win32service

Introduction

The win32service extension is a Windows specific extension that allows PHP to communicate with the Service Control Manager to start, stop, register and unregister services, and even allows your PHP scripts to run as a service.

Installing/Configuring

Requirements

Windows NT, Windows 2000, Windows XP or Windows Server 2003. Any version of Windows derived from Windows NT should be compatible.

Installation

Installing from PECL

1. You can download php_win32service.dll from http://snaps.php.net/win32/. Choose the PECL_X_X folder that matches you PHP version.
2. Copy the php_win32service.dll into your extension_dir.
3. Load the extension from your *php.ini*

```
extension=php_win32service.dll
```

Runtime Configuration

This extension has no configuration directives defined in *php.ini*.

Resource Types

This extension has no resource types defined.

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.

WIN32_SERVICE_CONTROL_CONTINUE (integer)

WIN32_SERVICE_CONTROL_INTERROGATE (integer)

WIN32_SERVICE_CONTROL_PAUSE (integer)

WIN32_SERVICE_CONTROL_STOP (integer)

WIN32_SERVICE_CONTROL_HARDWAREPROFILECHANGE (integer)

WIN32_SERVICE_CONTROL_POWEREVENT (integer)

WIN32_SERVICE_CONTROL_SESSIONCHANGE (integer)

WIN32_ERROR_CALL_NOT_IMPLEMENTED (integer)

WIN32_NO_ERROR (integer)

WIN32_SERVICE_RUNNING (integer)

WIN32_SERVICE_STOPPED (integer)

WIN32_SERVICE_STOP_PENDING (integer)

WIN32_SERVICE_WIN32_OWN_PROCESS (integer)

WIN32_SERVICE_INTERACTIVE_PROCESS (integer)

WIN32_SERVICE_STOPPED (integer)

WIN32_SERVICE_START_PENDING (integer)

WIN32_SERVICE_STOP_PENDING (integer)

WIN32_SERVICE_RUNNING (integer)

WIN32_SERVICE_CONTINUE_PENDING (integer)

WIN32_SERVICE_PAUSE_PENDING (integer)

WIN32_SERVICE_PAUSED (integer)

WIN32_SERVICE_ACCEPT_NETBINDCHANGE (integer)

WIN32_SERVICE_ACCEPT_PARAMCHANGE (integer)

WIN32_SERVICE_ACCEPT_PAUSE_CONTINUE (integer)

WIN32_SERVICE_ACCEPT_SHUTDOWN (integer)

WIN32_SERVICE_ACCEPT_STOP (integer)

WIN32_SERVICE_ACCEPT_HARDWAREPROFILECHANGE (integer)

WIN32_SERVICE_ACCEPT_POWEREVENT (integer)

WIN32_SERVICE_ACCEPT_SESSIONCHANGE (integer)

WIN32_SERVICE_FILE_SYSTEM_DRIVER (integer)

WIN32_SERVICE_KERNEL_DRIVER (integer)

WIN32_SERVICE_WIN32_SHARE_PROCESS (integer)

WIN32_SERVICE_RUNS_IN_SYSTEM_PROCESS (integer)

Examples

Example #1 - Registering a PHP script to run as a service

```
<?php
win32_create_service(array(
   'service' => 'dummyphp',                 # the name of your service
   'display' => 'sample dummy PHP service', # description
   'params' => 'c:\path\to\script.php run', # path to the script and
parameters
));
?>
```

Example #2 - Unregistering a service

```
<?php
win32_delete_service('dummyphp');
?>
```

Example #3 - Running as a service

```
<?php
if ($argv[1] == 'run') {
 win32_start_service_ctrl_dispatcher('dummyphp');

 while (WIN32_SERVICE_CONTROL_STOP != win32_get_last_control_message()) {
   # do your work here.
   # try not to take up more than 30 seconds before going around the loop
   # again
 }
}
?>
```

win32service Functions

win32_create_service

win32_create_service -- Creates a new service entry in the SCM database

Description

mixed **win32_create_service** (array `$details` [, string `$machine`])

Parameters

`details`
An array of service details:
`service`
The short name of the service. This is the name that you will use to control the service using the *net* command. The service must be unique (no two services can share the same name), and, ideally, should avoid having spaces in the name.

`display`
The display name of the service. This is the name that you will see in the Services Applet.

`user`
The name of the user account under which you want the service to run. If omitted, the service will run as the LocalSystem account. If the username is specified, you must also provide a password.

`password`
The password that corresponds to the *`user`*.

`path`
The full path to the executable module that will be launched when the service is started. If omitted, the path to the current PHP process will be used.

`params`
Command line parameters to pass to the service when it starts. If you want to run a PHP script as the service, then the first parameter should be the full path to the PHP script that you intend to run.

`load_order`
Controls the load_order. This is not yet fully supported.

`svc_type`
Sets the service type. If omitted, the default value is **WIN32_SERVICE_WIN32_OWN_PROCESS**. Don't change this unless you know what you're doing.

`start_type`
Specifies how the service should be started. The default is

WIN32_SERVICE_AUTO_START which means the the service will be launched when the machine starts up.

error_control
Informs the SCM what it should do when it detects a problem with the service. The default is **WIN32_SERVER_ERROR_IGNORE**. Changing this value is not yet fully supported.

machine
The optional machine name on which you want to create a service. If omitted, it will use the local machine.

Return Values

Returns **TRUE** on success, otherwise returns a win32 error code.

Examples

Example #4 - A win32_create_service() example

Any text that describes the purpose of the example, or what goes on in the example should go here (inside the

```
<?php
$x = win32_create_service(array(
       'service' => 'dummyphp',
       'display' => 'sample dummy PHP service',
       'params' => __FILE__ . ' run',
));
debug_zval_dump($x);
?>
```

See Also

- win32_delete_service()

win32_delete_service

win32_delete_service -- Deletes a service entry from the SCM database

Description

int **win32_delete_service** (string `$servicename` [, string `$machine`])

Attempts to delete a service from the SCM database. Administrative privileges are required for this to succeed.

This function really just marks the service for deletion. If other processes (such as the Services Applet) are open, then the deletion will be deferred until those applications are closed. If a service is marked for deletion, further attempts to delete it will fail, and attempts to create a new service with that name will also fail.

Parameters

`servicename`
The short name of the service.

`machine`
The optional machine name. If omitted, the local machine will be used.

Return Values

Returns **TRUE** on success, or a win32 error code on failure.

Examples

Example #5 - A win32_delete_service() example

Deletes the dummyphp service.

```
<?php
win32_delete_service('dummyphp');
?>
```

win32_get_last_control_message

win32_get_last_control_message -- Returns the last control message that was sent to this service

Description

int **win32_get_last_control_message** (void)

Returns the control code that was last sent to this service process. When running as a service you should periodically check this to determine if your service needs to stop running.

Return Values

Returns a control constant; one of **WIN32_SERVICE_CONTROL_CONTINUE**, **WIN32_SERVICE_CONTROL_INTERROGATE**, **WIN32_SERVICE_CONTROL_PAUSE**, **WIN32_SERVICE_CONTROL_STOP**, **WIN32_SERVICE_CONTROL_HARDWAREPROFILECHANGE**, **WIN32_SERVICE_CONTROL_POWEREVENT**, **WIN32_SERVICE_CONTROL_SESSIONCHANGE**.

See Also

- win32_start_service_ctrl_dispatcher()

win32_query_service_status

win32_query_service_status -- Queries the status of a service

Description

mixed **win32_query_service_status** (string `$servicename` [, string `$machine`])

Queries the current status for a service, returning an array of information.

Parameters

`servicename`
The short name of the service.

`machine`
The optional machine name. If omitted, the local machine will be used.

Return Values

Returns **FALSE** on failure, otherwise returns an array consisting of the following information:

`ServiceType`
The dwServiceType.

`CurrentState`
The dwCurrentState.

`ControlsAccepted`
Which service controls are accepted by the service.

`Win32ExitCode`
If the service exited, the return code from the process.

`ServiceSpecificExitCode`
If the service exited with an error condition, the service specific code that is logged in the event log is visible here.

`CheckPoint`
If the service is shutting down, holds the current check point number. This is used by the SCM as a kind of heart-beat to detect a wedged service process. The value of the check point is best interpreted in conjunction with the WaitHint value.

`WaitHint`
If the service is shutting down it will set WaitHint to a checkpoint value that will indicate

100% completion. This can be used to implement a progress indicator.

ProcessId
The Windows process identifier. If 0, the process is not running.

ServiceFlags
The dwServiceFlags.

win32_set_service_status

win32_set_service_status -- Update the service status

Description

bool **win32_set_service_status** (int `$status`)

Informs the SCM of the current status of a running service. This call is only valid for a running service process.

Parameters

`status`
: The service status code, one of **WIN32_SERVICE_RUNNING**, **WIN32_SERVICE_STOPPED**, **WIN32_SERVICE_STOP_PENDING**, **WIN32_SERVICE_START_PENDING**, **WIN32_SERVICE_CONTINUE_PENDING**, **WIN32_SERVICE_PAUSE_PENDING**, **WIN32_SERVICE_PAUSED**.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- win32_start_service_ctrl_dispatcher()

win32_start_service_ctrl_dispatcher

win32_start_service_ctrl_dispatcher -- Registers the script with the SCM, so that it can act as the service with the given name

Description

mixed **win32_start_service_ctrl_dispatcher** (string `$name`)

When launched via the Service Control Manager, a service process is required to "check-in" with it to establish service monitoring and communication facilities. This function performs the check-in by spawning a thread to handle the lower-level communication with the service control manager.

Once started, the service process should continue to check-in with the service control manager so that it can determine if it should terminate. This is achieved by periodically calling win32_get_last_control_message() and handling the return code appropriately.

Parameters

name
The short-name of the service, as registered by win32_create_service().

Return Values

Returns **TRUE** on success, otherwise **FALSE** or a win32 error code.

Examples

Example #6 - A win32_start_service_ctrl_dispatcher() example

Any text that describes the purpose of the example, or what goes on in the example should go here (inside the

```
<?php
if (!win32_start_service_ctrl_dispatcher('dummyphp')) {
 die("I'm probably not running under the service control manager");
}

while (WIN32_SERVICE_CONTROL_STOP != win32_get_last_control_message()) {
 # do some work here, trying not to take more than around 30 seconds
 # before coming back into the loop again
}
?>
```

See Also

- win32_get_last_control_message()

win32_start_service

win32_start_service -- Starts a service

Description

int **win32_start_service** (string `$servicename` [, string `$machine`])

Attempts to start the named service. Usually requires administrative privileges.

Parameters

`servicename`
: The short name of the service.

`machine`
: Optional machine name. If omitted, the local machine is used.

Return Values

Returns **WIN32_NO_ERROR** on success, or some other win32 error code on failure.

See Also

- win32_stop_service()

win32_stop_service

win32_stop_service -- Stops a service

Description

int **win32_stop_service** (string `$servicename` [, string `$machine`])

Stops a named service. Requires administrative privileges.

Parameters

`servicename`
: The short name of the service.

`machine`
: Optional machine name. If omitted, the local machine is used.

Return Values

Returns **WIN32_NO_ERROR** on success, or a win32 error code on failure.

See Also

- win32_start_service()