Net_Growl User Guide

Laurent Laville

Net_Growl User Guide

Laurent Laville

Table of Contents

List of Tables

List of Examples

This guide documents the final stable version 2.7.0

Chapter 1. Introduction

Net_Growl is a PHP Library that makes it possible to easily send a notification from your PHP script to Growl [http://growl.info].

Until now there were many UDP PHP implementations, but none for the new Growl Notification Transport Protocol (GNTP) - v1.0 [http://www.growlforwindows.com/gfw/help/gntp.aspx]

I've decided to enlarge work begun by Bertrand Mansion with the PEAR::Net_Growl [http://pear.php.net/package/Net_Growl] package (v 0.7.0).

While you can find some old compatible PHP4 versions (0.8.0b1, 0.9.0b1, 0.9.0b2), I've decided, with final stable version 2.0.0, to drop support of PHP4.

I'm focus now on a unique PHP5 library that includes all features of GNTP 1.0

Two different protocols may be used: the basic UDP (compatible with all platforms), and the new one GNTP.

The major version 2 is a full rewrites to PHP5, that used exceptions to raise errors.

Its recommanded to migrate to PHP5, because PHP4 version is not maintained anymore.

1.1. Features

Net_Growl provides :

Ability to use both protocol UDP and GNTP. UDP is limited to some features :

- Application and notification icons are not provided by PHP scripts.
- Callbacks are not supported.

GNTP is better than UDP :

- Application and notification icons may be provided by PHP scripts.
- Callbacks may be supported. (not yet with Net_Growl beta1)
- Different Hash and Encryption backend : MD5, SHA1 (for version 0.9.0b1) plus SHA256 and SHA512 (for version 2.0.0b1)

Ability to log messages sent and received to Growl. Log framework used with version 0.9.0b1 is PEAR::Log [http://pear.php.net/package/Log] package. While versions since 2.0.0b1 uses no particular log framework (basic text file I/O).

Ability to auto register before sending notification. Both versions may send notifications without to call the Net_Growl::register() function. Net_Growl will do it for you at first notification.

Ability to display default Growl Logo. If Application and Notification icons are invalid or not reachable, Net_Growl display the default Growl Logo.

since version 2.7.0, you have ability to define the default growl icon to use (see `defaultIcon` option), and its location (see `resourceDir` option).

1.2. Requirements

Mac OSX platform :

- Growl [http://growl.info/] requires Mac OS X 10.5 or higher.

Windows platform :

- Growl for Windows [http://www.growlforwindows.com/gfw/] is a Windows-compatible version of Growl, a notification system for Mac OS X.

Mandatory resources :

- PHP [http://www.php.net] 5.2.0 or newer
- pcre [http://www.php.net/manual/en/book.pcre.php] extension
- SPL [http://www.php.net/manual/en/book.spl.php] extension
- hash [http://www.php.net/manual/en/book.hash.php] extension requires no external libraries and is enabled by default as of PHP 5.1.2

Optional resources :

- mcrypt [http://www.php.net/manual/en/book.mcrypt.php] extension when using GNTP adapter and encryt feature

Chapter 2. Install Guide

Before you begin, ensure that you have at least PHP [http://www.php.net] 5.2.0 installed.

2.1. PEAR

Net_Growl should be installed using the PEAR Installer [http://pear.php.net/]. This installer is the backbone of PEAR, which provides a distribution system for PHP packages, and is shipped with every release of PHP since version 4.3.0.

```
$ pear install Net_Growl
```

2.2. By Hand

Do the following:

1. Download a release archive from http://pear.php.net/package/Net_Growl/download

2. Extract it to a directory that is listed in the `include_path` of your *php.ini* configuration file

Chapter 3. Getting Started

This simple tutorial will show you how to send different messages to Growl.

3.1. Register your application

Before to send any notification, you should register your application to Growl.

Consider an application as a group of elements included :

- a unique name to identify the application (required)
- one icon to represent visually your application (optional - used default growl icon if missing)
- a list of notification types that will receive future messages (required - an empty list does not have sense)

Do not register each time you send a new notification. It’s unnecessary.

Net_Growl will register your application at first notification send, if it was not implicitly called before with Net_Growl::register() .

Example 3.1. Register your first application

```
<?php
require_once 'Net/Growl/Autoload.php';

$notifications = array(
    'GROWL_NOTIFY_STATUS' => array(
        'display' => 'Status',
    ),
    'GROWL_NOTIFY_PHPERROR' => array(
        'display' => 'Error-Log'
    )
);
$appName  = 'PHP App Example using GNTP';
$password = '';
$options  = array(
    'protocol' => 'gntp',
    'timeout'  => 15,
);

try {
    $growl = Net_Growl::singleton($appName, $notifications, $password, $options);
    $growl->register();

} catch (Net_Growl_Exception $e) {
```

```
    echo 'Caught Growl exception: ' . $e->getMessage() . PHP_EOL;
?>
```

Previously, we have seen how to register an application with all definitions given by the Net_Growl class constructor. Now we will see an alternative solution using a Net_Growl_Application object.

Example 3.2. Register with a Net_Growl_Application object

```
<?php
require_once 'Net/Growl/Autoload.php';

$notifications = array(
    'GROWL_NOTIFY_PHPERROR'
);
$appName  = 'PHP App Example using GNTP';
$password = '';

$app = new Net_Growl_Application(
    $appName,
    $notifications,
    $password
);
$options = array(
    'protocol' => 'gntp',
);

$growl = Net_Growl::singleton($app, null, null, $options);
$growl->register();
?>
```

3.2. Notify a simple basic message

Distinct UDP and GNTP communication

Default options will used the basic UDP protocol on port 9887.

If you want to use the new GNTP, you should specify **options** *protocol* (= gntp)

See singleton method, and options parameter (#4).

We will reused the source code presented in register application feature, and used tip to auto-register application at first notification call.

```
<?php
require_once 'Net/Growl/Autoload.php';

$notifications = array(
    'GROWL_NOTIFY_STATUS' => array(
        'display' => 'Status',
    ),
    'GROWL_NOTIFY_PHPERROR' => array(
        'display' => 'Error-Log'
```

```
    )
);
$appName  = 'PHP App Example using GNTP';
$password = '';
$options  = array(
    'protocol' => 'gntp',
    'timeout'  => 15,
);

try {
    $growl = Net_Growl::singleton($appName, $notifications, $password, $options);

    $name        = 'GROWL_NOTIFY_STATUS';
    $title       = 'Congratulation';
    $description = 'You have successfully installed PEAR/Net_Growl.';
    $growl->publish($name, $title, $description);

} catch (Net_Growl_Exception $e) {
    echo 'Caught Growl exception: ' . $e->getMessage() . PHP_EOL;
?>
```

We have defined two notifications type [1] when register application on getting a Growl instance. But we use only one of them to send our basic *Congratulation* message.

3.3. Notify different message types

You can define as much notification types as you want, depending of your need. For example, Gmail Growl [http://gmailgrowl.blogspot.com/] sets 3 types

- New Mail
- state
- New Version

Here, in our example, we will set 2 notification types

- Status (GROWL_NOTIFY_STATUS)
- Error-Log (GROWL_NOTIFY_PHPERROR)

and send messages on both channels.

Here are the full script, we will explain just after :

```
<?php
require_once 'Net/Growl/Autoload.php';

$notifications = array(
```

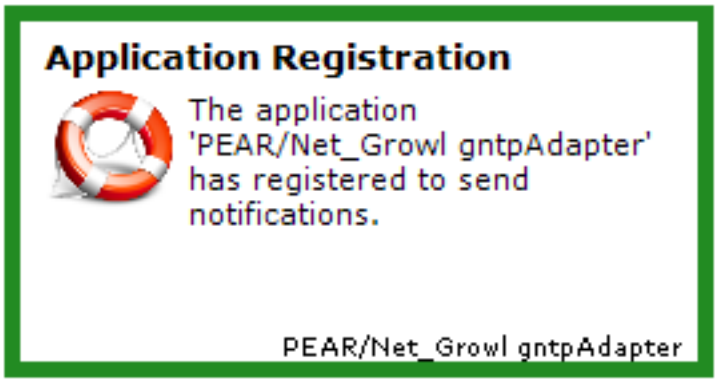
[1] GROWL_NOTIFY_STATUS and GROWL_NOTIFY_PHPERROR

```
    'GROWL_NOTIFY_STATUS' => array(
        'display' => 'Status',
    ),
    'GROWL_NOTIFY_PHPERROR' => array(
        'icon'    => 'http://www.laurent-laville.org/growl/images/firephp.png',
        'display' => 'Error-Log'
    )
);
$appName  = 'PHP App Example using GNTP';
$password = '';
$options  = array(
    'protocol' => 'gntp',
    'timeout'  => 15,
    'AppIcon'  => 'http://www.laurent-laville.org/growl/images/Help.png',
);

try {
    $growl = Net_Growl::singleton($appName, $notifications, $password, $options);

    $name        = 'GROWL_NOTIFY_STATUS';
    $title       = 'Congratulation';
    $description = 'You have successfully installed PEAR/Net_Growl.';
    $growl->publish($name, $title, $description);

    $name        = GROWL_NOTIFY_PHPERROR;
    $title       = 'New Error';
    $description = 'You have a new PHP error in your script.';
    $options     = array(
        'priority' => Net_Growl::PRIORITY_HIGH,
    );
    $growl->publish($name, $title, $description, $options);

    $name        = GROWL_NOTIFY_STATUS;
    $title       = 'Welcome';
    $description = "Welcome in PHP/GNTP world ! \n"
                 . "New GNTP protocol add icon support.";
    $options     = array(
        'icon'   => 'http://www.laurent-laville.org/growl/images/unknown.png',
        'sticky' => false,
    );
    $growl->publish($name, $title, $description, $options);

} catch (Net_Growl_Exception $e) {
    echo 'Caught Growl exception: ' . $e->getMessage() . PHP_EOL;
?>
```

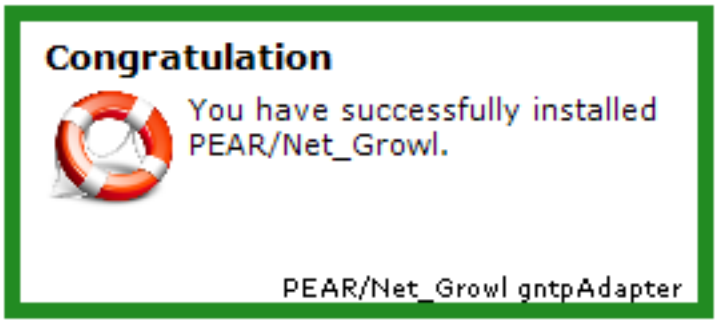
First message notified by code below

```
<?php
    $name        = 'GROWL_NOTIFY_STATUS';
    $title       = 'Congratulation';
    $description = 'You have successfully installed PEAR/Net_Growl.';
    $growl->publish($name, $title, $description);
```

will show the Toast notification (only on first script run)



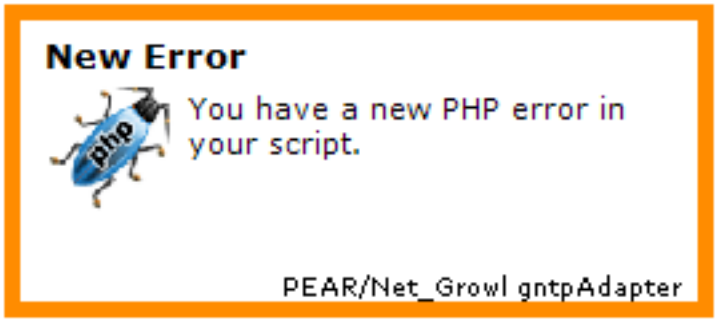
Followed by



Second message sent over the other channel (notification type) on a highest priority

```
<?php
    $name        = GROWL_NOTIFY_PHPERROR;
    $title       = 'New Error';
    $description = 'You have a new PHP error in your script.';
    $options     = array(
        'priority' => Net_Growl::PRIORITY_HIGH,
    );
    $growl->publish($name, $title, $description, $options);
```

will show this Toast notification

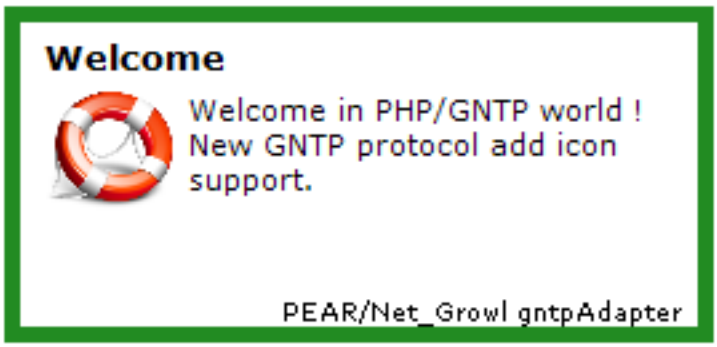


And finally the third and last message with default application icon [2] because resource **http://www.laurent-laville.org/growl/images/unknown.png** does not exist.

[2] http://www.laurent-laville.org/growl/images/Help.png (AppIcon option of singleton method)

```
<?php
    $name        = GROWL_NOTIFY_STATUS;
    $title       = 'Welcome';
    $description = "Welcome in PHP/GNTP world ! \n"
                 . "New GNTP protocol add icon support.";
    $options     = array(
        'icon'   => 'http://www.laurent-laville.org/growl/images/unknown.png',
        'sticky' => true,
    );
    $growl->publish($name, $title, $description, $options);
```

will show this other one Toast notification



Chapter 4. Secure your communication

This chapter is only for reader that use the GNTP adapter. Even if UDP is binary format protocol [http://growl.info/documentation/developer/protocol.php], it shouldn't be considered as secure.

GNTP is a MIME like format, contents are sent as readable plain text. You should think to use either :

- a HTTPS secure channel
- encrypt your data on a basic HTTP channel

We will see now, how to encrypt your data and what formats are supported.

The authorization of messages is accomplished by passing key information that proves that the sending application knows a shared secret with the notification system, namely a password. Users that want to authorize applications must share with them a password that will be used for both authorization and encryption.

By default, authorization is not required for requests orginating on the local machine.

Table 4.1. The supported hashing algorithms

Hash	Features
MD5	128-bit, 16 byte, 32 character length when hex encoded
SHA1	160-bit, 20 byte, 40 character length when hex encoded
SHA256	256-bit, 32 byte, 64 character length when hex encoded
SHA512	512-bit, 64 byte, 128 character length when hex encoded

Table 4.2. The supported encryption algorithms

Hash	Features
NONE	No encryption; messages are sent in plain text
AES	key length: 24 bytes (192 bit), block size: 16 byte (128 bit), iv size: 16 byte (128 bit)
DES	key length: 8 bytes (64 bit), block size: 8 byte (64 bit), iv size: 8 byte (64 bit)
3DES	key length: 24 bytes (192 bit), block size: 8 byte (64 bit), iv size: 8 byte (64 bit)

All encryption algorithms should use a block mode of CBC (Cipher Block Chaining) and PKCS5 (PKCS7) padding.

It is important to keep in mind that some encryption algorithms require keys that are longer than can be generated by some hashing algorithms. As such, not all hash/encryption combinations are valid (ex: MD5 hash produces a 16 byte result, but AES encryption requires a 24-byte key).

Table 4.3. Hash and Encryption algorithms compatibilities

Encryption	Hash
AES	SHA256, SHA512
DES	MD5, SHA1, SHA256, SHA512
3DES	SHA256, SHA512

To encrypt your data, its very easy. You have just to specify in **options** of Net_Growl::singleton method :

- password hash algorithm, see the supported hashing algorithms Table 4.1, “The supported hashing algorithms”.
- encryption algorithm, see the supported encryption algorithms .

Example 4.1. encrypted messages with AES/SHA256

```
<?php
require_once 'Net/Growl/Autoload.php';

$notifications = array(
    'GROWL_NOTIFY_PHPERROR'
);
$appName  = 'PHP App Example using GNTP/encrypt AES';
$password = 'mamasam';

$app = new Net_Growl_Application(
    $appName,
    $notifications,
    $password
);
$options = array(
    'protocol' => 'gntp',
    'timeout'  => 10,
    'encryptionAlgorithm'   => 'AES',
    'passwordHashAlgorithm' => 'SHA256',
);

try {
    $growl = Net_Growl::singleton($app, null, null, $options);
    $growl->register();

    $title       = 'Welcome in PHP/GNTP world';
    $description = "New GNTP protocol support 3 encryption algorithms ! \n"
                 . "DES, 3DES, AES with 4 hash algorithm \n"
```

```
                  . "MD5, SHA1, SHA256, SHA512.";
    $options     = array(
        'sticky' => true,
    );
    $growl->publish($name, $title, $description, $options);

} catch (Net_Growl_Exception $e) {
    echo 'Growl exception: ' . $e->getMessage() . PHP_EOL;
}
?>
```

Chapter 5. FAQ

If you are in trouble, perhaps this page will give you a solution.

5.1. Troubleshooting

5.1.1. The response times are slow

You can reduced timeout period on a stream (socket) to connect/read. Default is 30 secondes, like *php.ini* `default_socket_timeout` directive.

Example 5.1. Set timetout to 15 secondes

```
<?php
require_once 'Net/Growl/Autoload.php';

$notifications = array(
    'GROWL_NOTIFY_PHPERROR'
);
$appName  = 'PHP App Example using GNTP';
$password = '';

$app = new Net_Growl_Application(
    $appName,
    $notifications,
    $password
);
$options = array(
    'protocol' => 'gntp',
    'timeout'  => 15
);

$growl = Net_Growl::singleton($app, null, null, $options);
$growl->register();
?>
```

You can also suppress timeout by setting value to zero.

5.1.2. Notifications are not displayed

- Are you sure Growl is running (not stopped or paused) ?
- Check if your application and the notification type used is well registered
- If your application is well registered, check if notifications display are enabled
- Password provided by your application is probably unknown of Growl client (see Password Manager on Security Tab)

- Check if your code produces error/exceptions.

Example 5.2. Catch errors with a try catch

```
<?php
require_once 'Net/Growl/Autoload.php';

$notifications = array(
    'GROWL_NOTIFY_PHPERROR'
);
$appName  = 'PHP App Example using GNTP';
$password = '';
$options  = array(
    'protocol' => 'gntp',
);

try {
    $growl = Net_Growl::singleton($appName, $notifications, $password, $options);
    $growl->register();

} catch (Net_Growl_Exception $e) {
    echo 'Caught Growl exception: ' . $e->getMessage() . PHP_EOL;
}
?>
```

All errors produced by Net_Growl raise a Net_Growl_Exception

5.1.3. Net_Growl is not really verbose

To known what MIME messages are sent and received from Growl, activate the `verbose` mode. Give a valid path to a filename on `debug` option.

Example 5.3. Activate the debug mode

```
<?php
require_once 'Net/Growl/Autoload.php';

$notifications = array(
    'GROWL_NOTIFY_PHPERROR'
);
$appName  = 'PHP App Example using GNTP';
$password = '';
$options  = array(
    'protocol' => 'gntp',
    'debug'    => dirname(__FILE__) . DIRECTORY_SEPARATOR . 'netgrowl.log'
);

try {
    $growl = Net_Growl::singleton($appName, $notifications, $password, $options);
    $growl->register();
```

```
} catch (Net_Growl_Exception $e) {
    echo 'Caught Growl exception: ' . $e->getMessage() . PHP_EOL;
}
?>
```

5.1.4. My favorite application icons are not shown

- URL given are not valid or not reachable
- URL are good but resources are invalid images

If you give URL/resource that are not valid, Net_Growl will use default icon returns by Net_Growl::getDefaultGrowlIcon method.

5.1.5. How to detect error with new version 2.1

GNTP specialized response are now returned with version 2.1.0 (or greater). If you want to catch an error, test status code after each register() or notify() method. See Net_Growl_Response::getStatus

Example 5.4. Use the Net_Growl_Response object

```
<?php
require_once 'Net/Growl/Autoload.php';

$notifications = array(
    'GROWL_NOTIFY_PHPERROR'
);
$appName  = 'PHP App Example using GNTP';
$password = '';
$options  = array(
    'protocol' => 'gntp',
    'debug'    => dirname(__FILE__) . DIRECTORY_SEPARATOR . 'netgrowl.log'
);

try {
    $growl = Net_Growl::singleton($appName, $notifications, $password, $options);
    $resp  = $growl->register();

    if ($resp->getStatus() != 'OK') {
        die($resp);
    }

} catch (Net_Growl_Exception $e) {
    echo 'Caught Growl exception: ' . $e->getMessage() . PHP_EOL;
}
?>
```

Chapter 6. API

6.1. Overview

Table 6.1. All classes

Name	Description
Net_Growl	A PHP library that talk to Growl
Net_Growl_Application	Application object for Net_Growl
Net_Growl_Exception	Dedicated Exception for Net_Growl
Net_Growl_Udp	UDP adapter
Net_Growl_Gntp	GNTP adapter
Net_Growl_GntpMock	GNTP Mock adapter intended for test only
Net_Growl_Response	GNTP specialized response

6.2. Net_Growl Class

6.2.1. Synopsis

```
<?php
class Net_Growl
{
    /* constants */
    const string VERSION;
    const int UDP_PORT;
    const int GNTP_PORT;
    const int PRIORITY_LOW;
    const int PRIORITY_MODERATE;
    const int PRIORITY_NORMAL;
    const int PRIORITY_HIGH;
    const int PRIORITY_EMERGENCY;

    /* properties */
    protected array $options;
    protected array $growlNotificationCallback;
    protected int $growlNotificationCount;
    protected bool $isRegistered;
    protected static object $instance;

    private object $_application;
    private int $_growlNotificationLimit;
    private resource $_fp;

    /* methods */
```

```
    public static final object singleton(mixed &$application, array $notifications [, strin
    public static final void reset();
    public void __destruct();
    public array getOptions();
    public void setNotificationLimit($max);
    public object getApplication();
    public bool | Net_Growl_Response register();
    public bool | Net_Growl_Response notify(string $name, string $title [, string $descript
    public bool | Net_Growl_Response publish(string $name, string $title [, string $descrip
    public string getDefaultGrowlIcon();
    public static void autoload($class);
    public void errorHandler(int $errno, string $errstr, string $errfile, int $errline);

    protected object __construct(mixed &$application, array $notifications [, string $passw
    protected bool | Net_Growl_Response sendRequest(string $method, mixed $data [, bool $ca
    protected void debug(string $message [, string $priority = 'debug']);
    protected string utf8Encode($data);
    protected int strByteLen($string);

    private string _readLine(resource $fp);
}
```

6.2.2. Constants

Table 6.2. Net_Growl Constants

Name	Value	Description
VERSION	n/a	PHP/Net_Growl version
UDP_PORT	9887	Growl default UDP port
GNTP_PORT	23053	Growl default GNTP port
PRIORITY_LOW	-2	Growl low priority
PRIORITY_MODERATE	-1	Growl moderate priority
PRIORITY_NORMAL	0	Growl normal priority
PRIORITY_HIGH	1	Growl high priority
PRIORITY_EMERGENCY	2	Growl emergency priority

6.2.3. Methods

Table 6.3. Net_Growl Methods

Name	Description
singleton	Makes sure there is only one Growl connection open
setNotificationLimit	Limit the number of notifications
getApplication	Returns the registered application object

Name	Description
register	Sends a application register to Growl
notify	Sends a notification to Growl
getDefaultGrowlIcon	Returns Growl default icon logo binary data
autoload	Autoloader for PEAR compatible classes
errorHandler	Converts standard error into exception
getOptions	Gets options used with current Growl object
publish	Sends a notification to Growl (alias of notify method)

Net_Growl::singleton

Synopsis

require_once 'Net/Growl/Autoload.php';

object **Net_Growl::singleton**(&$application, $notifications, $password = '', $options = array())

Description. Makes sure there is only one Growl connection open.

Parameter

mixed $application
Can be either a Net_Growl_Application object or the application name string

array $notifications
List of notification types

string $password
(optional) Password for Growl

array $options
(optional) List of options :

- *host*, *port*, *protocol*, *timeout*
  - for Growl socket server
- *passwordHashAlgorithm*, *encryptionAlgorithm*
  - to secure communications
- *debug*
  - to know what data are sent and received.

Throws

Net_Growl_Exception
if class handler does not exists

Return value. object - Net_Growl

Net_Growl::setNotificationLimit

Synopsis

require_once 'Net/Growl/Autoload.php';

void **Net_Growl::setNotificationLimit**($max)

Description. This method limits the number of notifications to be displayed on the Growl user desktop. By default, there is no limit. It is used mostly to prevent problem with notifications within loops.

Parameter

int $max
Maximum number of notifications

Throws. no exceptions thrown

Return value. void

Net_Growl::getApplication

Synopsis

require_once 'Net/Growl/Autoload.php';

object **Net_Growl::getApplication**( )

Description. Returns the registered application object

Throws. no exceptions thrown

Return value. object - Net_Growl_Application

Net_Growl::register

Synopsis

require_once 'Net/Growl/Autoload.php';

bool | Net_Growl_Response **Net_Growl::register**( )

Description. Sends a application register to Growl

Throws

Net_Growl_Exception
if REGISTER failed

Return value. void

Net_Growl::notify

Synopsis

require_once 'Net/Growl/Autoload.php';

bool | Net_Growl_Response **Net_Growl::notify**($name, $title, $description = '', $options = array())

Description. Sends a notification to Growl

Growl notifications have a name, a title, a description and a few options, depending on the kind of display plugin you use. The bubble plugin is recommended, until there is a plugin more appropriate for these kind of notifications.

The current options supported by most Growl plugins are:

```
<?php
 array('priority' => 0, 'sticky' => false);
```

- sticky: whether the bubble stays on screen until the user clicks on it.
- priority: a number from -2 (low) to 2 (high), default is 0 (normal).

Parameter

string $name
Notification name

string $title
Notification title

string $description
(optional) Notification description

string $options
(optional) few Notification options

Throws

Net_Growl_Exception
if NOTIFY failed

Return value. bool - true

Net_Growl::getDefaultGrowlIcon

Synopsis

require_once 'Net/Growl/Autoload.php';

string **Net_Growl::getDefaultGrowlIcon**($return = true, $ver = *2*)

Description. Returns Growl default icon logo binary data. Decodes data encoded with MIME base64

Parameter

bool $return
(optional) If used and set to FALSE, getDefaultGrowlIcon() will output the binary representation instead of return it

string $ver
(optional) Icon version

Throws. no exceptions thrown

Return value. string - icon logo binary data

Example 6.1. Display Growl Icon

```
<?php
require_once 'Net/Growl/Autoload.php';

Net_Growl::getDefaultGrowlIcon(false);
?>
```

Net_Growl::autoload

Synopsis

require_once 'Net/Growl/Autoload.php';

void **Net_Growl::autoload**($class)

Description. Autoloader for PEAR compatible classes

Parameter

string $class
Class name

Throws

Net_Growl_Exception
if class handler cannot be loaded

Return value. void

Net_Growl::errorhandler

Synopsis

require_once 'Net/Growl/Autoload.php';

void **Net_Growl::errorhandler**($errno, $errstr, $errfile, $errline)

Description. Throws ErrorException when a standard error occured with severity level we are asking for (uses error_reporting)

Parameter

int $errno
contains the level of the error raised

string $errstr
contains the error message

string $errfile
contains the filename that the error was raised in

string $errline
contains the line number the error was raised at

Throws

ErrorException
corresponding to standard error/warning/notice raised

Return value. void

Net_Growl::getOptions

Synopsis

require_once ‘Net/Growl/Autoload.php’;

array **Net_Growl::getOptions**()

Description. Gets options used with current Growl object

Return value. array

Net_Growl::publish

Synopsis

require_once ‘Net/Growl/Autoload.php’;

bool | Net_Growl_Response **Net_Growl::publish**($name, $title, $description = '', $options = array())

Description. Sends a notification to Growl. Alias of notify() method.

Growl notifications have a name, a title, a description and a few options, depending on the kind of display plugin you use. The bubble plugin is recommended, until there is a plugin more appropriate for these kind of notifications.

The current options supported by most Growl plugins are:

```
<?php
 array('priority' => 0, 'sticky' => false);
```

- sticky: whether the bubble stays on screen until the user clicks on it.
- priority: a number from -2 (low) to 2 (high), default is 0 (normal).

Parameter

string $name
: Notification name

string $title
: Notification title

string $description
: (optional) Notification description

string $options
(optional) few Notification options

Throws

Net_Growl_Exception
if NOTIFY failed

Return value. bool - true

6.3. Net_Growl_Application Class

6.3.1. Synopsis

```
<?php
class Net_Growl_Application
{
    /* properties */
    private string $_growlAppName;
    private string $_growlAppPassword;
    private Net_Growl_Icon $_growlAppIcon;
    private array $_growlNotifications;

    /* methods */
    public object __construct([mixed $appName = null [, array $notifications = null [, stri
    public void addGrowlNotifications(array $notifications);
    public array getGrowlNotifications();
    public string setGrowlName(string $appName);
    public string getGrowlName();
    public string setGrowlPassword(string $password);
    public string getGrowlPassword();
    public string setGrowlIcon(mixed $appIcon);
    public string getGrowlIcon();

}
```

6.3.2. Methods

Table 6.4. Net_Growl_Application Methods

Name	Description
__construct	Constructs a new application to be registered by Growl
addGrowlNotifications	Adds notifications supported by this application
getGrowlNotifications	Returns the notifications accepted by Growl for this application
setGrowlName	Sets the application name for registration in Growl

Name	Description
getGrowlName	Returns the application name for registration in Growl
setGrowlPassword	Sets the password to be used by Growl to accept notification packets
getGrowlPassword	Returns the password to be used by Growl to accept notification packets
setGrowlIcon	Sets the application icon for registration in Growl
getGrowlIcon	Returns the application icon for registration in Growl

Net_Growl_Application::__construct

Synopsis

require_once 'Net/Growl/Autoload.php';

object **new Net_Growl_Application**($appName, $notifications, $password = '', $appIcon = '')

Description. Constructs a new application to be registered by Growl

Parameter

string $appName
Application name

array $notifications
Array of notifications

string $password
(optional) Password to be used to notify Growl

string $appIcon
(optional) Application icon

Throws. no exceptions thrown

Return value. object - Net_Growl_Application

Example 6.2. PEAR_Error growl handler

```
<?php
require_once 'Net/Growl/Autoload.php';
require_once 'PEAR.php';

define('GROWL_NOTIFY_PEARERROR', 'PEAR_Error');
```

```
function growlErrors($error)
{
    static $app;

    if (!isset($app)) {
        $app = new Net_Growl_Application(
            'Net_Growl', array(GROWL_NOTIFY_PEARERROR), 'mamasam'
        );
    }

    $growl = Net_Growl::singleton(
        $app, null, null, array('host' => '127.0.0.1')
    );
    $growl->notify(GROWL_NOTIFY_PEARERROR,
        get_class($error),
        $error->message.' in '.$_SERVER['SCRIPT_NAME'],
        array('sticky' => true)
    );
}

PEAR::setErrorHandling(PEAR_ERROR_CALLBACK, 'growlErrors');

PEAR::raiseError("The expected error you submitted does not exist");
?>
```

Net_Growl_Application::addGrowlNotifications

Synopsis

require_once ‘Net/Growl/Autoload.php’;

void **Net_Growl_Application::addGrowlNotifications**($notifications)

Description. Adds notifications supported by this application

Expected array format is:

```
<?php
 array('notification name' => array('option name' => 'option value'));
```

At the moment, only option name *enabled* is supported. Example:

```
<?php
 $notifications = array('Test Notification' => array('enabled' => true));
```

Parameter

array $notifications
Array of notifications to support

Throws. InvalidArgumentException

Return value. void

Net_Growl_Application::getGrowlNotifications

Synopsis

require_once 'Net/Growl/Autoload.php';

array **Net_Growl_Application::getGrowlNotifications**( )

Description. Returns the notifications accepted by Growl for this application

Expected array format is:

```
<?php
 array('notification name' => array('option name' => 'option value'));
```

At the moment, only option name *enabled* is supported. Example:

```
<?php
 $notifications = array('Test Notification' => array('enabled' => true));
 return $notifications;
```

Throws. no exceptions thrown

Return value. array - list of notifications type

Net_Growl_Application::setGrowlName

Synopsis

require_once 'Net/Growl/Autoload.php';

string **Net_Growl_Application::setGrowlName**($appName)

Description. Sets the application name for registration in Growl

Throws. InvalidArgumentException

Return value. void

Net_Growl_Application::getGrowlName

Synopsis

require_once 'Net/Growl/Autoload.php';

string **Net_Growl_Application::getGrowlName**( )

Description. Returns the application name for registration in Growl

Throws. no exceptions thrown

Return value. string — application name

Net_Growl_Application::setGrowlPassword

Synopsis

require_once 'Net/Growl/Autoload.php';

string **Net_Growl_Application::setGrowlPassword**($password)

Description. Sets the password to be used by Growl to accept notification packets

Throws. InvalidArgumentException

Return value. void

Net_Growl_Application::getGrowlPassword

Synopsis

require_once 'Net/Growl/Autoload.php';

string **Net_Growl_Application::getGrowlPassword**( )

Description. Returns the password to be used by Growl to accept notification packets

Throws. no exceptions thrown

Return value. string — password

Net_Growl_Application::setGrowlIcon

Synopsis

require_once 'Net/Growl/Autoload.php';

string **Net_Growl_Application::setGrowlIcon**($appIcon)

Description. Sets the application icon for registration in Growl

Throws. InvalidArgumentException

Return value. void

Net_Growl_Application::getGrowlIcon

Synopsis

require_once 'Net/Growl/Autoload.php';

string **Net_Growl_Application::getGrowlIcon**( )

Description. Returns the application icon for registration in Growl

Throws. no exceptions thrown

Return value. string — application icon binary data or empty if default image

6.4. Net_Growl_Exception Class

6.4.1. Synopsis

```
<?php
class Net_Growl_Exception extends Exception
{
}
```

6.4.2. Methods

Inherit all methods and properties from base Exception class. More details at http://www.php.net/manual/en/class.exception.php

6.5. Net_Growl_Udp Class

6.5.1. Synopsis

```
<?php
class Net_Growl_Udp
{
    /* methods */
    public object __construct(mixed $application [, array $notifications = array() [, strin
    public bool sendRegister();
    public bool sendNotify($name, $title, $description, $options);

}
```

6.5.2. Methods

Table 6.5. Net_Growl_Udp Methods

Name	Description
__construct	Constructs a new UDP adapter
sendRegister	Sends the REGISTER message type
sendNotify	Sends the NOTIFY message type

Net_Growl_Udp::__construct

Synopsis

require_once 'Net/Growl.php';

object **new Net_Growl_Udp**($application, $notifications = array(), $password = '', $options = array())

Description. Constructs a new UDP adapter

Parameter

mixed $application
Application name

array $notifications
List of notification types

string $password
(optional) Password for Growl

array $options
(optional) List of options :

- *host*, *port*, *protocol*, *timeout*
  - for Growl socket server
- *debug*
  - to know what data are sent and received.

Throws. no exceptions thrown

Return value. object - Net_Growl_Udp

Net_Growl_Udp::sendRegister

Synopsis

require_once 'Net/Growl.php';

bool **Net_Growl_Udp::sendRegister**( )

Description. Sends the REGISTER message type

Throws

Net_Growl_Exception
if remote server communication failure

Return value. true

Net_Growl_Udp::sendNotify

Synopsis

require_once 'Net/Growl.php';

bool **Net_Growl_Udp::sendNotify**( )

Description. Sends the NOTIFY message type

Throws

Net_Growl_Exception
if remote server communication failure

Return value. true

6.6. Net_Growl_Gntp Class

6.6.1. Synopsis

```
<?php
class Net_Growl_Gntp
{
    /* properties */
    private array $_passwordHashAlgorithm;

    /* methods */
```

```
    public object __construct(mixed $application [, array $notifications = array() [, strin
    public Net_Growl_Response sendRegister();
    public Net_Growl_Response sendNotify($name, $title, $description, $options);

    protected string genMessageStructure($method, $data [, $binaries = false]);

    private array _genKey($password);
    private array _genEncryption($key, $plainText);
    private string _toBool($value);
}
```

6.6.2. Methods

Table 6.6. Net_Growl_Gntp Methods

Name	Description
__construct	Constructs a new GNTP adapter
sendRegister	Sends the REGISTER message type
sendNotify	Sends the NOTIFY message type

Net_Growl_Gntp::__construct

Synopsis

require_once 'Net/Growl.php';

object **new Net_Growl_Gntp**($application, $notifications = array(), $password = '', $options = array())

Description. Constructs a new GNTP adapter

Parameter

mixed $application
Application name

array $notifications
List of notification types

string $password
(optional) Password for Growl

array $options
(optional) List of options :

- *host*, *port*, *protocol*, *timeout*

    - for Growl socket server
- *passwordHashAlgorithm, encryptionAlgorithm*
    - to secure communications
- *debug*
    - to know what data are sent and received.
- *resourceDir*
    - location of default icons; default to false, so use the @data_dir@ of PEAR
- *defaultIcon*
    - the default icon filename

Throws. no exceptions thrown

Return value. object - Net_Growl_Gntp

Net_Growl_Gntp::sendRegister

Synopsis

require_once 'Net/Growl.php';

Net_Growl_Response **Net_Growl_Gntp::sendRegister**( )

Description. Sends the REGISTER message type

Throws

Net_Growl_Exception
if remote server communication failure

Return value. Net_Growl_Response object

Net_Growl_Gntp::sendNotify

Synopsis

require_once 'Net/Growl.php';

Net_Growl_Response **Net_Growl_Gntp::sendNotify**( )

Description. Sends the NOTIFY message type

Throws

Net_Growl_Exception
if remote server communication failure

Return value. Net_Growl_Response object

6.7. Net_Growl_GntpMock Class

6.7.1. Synopsis

```
<?php
class Net_Growl_GntpMock
{
    /* properties */
    protected $responses = array();

    /* methods */
    public object __construct(mixed $application [, array $notifications = array() [, strin
    public Net_Growl_Response sendRegister();
    public Net_Growl_Response sendNotify($name, $title, $description, $options);
    public void addResponse($response)

    protected Net_Growl_Response sendRequest()
    protected Net_Growl_Response createResponseFromString($str)
    protected Net_Growl_Response createResponseFromFile($fp)
}
```

6.7.2. Methods

Table 6.7. Net_Growl_GntpMock Methods

Name	Description
__construct	Constructs a new GNTP Mock adapter
sendRegister	Mock sending the REGISTER message type
sendNotify	Mock sending the NOTIFY message type
addResponse	Adds response to the queue

Net_Growl_Gntp::__construct

Synopsis

require_once ‘Net/Growl.php’;

object **new Net_Growl_Gntp**($application, $notifications = array(), $password = '', $options = array())

Description. Constructs a new GNTP adapter

Parameter

mixed $application
Application name

array $notifications
List of notification types

string $password
(optional) Password for Growl

array $options
(optional) List of options :

- *host*, *port*, *protocol*, *timeout*
  - for Growl socket server
- *passwordHashAlgorithm*, *encryptionAlgorithm*
  - to secure communications
- *debug*
  - to know what data are sent and received.

Throws. no exceptions thrown

Return value. object - Net_Growl_GntpMock

Net_Growl_GntpMock::sendRegister

Synopsis

require_once 'Net/Growl.php';

Net_Growl_Response **Net_Growl_GntpMock::sendRegister**( )

Description. Mock sending the REGISTER message type

Throws

Net_Growl_Exception
if Net_Growl_Response not received

Return value. Net_Growl_Response object

Net_Growl_GntpMock::sendNotify

Synopsis

require_once 'Net/Growl.php';

Net_Growl_Response **Net_Growl_GntpMock::sendNotify**( )

Description. Mock sending the NOTIFY message type

Throws

Net_Growl_Exception
if Net_Growl_Response not received

Return value. Net_Growl_Response object

Net_Growl_GntpMock::addResponse

Synopsis

require_once 'Net/Growl.php';

void **Net_Growl_GntpMock::addResponse**($response)

Description. Adds response expected to the queue

Throws

Net_Growl_Exception
if $response is different to file pointer, string or Net_Growl_Exception

Return value. void

6.8. Net_Growl_Response Class

6.8.1. Synopsis

```
<?php
class Net_Growl_Response
{
    /* properties */
    protected string $version;
```

```
    protected string $code;
    protected string $action;
    protected integer $errorCode;
    protected string $errorDescription;
    protected string $machineName;
    protected string $softwareName;
    protected string $softwareVersion;
    protected string $platformName;
    protected string $platformVersion;
    protected string $body;

    /* methods */
    public object __construct(string $statusLine );
    public void appendBody(string $bodyChunk );
    public string getVersion();
    public string getStatus();
    public string getResponseAction();
    public integer getErrorCode();
    public string getErrorDescription();
    public string getOriginMachineName();
    public string getOriginSoftwareName();
    public string getOriginSoftwareVersion();
    public string getOriginPlatformName();
    public string getOriginPlatformVersion();
    public string __toString();

}
```

6.8.2. Methods

Table 6.8. Net_Growl_Response Methods

Name	Description
__construct	Constructs a new GNTP specialized response
appendBody	Append a string to the response body
getVersion	Returns GNTP protocol version
getStatus	Returns the status code
getResponseAction	Returns the request action
getErrorCode	Returns the error code
getErrorDescription	Returns the error description
getOriginMachineName	Returns the machine name/host name of the sending computer
getOriginSoftwareName	Returns the identity of the sending framework
getOriginSoftwareVersion	Returns the version of the sending framework
getOriginPlatformName	Returns the identify of the sending computer OS/platform
getOriginPlatformVersion	Returns the version of the sending computer OS/platform

Name	Description
__toString	Returns the String representation of the Growl response

Net_Growl_Response::__construct

Synopsis

require_once 'Net/Growl.php';

object **new Net_Growl_Response**($statusLine)

Description. Constructs a specialized response to a GNTP request

Parameter

mixed $statusLine
Response status line (e.g. "GNTP/1.0 -OK NONE")

Throws. no exceptions thrown

Return value. object - Net_Growl_Response

Net_Growl_Response::appendBody

Synopsis

require_once 'Net/Growl.php';

void **Net_Growl_Response::appendBody**( )

Description. Append a string to the response body excluding the protocol identifier, version, message type, and encryption algorithm id

Throws. no exceptions thrown

Return value. void

Net_Growl_Response::getVersion

Synopsis

require_once 'Net/Growl.php';

string **Net_Growl_Response::getVersion**( )

Description. Returns GNTP protocol version (e.g. 1.0, 1.1)

Throws. no exceptions thrown

Return value. string

Net_Growl_Response::getStatus

Synopsis

require_once 'Net/Growl.php';

string **Net_Growl_Response::getStatus**( )

Description. Returns the status code (OK | ERROR)

Throws. no exceptions thrown

Return value. string

Net_Growl_Response::getResponseAction

Synopsis

require_once 'Net/Growl.php';

string **Net_Growl_Response::getResponseAction**( )

Description. Returns the request action (REGITER | NOTIFY)

Throws. no exceptions thrown

Return value. string

Net_Growl_Response::getErrorCode

Synopsis

require_once 'Net/Growl.php';

int **Net_Growl_Response::getErrorCode**( )

Description. Returns the error code

Throws. no exceptions thrown

Return value. integer

Net_Growl_Response::getErrorDescription

Synopsis

require_once 'Net/Growl.php';

string **Net_Growl_Response::getErrorDescription**( )

Description. Returns the error description

Throws. no exceptions thrown

Return value. string

Net_Growl_Response::getOriginMachineName

Synopsis

require_once 'Net/Growl.php';

string **Net_Growl_Response::getOriginMachineName**( )

Description. Returns the machine name/host name of the sending computer

Throws. no exceptions thrown

Return value. string

Net_Growl_Response::getOriginSoftwareName

Synopsis

require_once 'Net/Growl.php';

string **Net_Growl_Response::getOriginSoftwareName**( )

Description. Returns the identity of the sending framework

- Example1: Growl/Win
- Example2: GrowlAIRConnector

Throws. no exceptions thrown

Return value. string

Net_Growl_Response::getOriginSoftwareVersion

Synopsis

require_once ‘Net/Growl.php’;

string **Net_Growl_Response::getOriginSoftwareVersion**( )

Description. Returns the version of the sending framework

- Example1: 2.0.0.28
- Example2: 1.2

Throws. no exceptions thrown

Return value. string

Net_Growl_Response::getOriginPlatformName

Synopsis

require_once ‘Net/Growl.php’;

string **Net_Growl_Response::getOriginPlatformName**( )

Description. Returns the identify of the sending computer OS/platform

- Example1: Microsoft Windows NT 5.1.2600 Service Pack 3
- Example2: Mac OS X

Throws. no exceptions thrown

Return value. string

Net_Growl_Response::getOriginPlatformVersion

Synopsis

require_once ‘Net/Growl.php’;

string **Net_Growl_Response::getOriginPlatformVersion**( )

Description. Returns the version of the sending computer OS/platform

- Example1: 5.1.2600.196608

- Example2: 10.6

Throws. no exceptions thrown

Return value. string

Net_Growl_Response::__toString

Synopsis

require_once ‘Net/Growl.php’;

string **Net_Growl_Response::__toString**( )

Description. Returns the String representation of the Growl response

- Example1: Response REGISTER OK (Growl/Win 2.0.0.28)
- Example2: Response ERROR 300 No notifications registered (Growl/Win 2.0.0.28)

Throws. no exceptions thrown

Return value. string

Chapter 7. License

The full legal text of the BSD 3-clause license is given below.

Glossary

GNTP
: Growl uses GNTP (Growl Notification Transport Protocol) [http://www.growlforwindows.com/gfw/help/gntp.aspx] to send notifications. GNTP is a MIME-like format.

Growl
: Growl [http://growl.info] is a global notification system for Mac OS X. Any application can send a notification to Growl, which will display an attractive message on your screen. Growl currently works with a growing number of applications.

Notification
: Notifications are a way for your applications to provide you with new information, without you having to switch from the application you’re already in.

UDP
: Growl uses on all platforms the basic UDP (User Datagram Protocol) [http://growl.info/documentation/developer/protocol.php] to send notifications. UDP is a binary format.