPMailer Workflow

Overview



3. Installation Quick Reference

PHPMailer supports two installation methods:

3.1. Composer Installation (Recommended)

```
composer require phpmailer/phpmailer
```

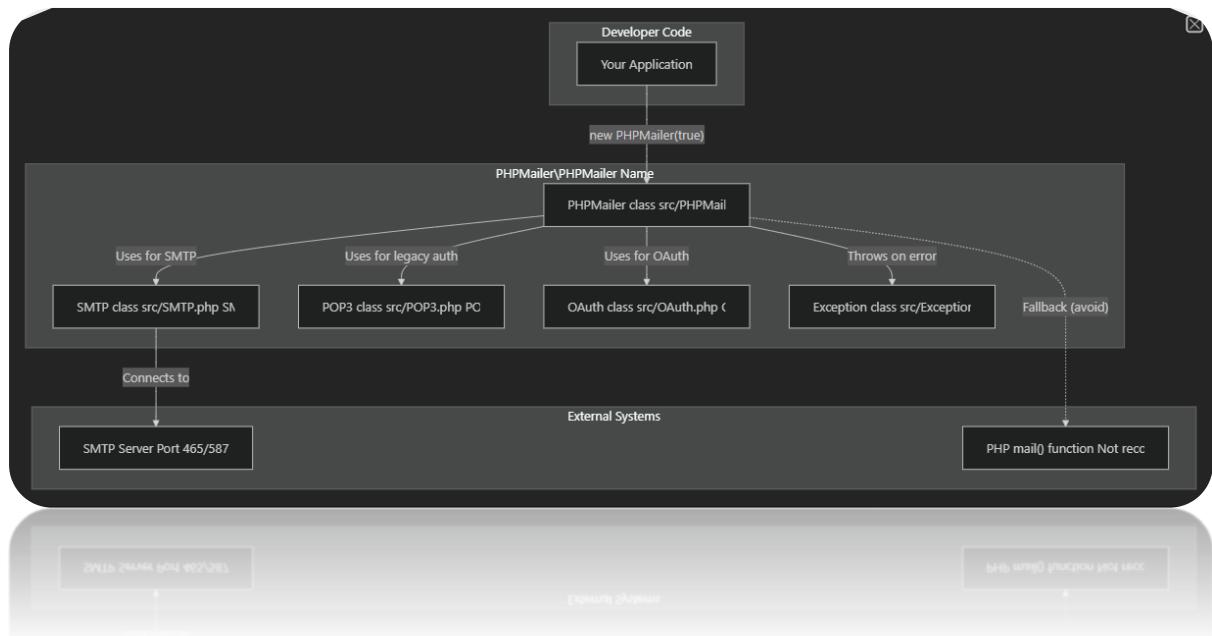
This creates the autoloader at vendor/autoload.php and installs PHPMailer under the PHPMailer\PHPMailer\ namespace.

3.2. Manual Installation

Download the library and manually require the source files:

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

4.Core Architecture

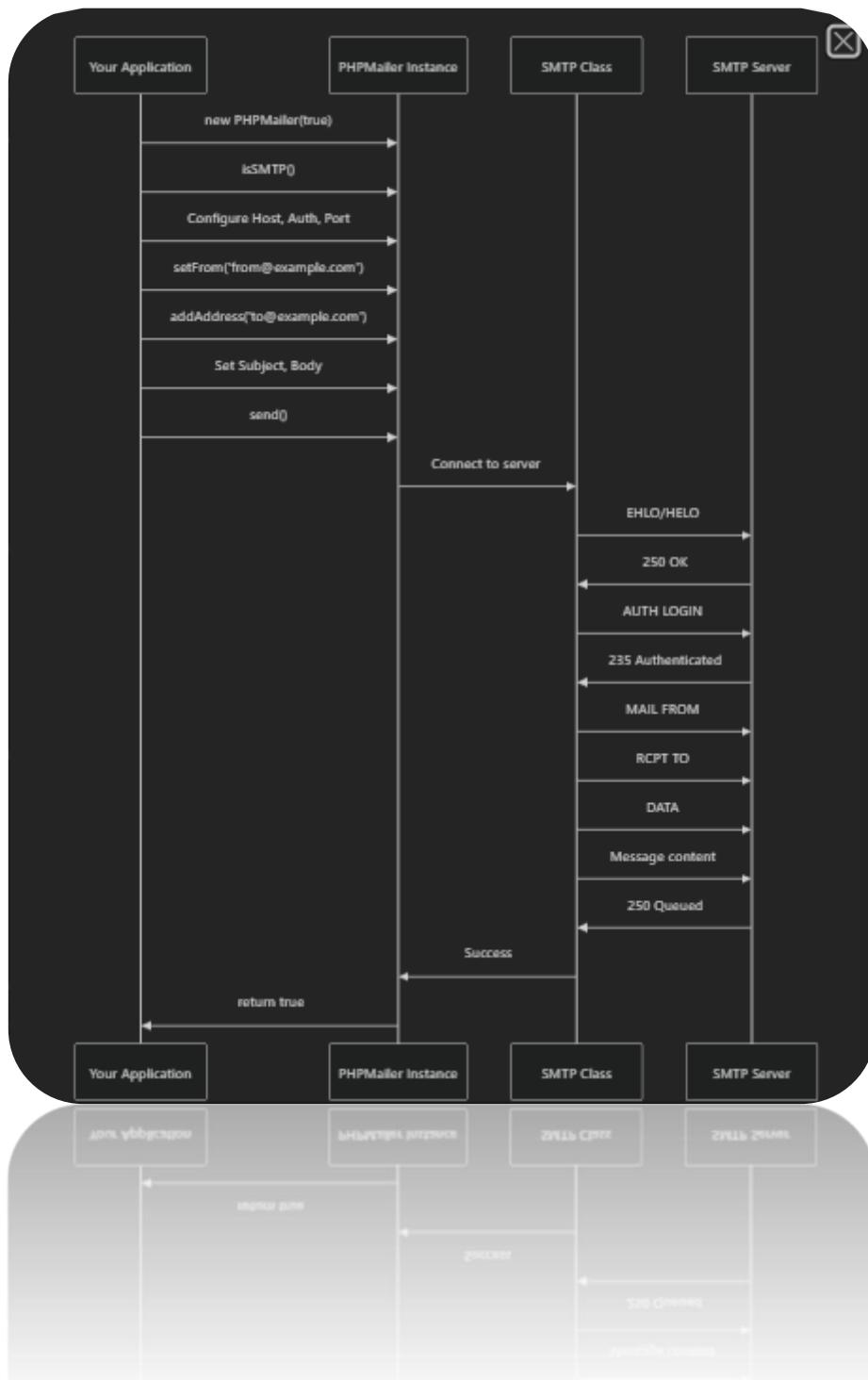


Key class responsibilities:

<i>Class</i>	<i>Location</i>	<i>Primary Responsibility</i>
<i>PHPMailer</i>	src/PHPMailer.php	Email composition, validation, and sending orchestration
<i>SMTP</i>	src/SMTP.php	Low-level SMTP protocol implementation and connection management
<i>OAuth</i>	src/OAuth.php	OAuth2 XOAUTH2 authentication mechanism
<i>POP3</i>	src/POP3.php	POP-before-SMTP authentication (legacy pattern)
<i>Exception</i>	src/Exception.php	PHPMailer-specific exception type for error handling

5. Minimal Working Example

The following demonstrates the minimum code required to send an email via SMTP:



The complete example is shown here (without error handling for brevity):

```
use PHPMailer\PHPMailer\PHPMailer;
require 'vendor/autoload.php';

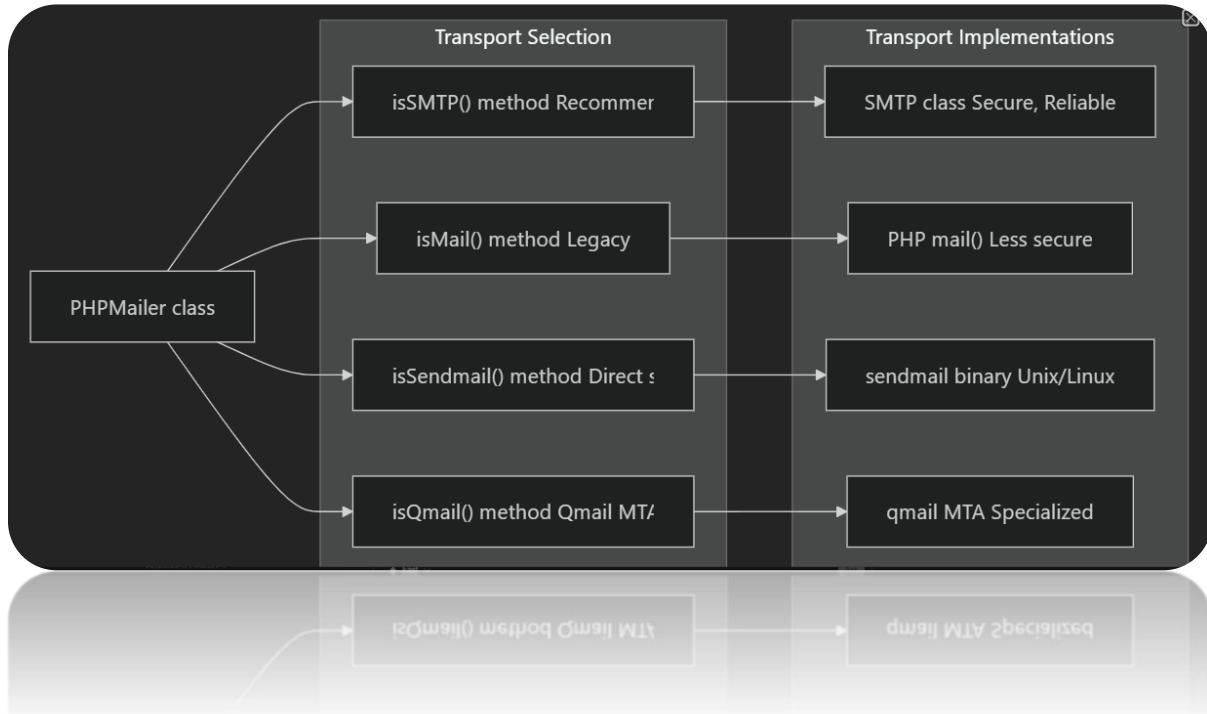
$mail = new PHPMailer(true);
$mail->isSMTP();
$mail->Host = 'smtp.example.com';
$mail->SMTPAuth = true;
$mail->Username = 'user@example.com';
$mail->Password = 'secret';
$mail->SMTPSecure = PHPMailer::ENCRYPTION_SMTPS;
$mail->Port = 465;

$mail->setFrom('from@example.com', 'Mailer');
$mail->addAddress('joe@example.net', 'Joe User');
$mail->Subject = 'Here is the subject';
$mail->Body = 'This is the message body';

$mail->send();
```

6.Transport Methods

PHPMailer supports multiple email transport methods:



Recommended approach: Use `isSMTP()` with proper authentication and encryption. The `PHP mail()` function has security limitations and should be avoided when possible.

For detailed SMTP configuration including encryption options (SMTPLS vs STARTTLS) and authentication methods, see [SMTP Transport](#). For OAuth2 authentication setup, see [OAuth2 Setup](#).

7. Basic Configuration Concepts

After instantiating PHPMailer, you configure it through public properties and methods:

<i>Configuration Area</i>	<i>Key Properties/Methods</i>	<i>Example</i>
<i>Server Settings</i>	Host, Port, SMTPAuth, SMTPSecure	\$mail->Host = 'smtp.gmail.com'
<i>Authentication</i>	Username, Password, or setOAuth()	\$mail->Username = 'user@example.com'
<i>Sender Identity</i>	setFrom(), addReplyTo()	\$mail->setFrom('from@example.com')
<i>Recipients</i>	addAddress(), addCC(), addBCC()	\$mail->addAddress('to@example.com')
<i>Content</i>	Subject, Body, AltBody, isHTML()	\$mail->Subject = 'Test'
<i>Attachments</i>	addAttachment(), addEmbeddedImage()	\$mail->addAttachment('/path/file.pdf')
<i>Debugging</i>	SMTPDebug, Debugoutput	\$mail->SMTPDebug = SMTP::DEBUG_SERVER

8.Key Method Summary

The following table maps common tasks to PHPMailer methods:

<i>Task</i>	<i>Method(s)</i>	<i>Description</i>
<i>Send email</i>	<code>send()</code>	Validates and transmits the email
<i>Set sender</i>	<code>setFrom(\$address, \$name)</code>	Defines From header
<i>Add recipient</i>	<code>addAddress(\$address, \$name)</code>	Adds To recipient
<i>Add CC</i>	<code>addCC(\$address, \$name)</code>	Adds CC recipient
<i>Add BCC</i>	<code>addBCC(\$address, \$name)</code>	Adds BCC recipient (hidden)
<i>Set reply-to</i>	<code>addReplyTo(\$address, \$name)</code>	Defines Reply-To header
<i>Attach file</i>	<code>addAttachment(\$path, \$name)</code>	Adds file attachment
<i>Embed image</i>	<code>addEmbeddedImage(\$path, \$cid)</code>	Embeds image for HTML email
<i>Enable SMTP</i>	<code>isSMTP()</code>	Selects SMTP transport
<i>Enable HTML</i>	<code>isHTML(\$bool)</code>	Enables HTML body format
<i>Load language</i>	<code>setLanguage(\$code, \$path)</code>	Loads localized error messages
<i>Clear recipients</i>	<code>clearAddresses()</code>	Removes all To/CC/BCC (for reuse)

9.Core Feature Categories

9.1.Transport Mechanisms

PHPMailer supports multiple email transport methods, configured through the Mailer property:

<i>Transport Method</i>	<i>Configuration Method</i>	<i>Description</i>	<i>Recommended</i>
<i>SMTP</i>	<code>isSMTP()</code>	Uses SMTP with SMTP class	protocol ✓ Yes
<i>PHP mail()</i>	<code>isMail()</code>	Uses PHP's in mail() function	built- No
<i>Sendmail</i>	<code>isSendmail()</code>	Direct sendmail binary invocation	Limited cases
<i>Qmail</i>	<code>isQmail()</code>	Qmail-specific implementation	Limited cases

9.2.Message Composition

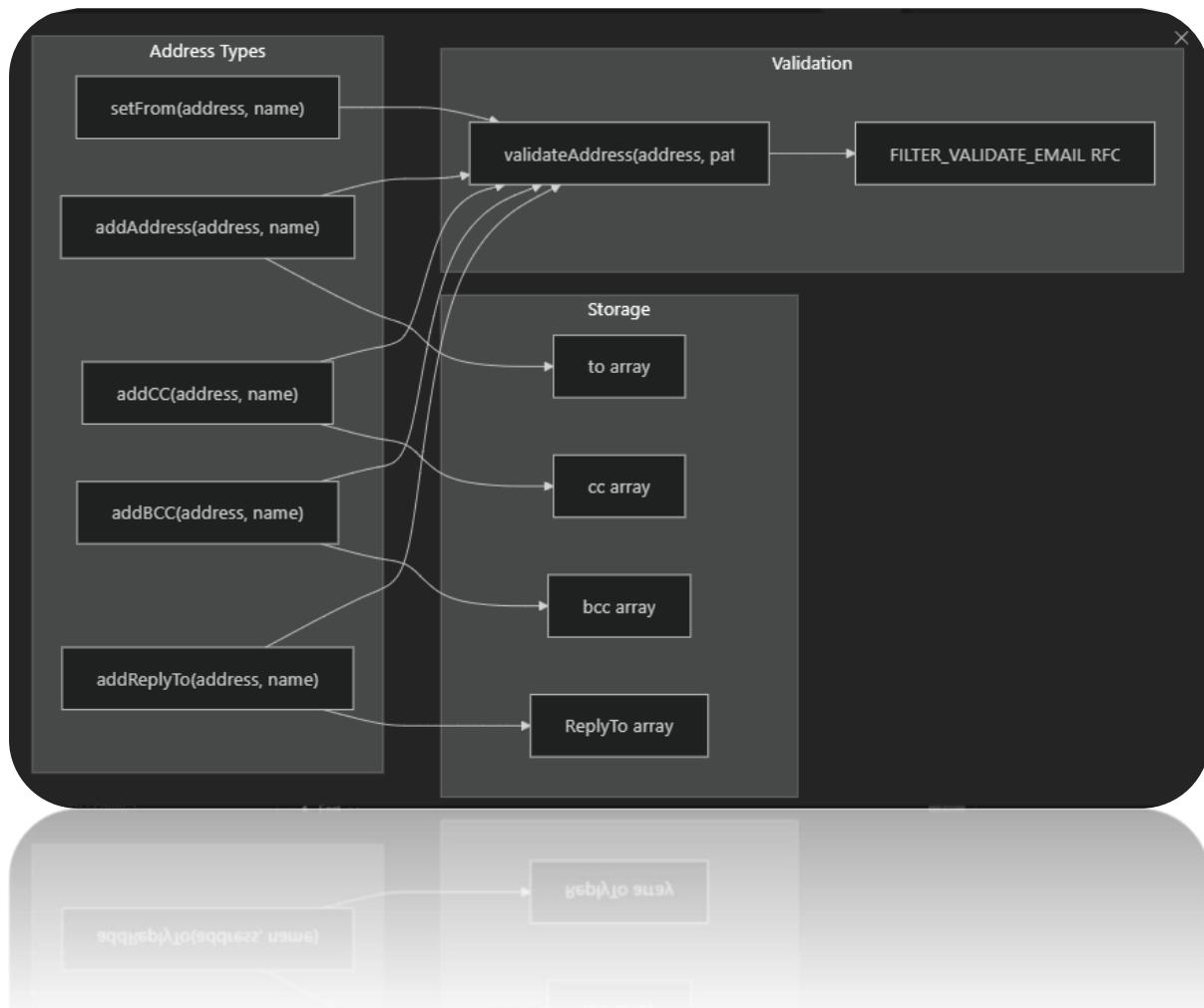
PHPMailer provides comprehensive message composition features:

<i>Feature</i>	<i>Properties/Methods</i>	<i>Description</i>
<i>HTML Content</i>	isHTML(), Body	Set HTML message body
<i>Plain Text</i>	AltBody	Alternative plain text for non-HTML clients
<i>Subject</i>	Subject	Email subject line
<i>Character Encoding</i>	CharSet	Default UTF-8 support
<i>Content Transfer Encoding</i>	Encoding	8bit, base64, binary, quoted-printable
<i>Multipart/alternative</i>	Automatic	Automatically creates multipart when both HTML and plain text provided

The `isHTML(true)` method configures the email for HTML content and enables automatic multipart/alternative generation when `AltBody` is set.

9.3.Address Management

PHPMailer supports multiple address types with validation:



All address methods support an optional name parameter for the display name. Addresses are automatically validated to prevent header injection attacks.

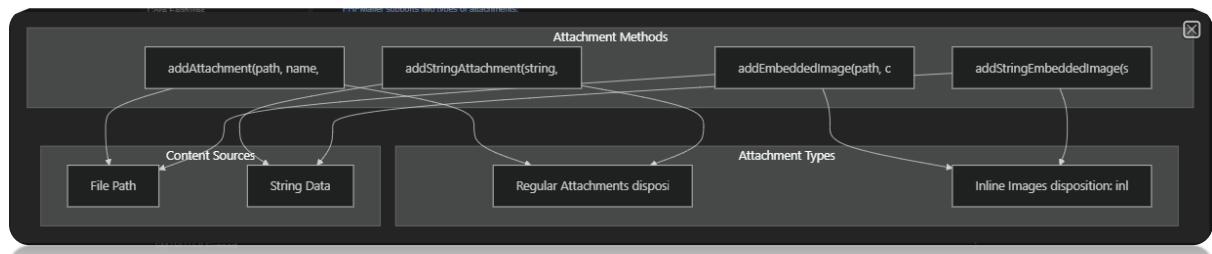
9.4. Content Encoding and Character Sets

PHPMailer provides comprehensive character set and encoding support:

<i>Property</i>	<i>Values</i>	<i>Default</i>	<i>Purpose</i>
<i>CharSet</i>	UTF-8, ISO-8859-1, etc.	UTF-8	Message character encoding
<i>Encoding</i>	8bit, 7bit, base64, binary, 8bit quoted-printable		Content transfer encoding
<i>SMTPUTF8</i>	true, false	Automatic	Enable RFC 6531 SMTPUTF8 support

9.5.Attachments and Embedded Content

PHPMailer supports two types of attachments:



Inline images are referenced in HTML content using Content-ID (CID) references like ``.

9.6.Authentication Mechanisms

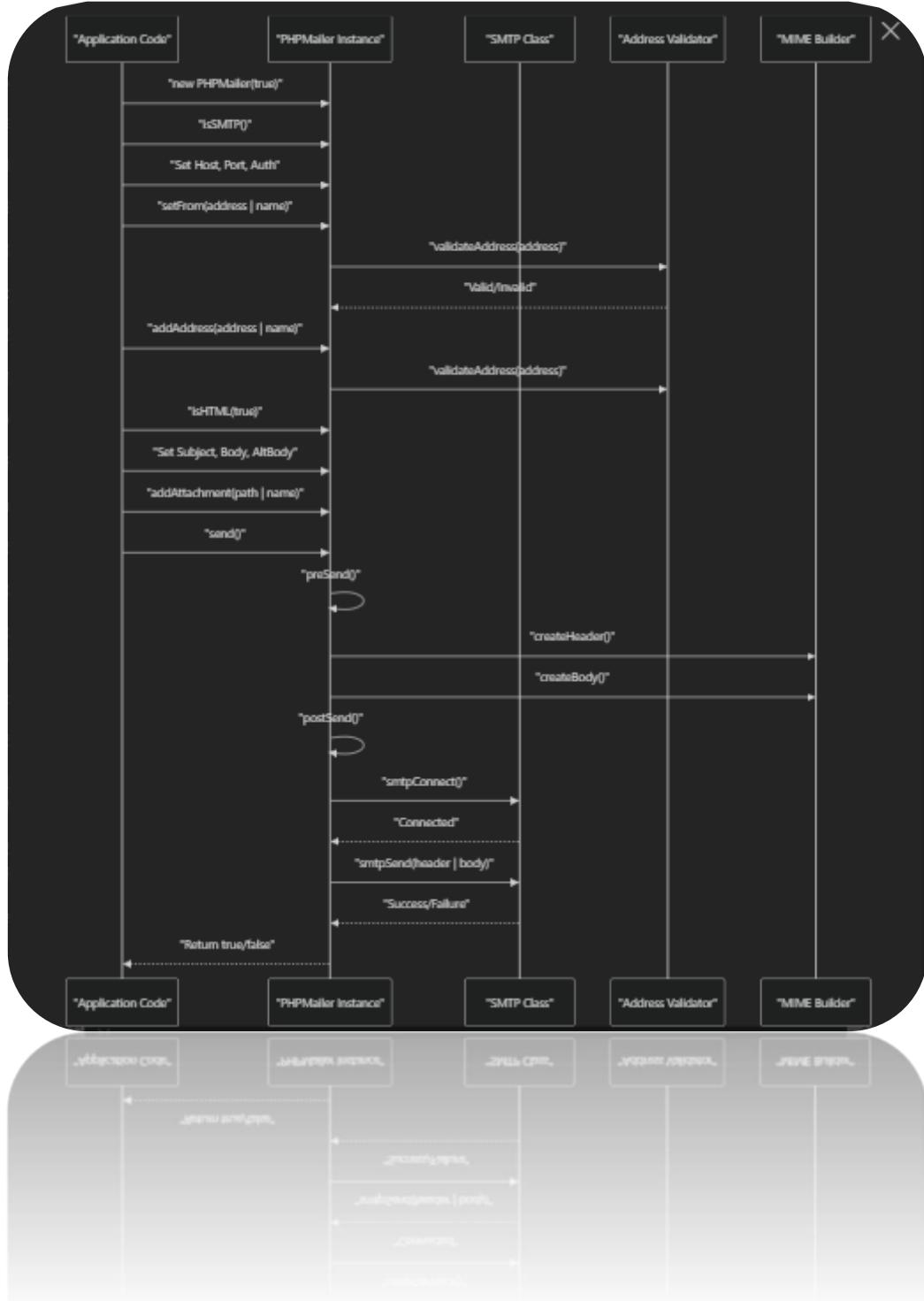
PHPMailer supports multiple SMTP authentication mechanisms:

<i>Mechanism</i>	<i>Property/Class</i>	<i>Description</i>
<i>LOGIN</i>	<code>SMTPAuth = true</code>	Basic username/password
<i>PLAIN</i>	<code>SMTPAuth = true</code>	Plain text authentication
<i>CRAM-MD5</i>	<code>SMTPAuth = true</code>	Challenge-response
<i>XOAUTH2</i>	<code>OAuth class</code>	OAuth2 token-based

The authentication mechanism is automatically negotiated with the SMTP server. For OAuth2 authentication, the OAuth class in `src/OAuth.php` integrates with the `league/oauth2-client` library.

10. Feature Integration Flow

The following diagram shows how core features integrate during the email sending process:



11.Key Properties and Methods Reference

11.1.Essential Configuration Properties

<i>Property</i>	<i>Type</i>	<i>Default</i>	<i>Purpose</i>
<i>Mailer</i>	string	'mail'	Transport method (mail, sendmail, qmail, smtp)
<i>Host</i>	string	'localhost'	SMTP server hostname(s)
<i>Port</i>	int	25	SMTP TCP port
<i>SMTPAuth</i>	bool	false	Enable SMTP authentication
<i>Username</i>	string	"	SMTP username
<i>Password</i>	string	"	SMTP password
<i>SMTPSecure</i>	string	"	Encryption type ('tls', 'ssl', or constants)
<i>SMTPAutoTLS</i>	bool	true	Automatically enable TLS encryption
<i>CharSet</i>	string	'UTF-8'	Character encoding
<i>Encoding</i>	string	'8bit'	Content transfer encoding

11.2. Message Content Properties

<i>Property</i>	<i>Type</i>	<i>Default</i>	<i>Purpose</i>
<i>Subject</i>	string	"	Email subject line
<i>Body</i>	string	"	Message body (HTML or plain text)
<i>AltBody</i>	string	"	Plain text alternative for HTML emails
<i>WordWrap</i>	int	0	Word wrap length (0 = no wrap)
<i>AllowEmpty</i>	bool	false	Allow sending empty message body
<i>ContentType</i>	string	'text/plain'	MIME Content-type

11.3.Core Methods

<i>Method</i>	<i>Parameters</i>	<i>Returns</i>	<i>Purpose</i>
<i>isSMTP()</i>	none	void	Set mailer to use SMTP
<i>isMail()</i>	none	void	Set mailer to use PHP mail()
<i>isSendmail()</i>	none	void	Set mailer to use sendmail
<i>setFrom()</i>	address, name, auto	bool	Set From address
<i>addAddress()</i>	address, name	bool	Add To recipient
<i>addCC()</i>	address, name	bool	Add CC recipient
<i>addBCC()</i>	address, name	bool	Add BCC recipient
<i>addReplyTo()</i>	address, name	bool	Add Reply-To address
<i>addAttachment()</i>	path, name, encoding, type, disposition	bool	Add file attachment
<i>isHTML()</i>	isHtml	void	Set message format to HTML or plain text
<i>send()</i>	none	bool	Send the email
<i>clearAddresses()</i>	none	void	Clear all recipients
<i>clearAttachments()</i>	none	void	Clear all attachments

12. Feature Support Matrix

The following table summarizes PHPMailer's feature support:

<i>Feature</i>	<i>Support Level</i>	<i>Requirements</i>
<i>SMTP Transport</i>	Full	None
<i>TLS/SSL Encryption</i>	Full	ext-openssl
<i>SMTP Authentication</i>	Full	None
<i>OAuth2 Authentication</i>	Full	league/oauth2-client + provider packages
<i>HTML Email</i>	Full	None
<i>Attachments</i>	Full	None
<i>Inline Images</i>	Full	None
<i>Multiple Recipients</i>	Full	None
<i>UTF-8 / Unicode</i>	Full	None
<i>SMTPUTF8</i>	Full	Server support
<i>DKIM Signing</i>	Full	ext-openssl
<i>S/MIME Signing</i>	Full	ext-openssl
<i>Multi-language Errors</i>	50+ languages	Language files in language/ directory
<i>iCalendar Events</i>	Full	None
<i>POP-before-SMTP</i>	Full	src/POP3.php

13.Core Architecture

13.1.Architectural Overview

PHPMailer follows a **modular architecture** with clear separation between core functionality and optional components. The system is designed around a central PHPMailer class that coordinates between validation, message construction, and transport layers.

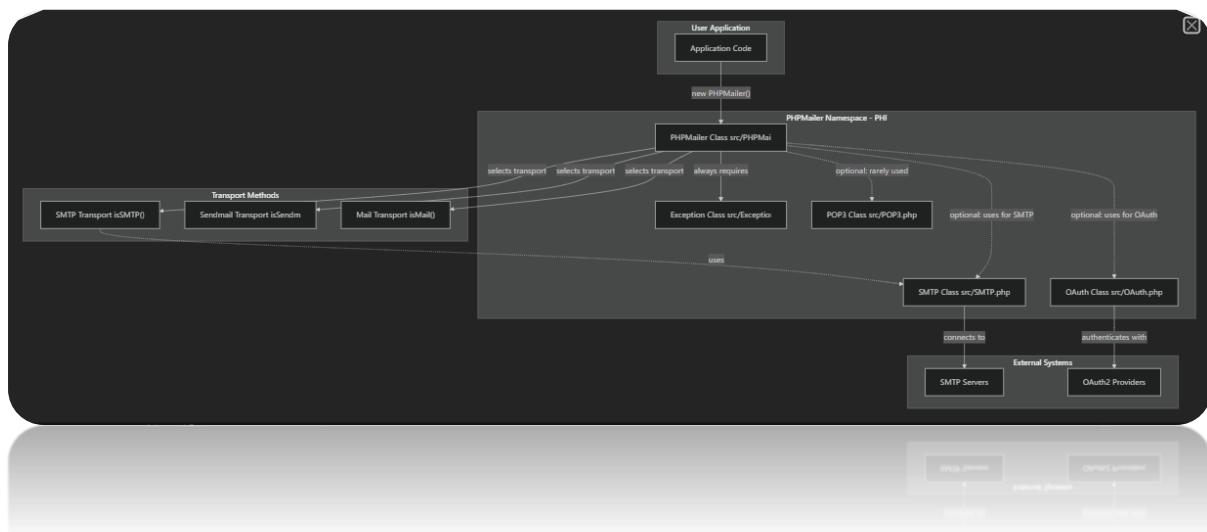


Diagram: PHPMailer Core Component Structure

The architecture emphasizes **optional dependency loading**—components like SMTP, OAuth, and POP3 are only required when their specific functionality is needed. The Exception class is always required as it is used internally for error handling.

13.2. Namespace Organization and PSR-4 Autoloading

PHPMailer uses the PHPMailer\PHPMailer namespace and follows **PSR-4 autoloading standards**. All core classes reside in the src/ directory and map directly to the namespace structure.

13.2.1. Autoloader Configuration

PHPMailer\PHPMailer\ → src/

This mapping is defined in PHPMailer-master/composer.json64-68:

```
"autoload": {  
    "psr-4": {  
        "PHPMailer\\PHPMailer\\": "src/"  
    }  
}
```

13.2.2. Class Loading Pattern

Classes are loaded using standard PHP namespace imports:

```
use PHPMailer\PHPMailer\PHPMailer;  
use PHPMailer\PHPMailer\SMTP;  
use PHPMailer\PHPMailer\Exception;
```

For manual loading without Composer, each class file must be explicitly required in dependency order: Exception.php → PHPMailer.php → SMTP.php.

13.3.Core Class Architecture

PHPMailer's architecture centers on five primary classes, each with distinct responsibilities:

<i>Class</i>	<i>File Path</i>	<i>Responsibility</i>	<i>Dependency Type</i>
<i>PHPMailer</i>	src/PHPMailer.php	Main email configuration, message construction, and sending coordinator	Required
<i>Exception</i>	src/Exception.php	Error handling and exception management	Required (used internally)
<i>SMTP</i>	src/SMTP.php	SMTP protocol implementation and server for SMTP transport communication	Optional (required)
<i>OAuth</i>	src/OAuth.php	XOAUTH2 authentication mechanism	Optional (required for OAuth2)
<i>POP3</i>	src/POP3.php	POP3 protocol for POP-before- SMTP authentication	Optional (rarely used)

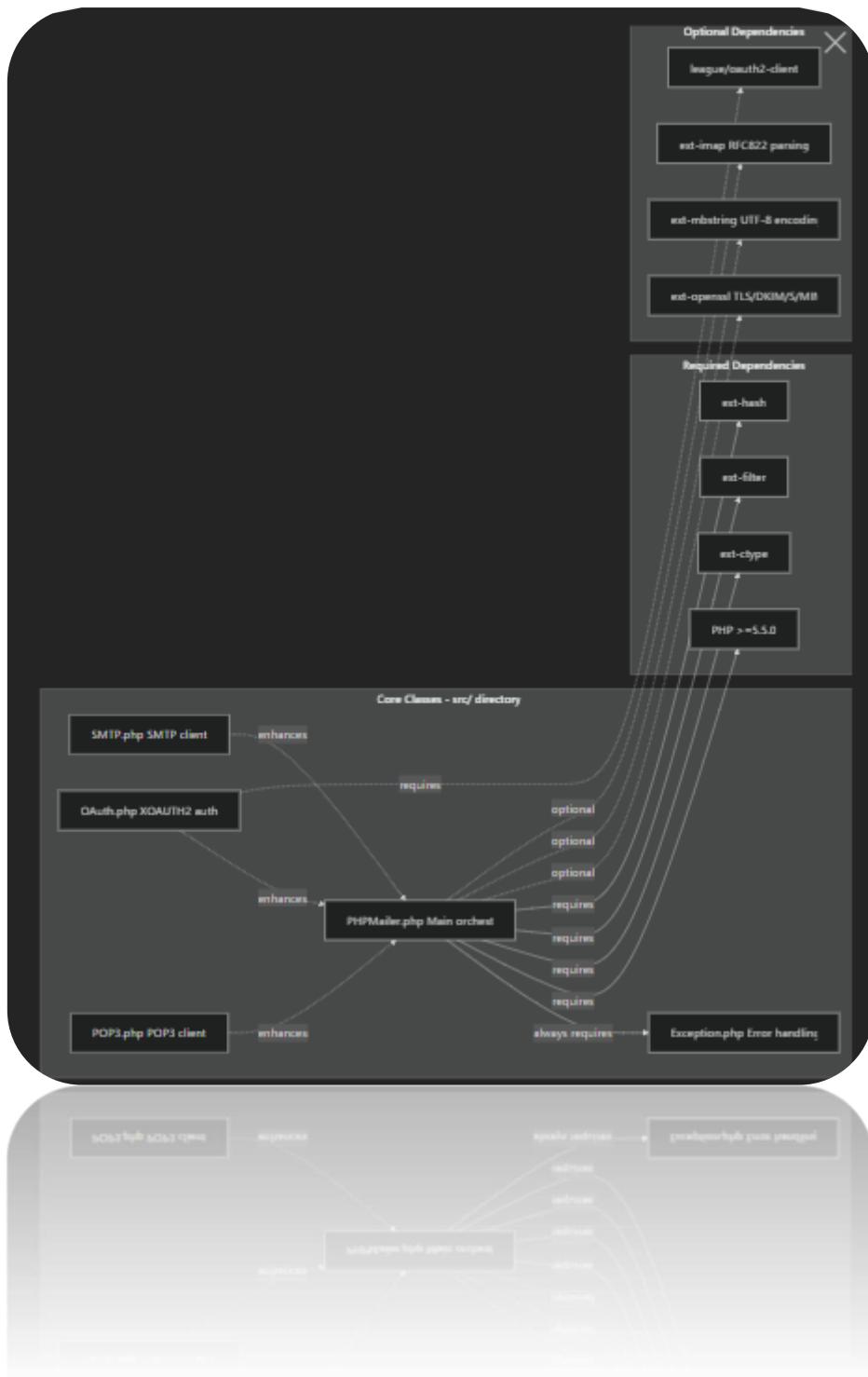


Diagram: Class Dependencies and Extension Requirements

Minimal Installation Requirements

The absolute minimum installation requires:

- `src/PHPMailer.php`
- `src/Exception.php`
- PHP 5.5+ with `ext-ctype`, `ext-filter`, `ext-hash`

For SMTP functionality, add `src/SMTP.php`. For OAuth2, add `src/OAuth.php` plus OAuth2 Composer packages.

13.4.Transport Layer Architecture

PHPMailer implements a **transport abstraction layer** that allows email delivery through three different mechanisms without changing the message construction code.

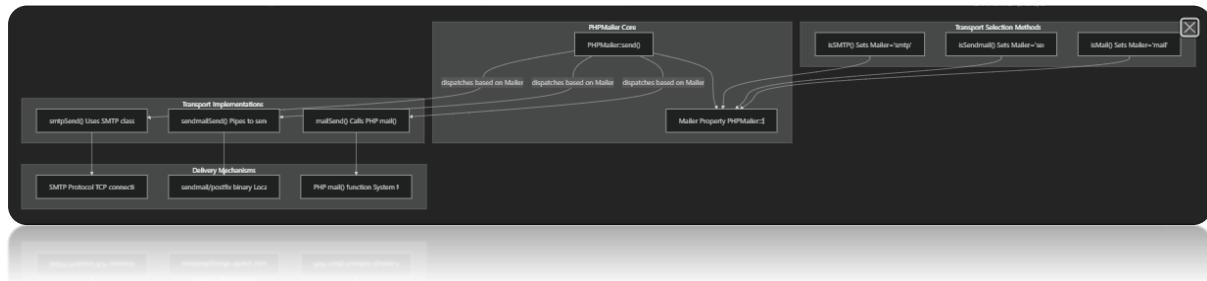


Diagram: Transport Layer Abstraction

Transport Methods

The transport method is configured by calling one of three methods on the PHPMailer instance:

- `isSMTP()`: Routes through the SMTP class for direct SMTP protocol communication
- `isSendmail()`: Executes a local sendmail-compatible binary (Linux/BSD/macOS default)
- `isMail()`: Uses PHP's built-in `mail()` function

13.5. Message Construction Pipeline

Email messages in PHPMailer flow through a multi-stage construction pipeline before being sent:

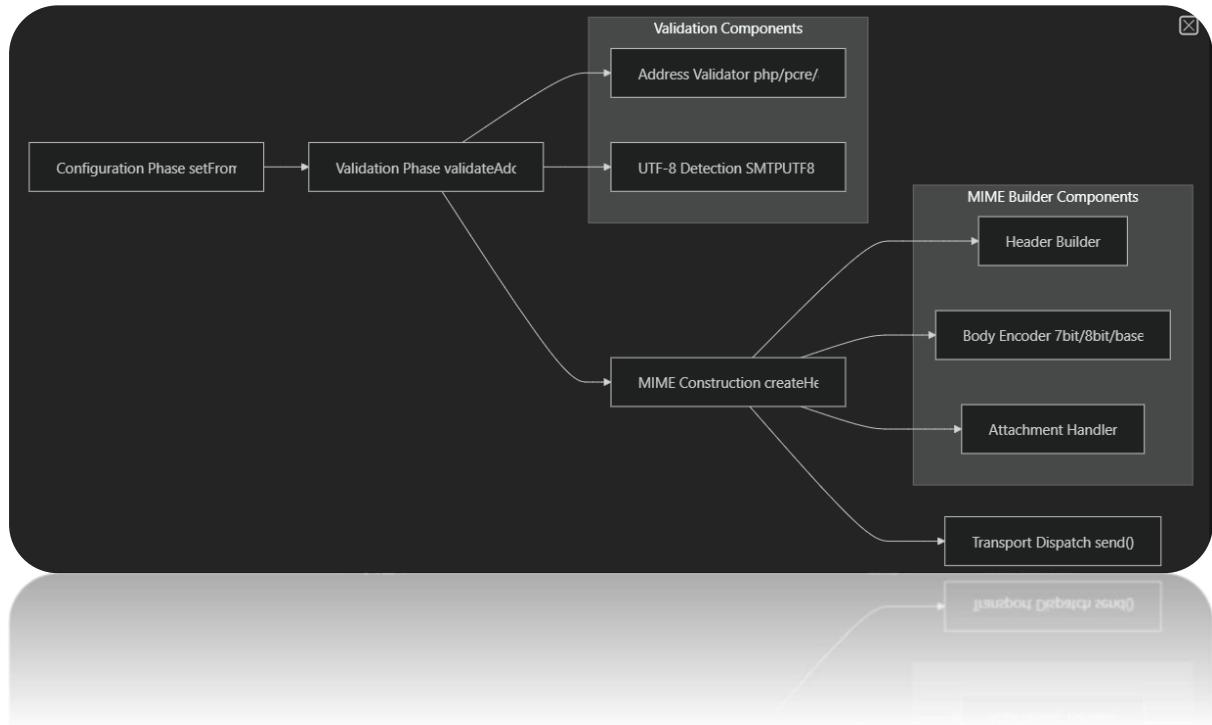


Diagram: Message Construction Pipeline

The pipeline enforces **validation before construction** to prevent malformed messages. UTF-8 addresses automatically trigger SMTPUTF8 mode, switching validators and enabling internationalized email addresses.

13.6.Error Handling Architecture

PHPMailer implements a **dual error handling strategy** supporting both exception-based and boolean return value patterns:

13.6.1.Exception Mode (Recommended)

```
$mail = new PHPMailer(true); // Enable exceptions
```

When exceptions are enabled, all errors throw PHPMailer\PHPMailer\Exception instances with localized error messages.

13.6.2.Boolean Mode (Legacy)

```
$mail = new PHPMailer(false); // Disable exceptions
if (!$mail->send()) {
    echo $mail->ErrorInfo; // Error details available here
}
```

13.6.3.Error Message Localization

Error messages are loaded from the language/ directory based on the configured locale. The setLanguage() method selects the appropriate language file:

```
$mail->setLanguage('fr', '/path/to/language');
```

13.7. Security Architecture

PHPMailer implements **defense-in-depth security** with multiple validation layers protecting against common attack vectors:

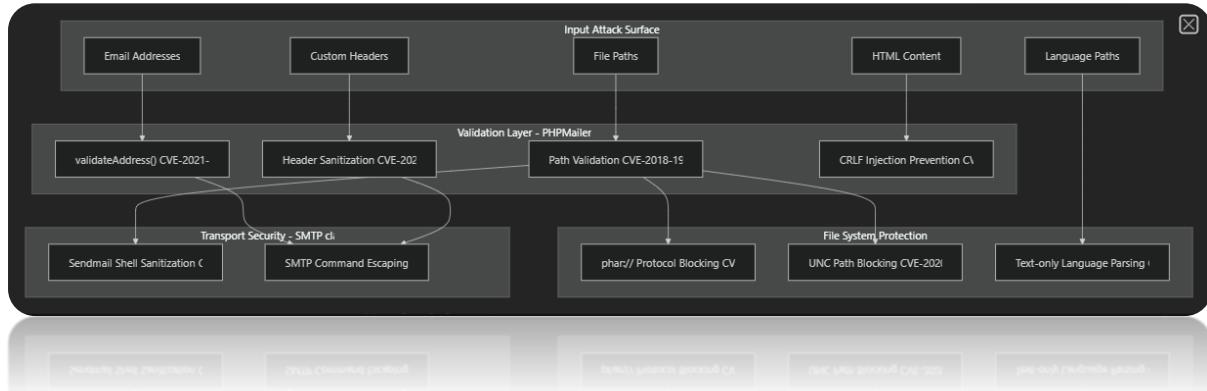


Diagram: Security Validation Layers

Key Security Patterns

1. **Protocol filtering:** Blocks phar://, file://, and UNC paths in file operations
2. **Text-only parsing:** Language files are parsed as text, not executed as PHP code
3. **CRLF sanitization:** Prevents header injection by stripping control characters
4. **Shell escaping:** Sendmail parameters are properly escaped to prevent command injection
5. **Input validation:** All email addresses undergo strict RFC validation

PHPMailer has resolved **19+ documented CVEs** through these defense layers. See Vulnerability History for complete security timeline.

13.8. Configuration Management

PHPMailer uses **public property assignment** for configuration rather than a centralized configuration object. The main configuration categories are:

Category	Example Properties	Configuration Methods
<i>SMTP Settings</i>	Host, Port, SMTPAuth, Username, Password, SMTPSecure	isSMTP(), SMTPOptions array
<i>Recipients</i>	Internal arrays	setFrom(), addAddress(), addCC(), addBCC(), addReplyTo()
<i>Content</i>	Subject, Body, AltBody, CharSet	isHTML(), msgHTML()
<i>Attachments</i>	Internal array	addAttachment(), addStringAttachment(), addEmbeddedImage()
<i>Authentication</i>	AuthType, oauth	OAuth property, getSMTPInstance()
<i>Debugging</i>	SMTPDebug, Debugoutput	Debug constants in SMTP class
<i>Language</i>	language array	setLanguage()

Configuration Example

```
$mail = new PHPMailer(true);
$mail->isSMTP(); // Transport selection
$mail->Host = 'smtp.example.com'; // Direct property assignment
$mail->SMTPAuth = true; // Enable authentication
$mail->Username = 'user@example.com'; // Credentials
$mail->Password = 'secret';
$mail->SMTPSecure = PHPMailer::ENCRYPTION_SMTPS;
$mail->Port = 465;
```

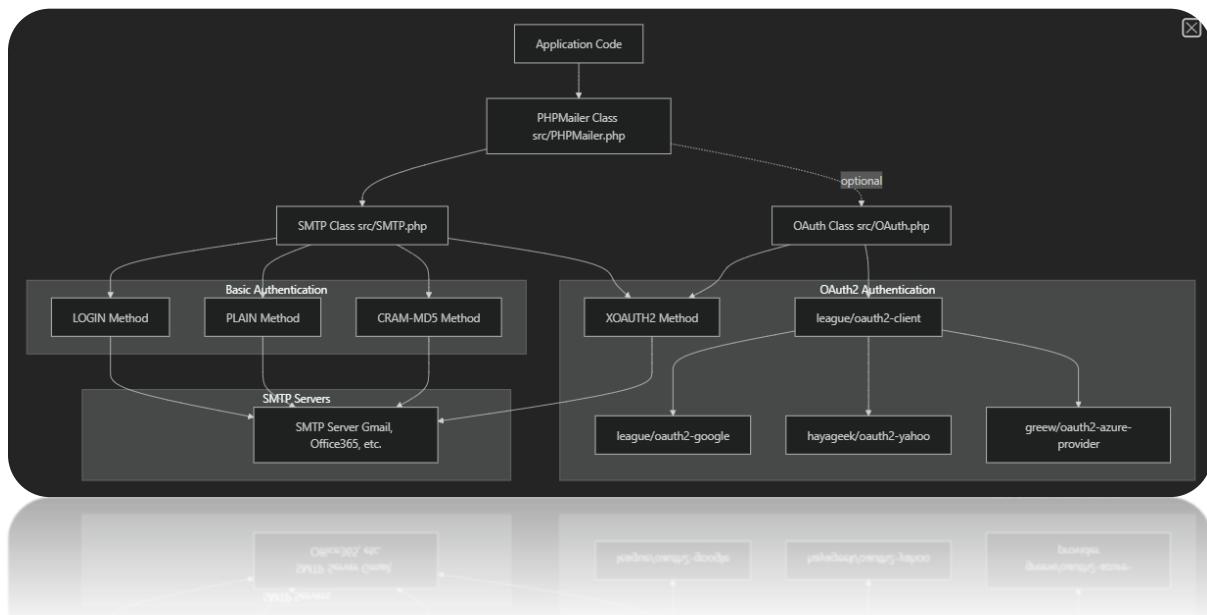
This pattern provides **maximum flexibility** but requires careful validation of user-provided configuration values.

14.Authentication and Security

14.1.Authentication Architecture

PHPMailer supports multiple authentication mechanisms for connecting to SMTP servers, ranging from basic password-based methods to modern OAuth2 token-based authentication. The authentication system is implemented primarily in the SMTP class with OAuth2 support provided by the OAuth class.

Authentication Methods Diagram



14.2. Authentication Methods Summary

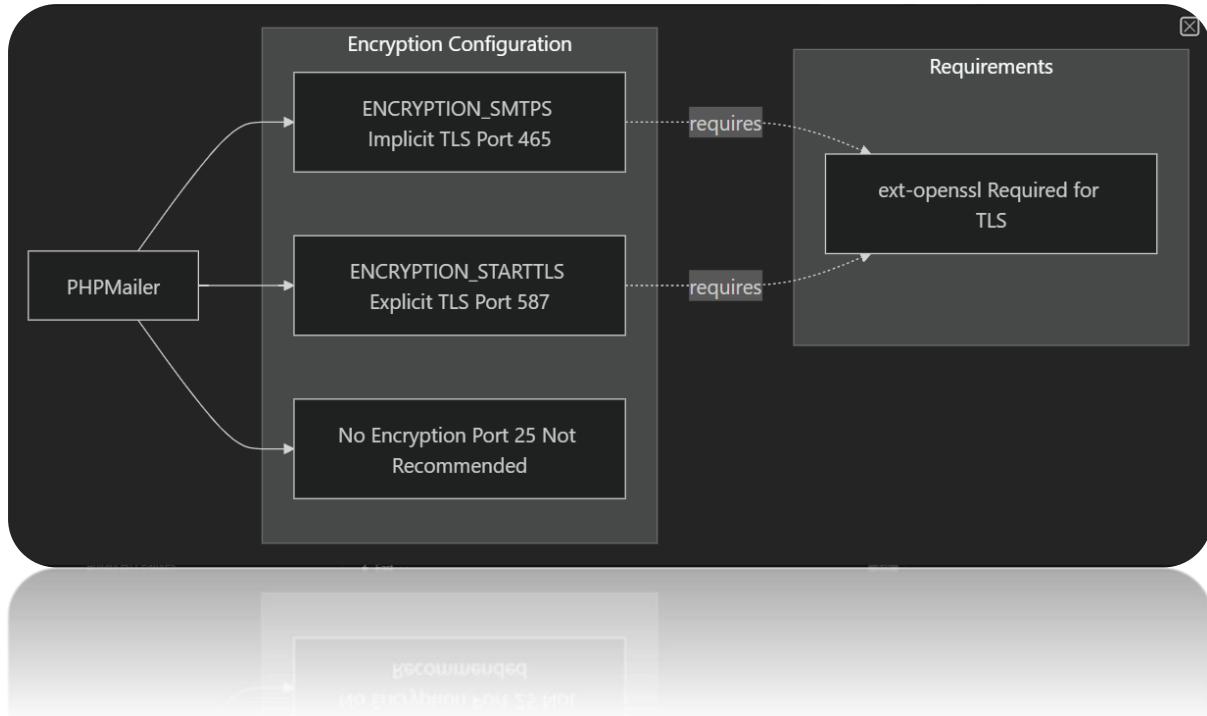
<i>Method</i>	<i>Security Level</i>	<i>Use Case</i>	<i>Dependencies</i>
<i>LOGIN</i>	Basic	Legacy servers, requires TLS	None
<i>PLAIN</i>	Basic	Simple authentication, requires TLS	None
<i>CRAM-MD5</i>	Medium	Challenge-response, no plaintext password	ext-hash
<i>XOAUTH2</i>	High	Modern token-based, no password storage	OAuth class + provider packages

The authentication method is configured through the `SMTPAuth` and `AuthType` properties in the `PHPMailer` class. When `SMTPAuth` is enabled, the `SMTP` class automatically negotiates the most secure available method supported by both the client and server.

14.3.Transport Layer Security

PHPMailer supports encrypted SMTP connections using TLS/SSL protocols to protect credentials and message content during transmission. The library provides two encryption modes configured through the SMTPSecure property:

Encryption Modes

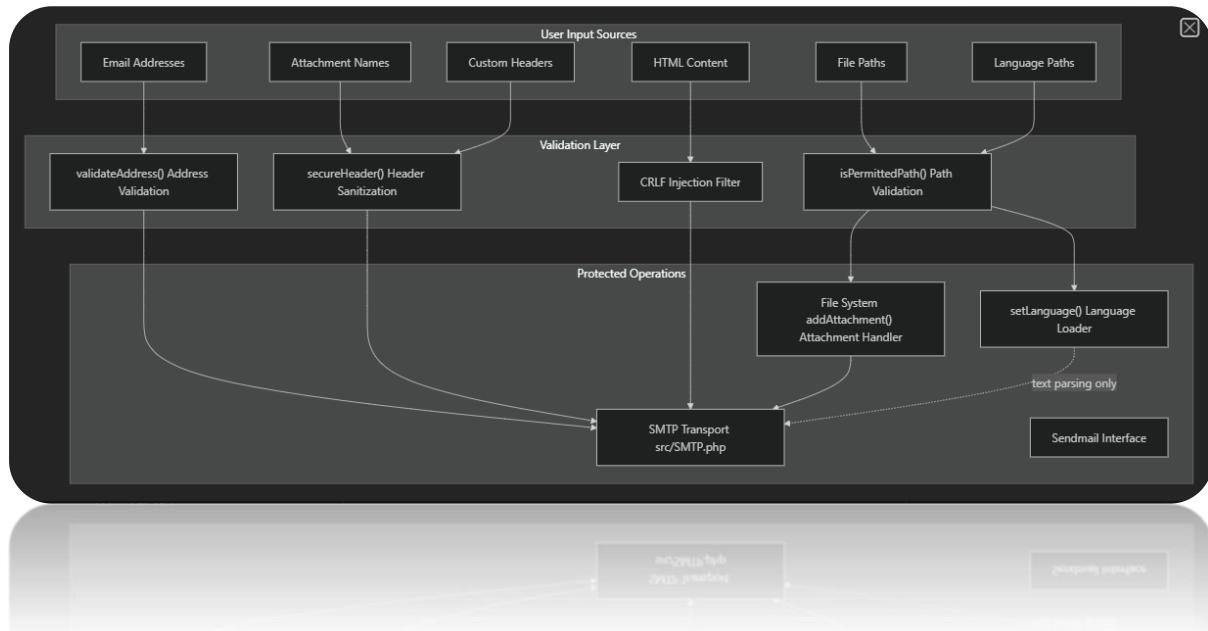


The ENCRYPTION_SMTPS mode (implicit TLS) establishes an encrypted connection immediately on port 465. The ENCRYPTION_STARTTLS mode (explicit TLS) starts an unencrypted connection on port 587 and upgrades it to TLS using the STARTTLS command. Both modes require the ext-openssl PHP extension.

14.4. Security Architecture

PHPMailer implements a defense-in-depth security architecture with multiple validation layers that sanitize user inputs before they reach critical systems like file operations or network transport. This architecture has evolved through 19+ documented CVEs to address various attack vectors.

14.4.1. Security Layers and Attack Surface



14.4.2. Input Validation and Attack Mitigation

PHPMailer validates and sanitizes all user-controllable inputs to prevent various attack vectors. The following table summarizes the primary input validation mechanisms:

<i>Input Type</i>	<i>Validation Method</i>	<i>Mitigated CVEs</i>	<i>Implementation</i>
<i>Email Addresses</i>	<code>validateAddress()</code>	CVE-2021-3603	Prevents untrusted function calls
<i>File Paths</i>	<code>isPermittedPath()</code>	CVE-2018-19296, CVE-2020-36326	Blocks phar://, UNC paths
<i>Attachment Names</i>	<code>secureHeader()</code>	CVE-2020-13625	RFC822 compliant escaping
<i>Message Headers</i>	CRLF sanitization	CVE-2015-8476	Removes \r\n sequences
<i>Language Paths</i>	Path validation + text parsing	CVE-2021-34551, CVE-2010-4914	Blocks code execution
<i>Sendmail Commands</i>	Shell sanitization	CVE-2016-10033, CVE-2016-10045, CVE-2007-3215	Escapes shell metacharacters

14.4.3 Key Security Features

Path Protocol Filtering: The library blocks dangerous path protocols like phar:// that could enable object injection attacks or remote code execution. UNC paths (Windows network paths) are also restricted in file operations.

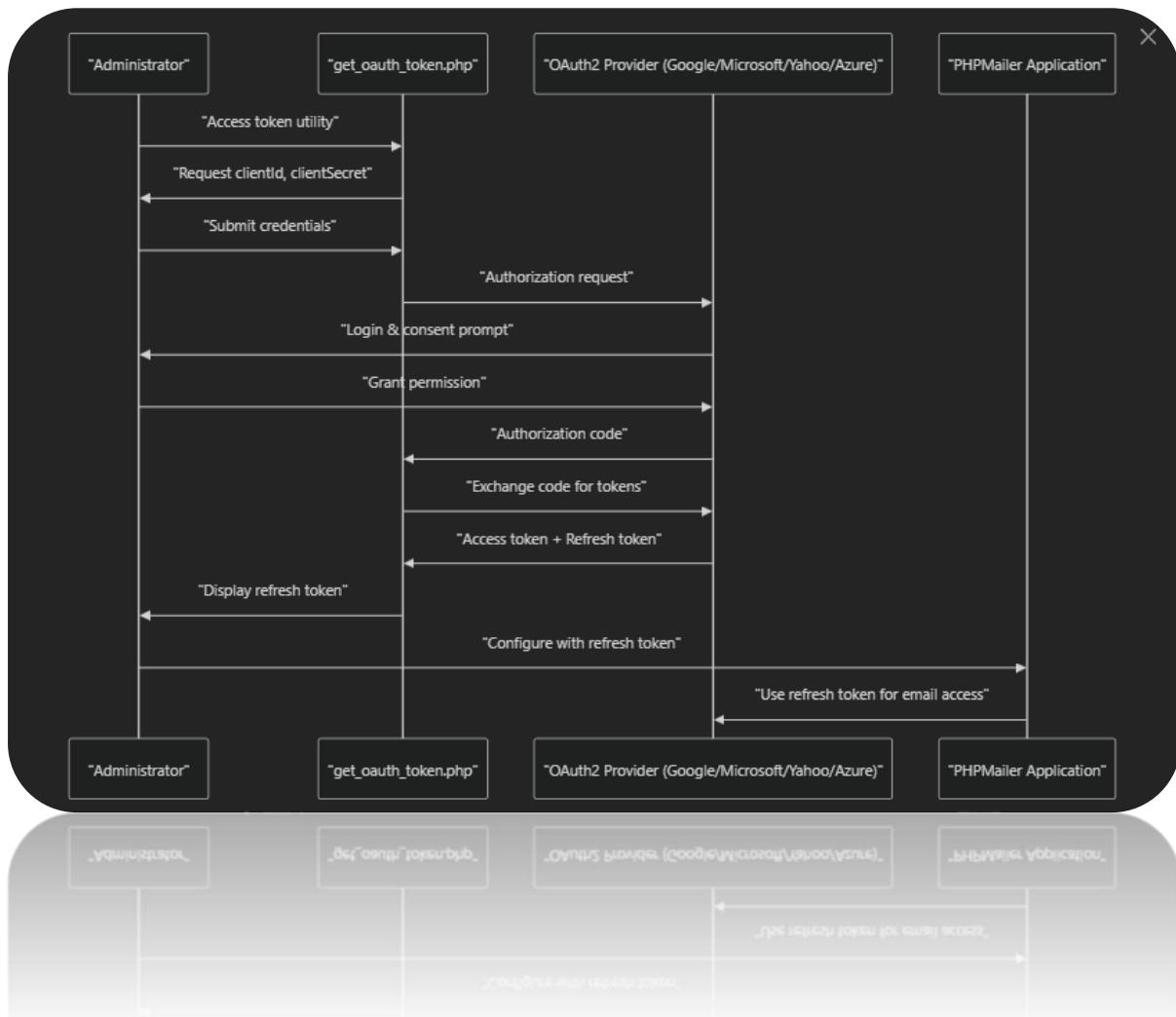
Text-Only Language Parsing: As of version 6.5.0, language files are parsed as text data rather than executed as PHP code, eliminating the possibility of code injection through the language loading mechanism.

Header Injection Prevention: All email headers are sanitized to prevent CRLF injection attacks where an attacker could inject additional headers or message content by including \r\n sequences in input data.

14.5. OAuth2 Token Acquisition

PHPMailer provides OAuth2 authentication support through integration with the League OAuth2 Client library and provider-specific packages. The `get_oauth_token.php` utility assists with the OAuth2 authorization flow and token acquisition.

OAuth2 Token Flow



The token acquisition utility supports four OAuth2 providers:

<i>Provider</i>	<i>Package</i>	<i>Scope Configuration</i>
<i>Google</i>	<code>league/oauth2-google</code>	<code>https://mail.google.com/</code>
<i>Yahoo</i>	<code>hayageek/oauth2-yahoo</code>	Default scope
<i>Microsoft</i>	<code>thenetworg/oauth2-azure</code>	<code>wl.imap, wl.offline_access</code>
<i>Azure</i>	<code>greew/oauth2-azure-provider</code>	<code>https://outlook.office.com/SMTP.Send, offline_access</code>

14.6..DKIM and S/MIME Signing

PHPMailer supports two email signing mechanisms for authentication and encryption:

DKIM Signing: DomainKeys Identified Mail signing adds cryptographic signatures to outgoing messages, allowing receiving servers to verify that the message was sent by an authorized server for the domain. DKIM signing requires the ext-openssl extension.

S/MIME Signing: Secure/Multipurpose Internet Mail Extensions provides end-to-end encryption and digital signatures. S/MIME also requires the ext-openssl extension for cryptographic operations.

Both signing mechanisms enhance email deliverability and protect against spoofing attacks by providing cryptographic proof of message authenticity.

14.7.Required and Optional Security Dependencies

PHPMailer's security features have the following dependency requirements:

14.7.1.Core Security Requirements

<i>Extension</i>	<i>Purpose</i>	<i>Status</i>
<i>ext-hash</i>	CRAM-MD5 authentication	Required
<i>ext-filter</i>	Input filtering	Required
<i>ext-ctype</i>	Character type checking	Required

14.7.2. Optional Security Extensions

<i>Extension</i>	<i>Enables</i>	<i>Impact</i>
<i>ext-openssl</i>	TLS/SSL encryption, DKIM signing, S/MIME	High - required for secure SMTP
<i>ext-mbstring</i>	UTF-8 address handling, multibyte encoding	Medium - required for international support
<i>ext-imap</i>	Advanced RFC822 address parsing	Low - enhances address validation

Without *ext-openssl*, PHPMailer cannot establish encrypted SMTP connections, making credentials vulnerable to interception. Without *ext-mbstring*, the library cannot properly handle international email addresses or content.

15. Advanced Features

15.1. Overview

PHPMailer's advanced features extend the library's capabilities to handle complex email scenarios. These features are modular and typically require optional PHP extensions or Composer packages. The advanced features include:

- **UTF-8 and SMTPUTF8 Support** - Full Unicode support in email addresses, headers, and message bodies
- **Attachments and Inline Images** - File attachments, embedded images, and iCalendar events
- **DKIM and S/MIME Signing** - Email authentication and encryption
- **Custom Headers and Encoding** - Fine-grained control over message structure and character encodings

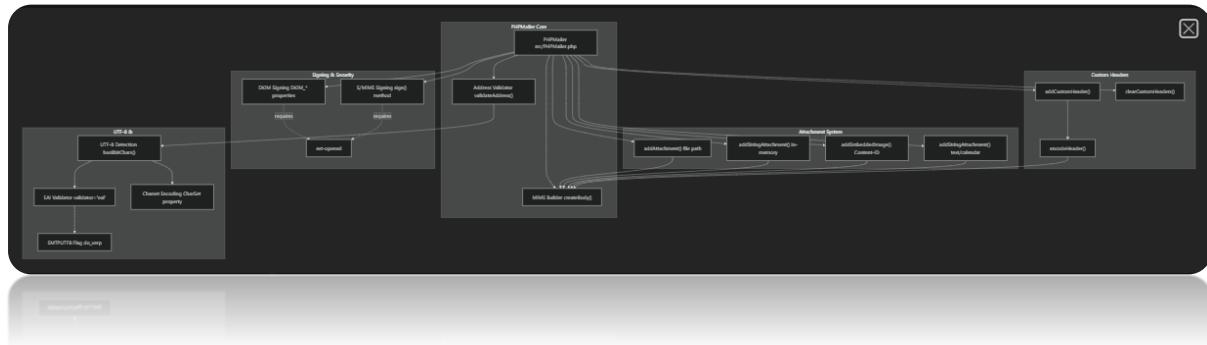
Most advanced features are automatically enabled when the required dependencies are present, with PHPMailer detecting capabilities and adapting its behavior accordingly.

15.2. Feature Dependencies

The following table shows which PHP extensions and optional packages are required for each advanced feature:

<i>Feature</i>	<i>Required Extensions</i>	<i>Optional Packages</i>	<i>Auto-Detected</i>
<i>UTF-8 Email Addresses</i>	<code>ext-mbstring</code> (recommended)	-	Yes
<i>SMTPUTF8 Protocol</i>	<code>ext-mbstring</code>	-	Yes
<i>RFC822 Address Parsing</i>	<code>ext-imap</code>	-	No
<i>File Attachments</i>	<code>None</code>	-	N/A
<i>Inline Images</i>	<code>None</code>	-	N/A
<i>iCalendar Events</i>	<code>None</code>	-	N/A
<i>DKIM Signing</i>	<code>ext-openssl</code>	-	No
<i>S/MIME Signing</i>	<code>ext-openssl</code>	-	No
<i>TLS/SSL Encryption</i>	<code>ext-openssl</code>	-	No
<i>Debug Logging</i>	-	<code>psr/log</code>	No
<i>IMAP Upload</i>	-	<code>directorytree/imapengine</code>	No

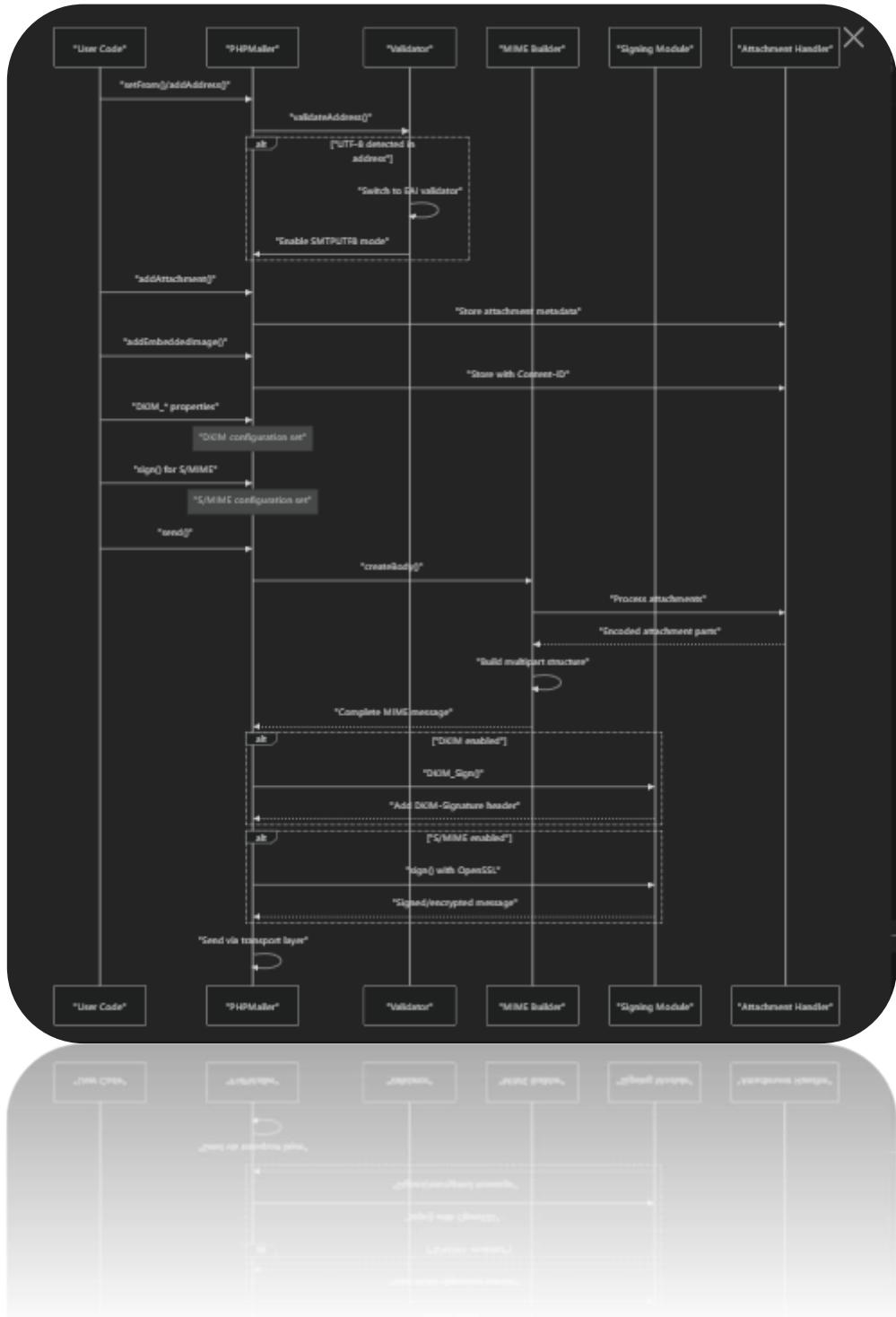
15.3. Feature Architecture



Feature Dependency and Method Architecture

This diagram shows how advanced features integrate with the PHPMailer core through specific methods and properties. UTF-8 detection is automatic, attachment methods feed into the MIME builder, and signing features require OpenSSL.

15.4. Message Composition Flow



Advanced Message Composition Sequence

This diagram illustrates how advanced features are integrated during the message composition process. UTF-8 detection happens during validation, attachments are processed during MIME construction, and signing occurs after the message is fully built.

15.5. Encoding Support

PHPMailer supports multiple content transfer encodings for message bodies and attachments:

<i>Encoding</i>	<i>Property Constant</i>	<i>Use Case</i>
<i>7bit</i>	<code>ENCODING_7BIT</code>	ASCII-only content
<i>8bit</i>	<code>ENCODING_8BIT</code>	UTF-8 content with 8BITMIME servers
<i>base64</i>	<code>ENCODING_BASE64</code>	Binary attachments, non-ASCII text
<i>binary</i>	<code>ENCODING_BINARY</code>	Raw binary data (rare)
<i>quoted-printable</i>	<code>ENCODING_QUOTED_PRINTABLE</code>	Text with occasional non-ASCII

The encoding is selected automatically based on content analysis, but can be overridden using the `Encoding` property. When `SMTPUTF8` mode is active, PHPMailer can send UTF-8 content without encoding.

15.6.Character Set Handling

PHPMailer's CharSet property controls the character encoding for email content. The default is UTF-8, which is recommended for modern applications. When ext-mbstring is available, PHPMailer can handle multibyte character sets properly. Without it, character set conversion and validation may be limited.

The has8bitChars() method detects whether content contains non-ASCII characters, triggering automatic adjustments to encoding and validation strategies.

15.7. Automatic Feature Detection

Several advanced features are enabled automatically when PHPMailer detects the necessary conditions:

1. **SMTPUTF8 Mode** - Automatically enabled when a UTF-8 email address is detected and the validator is compatible SMTPUTF8.md42-44
2. **8BITMIME Support** - Detected from SMTP server capabilities during connection
3. **EAI Validator** - Automatically selected when UTF-8 addresses are used with the default validator SMTPUTF8.md44

This automatic behavior ensures that messages are sent using the most appropriate protocol extensions without requiring explicit configuration.

15.8. Advanced Feature Categories

15.8.1. UTF-8 and SMTPUTF8

Full internationalization support allowing UTF-8 in email addresses, headers, and message bodies. When UTF-8 addresses are detected, PHPMailer automatically enables SMTPUTF8 protocol mode and switches to the EAI (Email Address Internationalization) validator. This feature requires server-side SMTPUTF8 support.

15.8.2. Attachments and Inline Images

PHPMailer supports three types of file attachments:

- **File attachments** via `addAttachment()` - Regular attachments from file paths
- **String attachments** via `addStringAttachment()` - Attachments from in-memory data
- **Embedded images** via `addEmbeddedImage()` - Images referenced in HTML via Content-ID

Additionally, iCalendar events can be attached as text/calendar MIME parts.

15.8.3.DKIM and S/MIME Signing

Email authentication and encryption through two mechanisms:

- **DKIM (DomainKeys Identified Mail)** - Domain-level authentication using public key cryptography
- **S/MIME (Secure/Multipurpose Internet Mail Extensions)** - End-to-end message signing and encryption

15.8.4.Custom Headers and Encoding

Fine-grained control over message structure through:

- `addCustomHeader()` - Add arbitrary RFC-compliant headers
- `clearCustomHeaders()` - Remove previously added custom headers
- `encodeHeader()` - Manually encode header values using RFC 2047

Custom headers allow implementation of specialized email features and protocols.