# Expect

# Introduction

This extension allows to interact with processes through PTY. You may consider using the *expect://* wrapper with the filesystem functions which provide a simpler and more intuitive interface.

# Installing/Configuring

## Requirements

This module uses the functions of the » expect library. You need libexpect version >= 5.43.0.

## Installation

This » PECL extension is not bundled with PHP. Information for installing this PECL extension may be found in the manual chapter titled Installation of PECL extensions. Additional information such as new releases, downloads, source files, maintainer information, and a CHANGELOG, can be located here: » http://pecl.php.net/package/expect.

In PHP 4 this PECL extensions source can be found in the *ext/* directory within the PHP source or at the PECL link above. In order to use these functions you must compile PHP with expect support by using the *--with-expect[=DIR]* configure option.

Windows users will enable *php_expect.dll* inside of *php.ini* in order to use these functions. In PHP 4 this DLL resides in the *extensions/* directory within the PHP Windows binaries download. The DLL for this PECL extension may be downloaded from either the » PHP Downloads page or from » http://pecl4win.php.net/

## Runtime Configuration

The behaviour of these functions is affected by settings in *php.ini*.

In order to configure expect extension, there are configuration options in the configuration file *php.ini*.

### Expect Configure Options

| Name | Default | Changeable | Changelog |
|---|---|---|---|
| expect.timeout | "10" | PHP_INI_ALL | |
| expect.loguser | "1" | PHP_INI_ALL | |
| expect.logfile | "" | PHP_INI_ALL | |

For further details and definitions of the PHP_INI_* constants, see the php.ini directives.

Here's a short explanation of the configuration directives.

*expect.timeout* integer
: The timeout period for waiting for the data, when using the expect_expectl() function. A value of "-1" disables a timeout from occurring.

| **Note** |
| --- |
| A value of "0" causes the expect_expectl() function to return immediately. |

*expect.loguser* boolean
: Whether expect should send any output from the spawned process to stdout. Since interactive programs typically echo their input, this usually suffices to show both sides of the conversation.

*expect.logfile* string
: Name of the file, where the output from the spawned process will be written. If this file doesn't exist, it will be created.

| **Note** |
| --- |
| If this configuration is not empty, the output is written regardless of the value of expect.loguser. |

## Resource Types

expect_popen() returns an open PTY stream used by expect_expectl().

# Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

**EXP_GLOB** ( integer )
Indicates that the pattern is a glob-style string pattern.

**EXP_EXACT** ( integer )
Indicates that the pattern is an exact string.

**EXP_REGEXP** ( integer )
Indicates that the pattern is a regexp-style string pattern.

**EXP_EOF** ( integer )
Value, returned by expect_expectl(), when EOF is reached.

**EXP_TIMEOUT** ( integer )
Value, returned by expect_expectl() upon timeout of seconds, specified in value of expect.timeout

**EXP_FULLBUFFER** ( integer )
Value, returned by expect_expectl() if no pattern have been matched.

# Examples

## Expect Usage Examples

**Example #1 - Expect Usage Example**

This example connects to the remote host via SSH, and prints the remote uptime.

```
<?php
ini_set("expect.loguser", "Off");

$stream = fopen("expect://ssh root@remotehost uptime", "r");

$cases = array (
   array (0 => "password:", 1 => PASSWORD)
);

switch (expect_expectl ($stream, $cases)) {
   case PASSWORD:
       fwrite ($stream, "password\n");
       break;

   default:
       die ("Error was occurred while connecting to the remote host!\n");
}

while ($line = fgets($stream)) {
     print $line;
}
fclose ($stream);
?>
```

The following example connects to the remote host, determines whether installed OS is for 32 or 64 bit, then runs update for specific package.

**Example #2 - Another Expect Usage Example**

```
<?php
ini_set("expect.timeout", -1);
ini_set("expect.loguser", "Off");

$stream = expect_popen("ssh root@remotehost");

while (true) {
   switch (expect_expectl ($stream, array (
           array ("password:", PASSWORD), // SSH is asking for password
           array ("yes/no)?", YESNO), // SSH is asking whether to store the
host entry
           array ("~$ ", SHELL, EXP_EXACT), // We've got the shell!
   ))) {
       case PASSWORD:
```

```
            fwrite ($stream, "secret\n");
            break;

        case YESNO:
            fwrite ($stream, "yes\n");
            break;

        case SHELL:
            fwrite ($stream, "uname -a\n");
            while (true) {
                    switch (expect_expectl ($stream, array (
                            array ("~$ ", SHELL, EXP_EXACT), // We've got the
shell!
                            array ("^Linux.*$", UNAME, EXP_REGEXP), // uname
-a output
                    ), $match)) {
                        case UNAME:
                            $uname .= $match[0];
                            break;

                        case SHELL:
                            // Run update:
                            if (strstr ($uname, "x86_64")) {
                                    fwrite ($stream, "rpm -Uhv
http://mirrorsite/somepath/some_64bit.rpm\n");
                            } else {
                                    fwrite ($stream, "rpm -Uhv
http://mirrorsite/somepath/some_32bit.rpm\n");
                            }
                            fwrite ($stream, "exit\n");
                            break 2;

                        case EXP_TIMEOUT:
                        case EXP_EOF:
                            break 2;

                        default:
                            die ("Error has occurred!\n");
                    }
            }
            break 2;

        case EXP_TIMEOUT:
        case EXP_EOF:
            break 2;

        default:
            die ("Error has occurred!\n");
    }
}

fclose ($stream);
?>
```

# Expect Functions

# expect_expectl

expect_expectl -- Waits until the output from a process matches one of the patterns, a specified time period has passed, or an EOF is seen

## Description

int **expect_expectl** ( resource `$expect`, array `$cases` [, array `&$match` ] )

Waits until the output from a process matches one of the patterns, a specified time period has passed, or an EOF is seen.

If *`match`* is provided, then it is filled with the result of search. The matched string can be found in *`match[0]`*. The match substrings (according to the parentheses) in the original pattern can be found in *`match[1]`*, *`match[2]`*, and so on, up to *`match[9]`* (the limitation of libexpect).

## Parameters

*`expect`*
An Expect stream, previously opened with expect_popen().

*`cases`*
An array of expect cases. Each expect case is an indexed array, as described in the following table:

**Expect Case Array**

| Index Key | Value Type | Description | Is Mandatory | Default Value |
|---|---|---|---|---|
| 0 | string | pattern, that will be matched against the output from the stream | yes | |
| 1 | mixed | value, that will be returned by this function, if the pattern matches | yes | |
| 2 | integer | pattern type, one of: **EXP_GLOB**, **EXP_EXACT** or **EXP_REGEXP** | no | **EXP_GLOB** |

## Return Values

Returns value associated with the pattern that was matched.

On failure this function returns: **EXP_EOF**, **EXP_TIMEOUT** or **EXP_FULLBUFFER**

## ChangeLog

| Version | Description |
|---|---|
| 0.2.1 | Prior to version 0.2.1, in *match* parameter a match string was returned, not an array of match substrings. |

## Examples

**Example #3 - expect_expectl() example**

```
<?php
// Copies file from remote host:
ini_set ("expect.timeout", 30);

$stream = fopen ("expect://scp user@remotehost:/var/log/messages
/home/user/messages.txt", "r");

$cases = array (
 array (0 => "password:", 1 => PASSWORD),
 array (0 => "yes/no)?", 1 => YESNO)
);

while (true) {
switch (expect_expectl ($stream, $cases))
{
 case PASSWORD:
  fwrite ($stream, "password\n");
  break;

 case YESNO:
  fwrite ($stream, "yes\n");
  break;

 case EXP_TIMEOUT:
 case EXP_EOF:
  break 2;

 default:
  die ("Error has occurred!\n");
}
}
```

```
fclose ($stream);
?>
```

## See Also

- expect_popen()

# expect_popen

expect_popen -- Execute command via Bourne shell, and open the PTY stream to the process

## Description

resource **expect_popen** ( string `$command` )

Execute command via Bourne shell, and open the PTY stream to the process.

## Parameters

*`command`*
Command to execute.

## Return Values

Returns an open PTY stream to the process'es stdio, stdout and stderr.

On failure this function returns **FALSE**.

## Examples

**Example #4 - expect_popen() example**

```
<?php
// Login to the PHP.net CVS repository:
$stream = expect_popen ("cvs -d :pserver:anonymous@cvs.php.net:/repository
login");
sleep (3);
fwrite ($stream, "phpfi\n");
fclose ($stream);
?>
```

## See Also

- popen()